



UNIVERSIDAD DE JAÉN
Nombre del Centro

Trabajo Fin de Grado

INTEGRACIÓN EN APLICACIONES DE MAPAS DE MAPAS DE VISIBILIDAD

Alumno: Manuel Ruano Peinado

Tutor: Rafael Jesús Segura Sánchez
Dpto: Informática

Febrero, 2017



Universidad de Jaén
Escuela Politécnica Superior de Jaén
Departamento de Informática

Don Rafael Segura Sánchez, tutor del Proyecto Fin de Carrera titulado: Mapas de Mapas de Visibilidad, que presenta Manuel Ruano Peinado, autoriza su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, FEBRERO de 2017

El alumno:

Manuel Ruano Peinado

Los tutores:

Rafael Segura Sánchez

Agradecimientos

Con la finalización de este trabajo termina una etapa muy importante en mi vida pero a su vez se abre otra mucho más importante.

Me gustaría dedicar este Trabajo Fin de Grado a todo aquel que ha aportado algo a lo largo de todo este camino del que forman parte familiares, amigos, profesores y conocidos que, aunque no se han dado cuenta, entre todos han construido ese camino que nos lleva hasta este final.

En especial a mis padres, ya que sin su apoyo y sacrificio no hubiese podido llegar donde me encuentro hoy.

A mis compañeros y amigos, con los que he compartido muchísimos buenos momentos tanto dentro como fuera de la universidad.

Muchísimas gracias a todos.

Índice

1.	Introducción al problema	5
1.1.	Funcionamiento de las Atalayas.....	7
1.2.	Descripción del problema.....	8
1.3.	Motivación.....	11
1.4.	Objetivos	11
1.5.	Estructura del documento	12
2.	Gestión y Planificación	13
2.1.	Planificación del Proyecto	13
2.1.1.	Diagrama de Grant.....	15
2.1.2.	Diagrama Pert.....	16
2.2.	Análisis de costes	18
3.	Análisis.....	22
3.1.	Requisitos	22
3.1.1.	Requisitos Funcionales	22
3.1.2.	Requisitos No Funcionales.....	22
3.1.3.	Requisitos del Cliente.....	23
3.1.4.	Requisitos del Servidor	23
3.2.	Mapas	25
3.3.	Lenguaje de Programación	28
3.4.	Servidores, Servicios y Otros	28
3.5.	Librerías	28
4.	Diseño.....	29
4.1.	Diagrama de clases.....	29
4.2.	Diagrama de Casos de Uso	31
4.3.	Diagrama de Actividad	32
4.4.	Patrón Modelo-Vista-Controlador	33
4.5.	Diagramas de secuencia.....	35
4.5.1.	Carga del mapa.....	35
4.5.2.	Generar capa de visibilidad en un punto	36
4.5.3.	Crear Nuevo Marcador	37
4.5.4.	Guardar marcador	38
4.5.5.	Calcular la capa de visibilidad desde un marcador no almacenado	39

4.5.6.	Calcular visibilidad de los marcadores desde uno no almacenado	40
4.5.7.	Eliminar marcador no almacenado	41
4.5.8.	Abrir infoWindow marcador	42
4.5.9.	Calcular la visibilidad de un marcador almacenado	43
4.5.10.	Calcular la capa de visibilidad desde un marcador almacenado	44
4.5.11.	Eliminar un marcador	45
4.6.	StoryBoards	46
5.	Implementación	50
5.1.	Servidor WAMP	50
5.2.	Google Maps JavaScript API	51
5.2.1.	Clave de Google Maps	51
5.2.2.	Creación del Mapa	53
5.2.3.	Google Maps Elevation	53
5.2.4.	Google Maps Markers	55
5.2.5.	Superposiciones de Imágenes	56
5.3.	Funciones	57
5.3.1.	Cálculo de la capa de visibilidad	57
5.3.2.	Función cargarVisibilidad	59
5.3.3.	Visibilidad de los POI	61
5.3.4.	Otras Funciones y Características	62
6.	Instalación y Configuración del Servidor	65
6.1.	Instalación	65
7.	Conclusiones y Posibles Mejoras	71
	Bibliografía	76
	Anexo 1. Comprobación de los datos obtenidos mediante la aplicación	79

1. Introducción al problema

Las atalayas (del árabe hispánico **attaláya'**) o torres de vigilancia se construían estratégicamente en puntos donde la visibilidad era muy amplia, generalmente en líneas de frontera o asociadas a caminos transitables. No es habitual que estén aisladas ya que funcionan a como de eslabones de una cadena.



Ilustración 1- Torre de la Nava

Su localización suele ser en montañas elevadas o extensas llanuras, que permiten una amplia visualización del contorno, además de la visualización de las atalayas colindantes.

La función de estas torres no era la de combatir al enemigo, ya que apenas tenían resistencia, si no la importancia de la comunicación ante una alarma de intrusiones enemigas. De este modo y mediante su situación estratégica podían prepararse ante un ataque poniendo a los civiles y ganado bajo las murallas de sus fortalezas.

El conocer la ubicación de las atalayas y otros elementos defensivos es fundamental para comprender el territorio. Desgraciadamente, muchas de estas construcciones defensivas se han perdido con el paso de los años, bien por deterioro debido a la falta de utilización, o bien porque han sido directamente integradas dentro de otras construcciones con el fin de dar uso.



Ilustración 2 - Torre de la Aldehuella

El emplazamiento de estas construcciones de carácter defensivo, comprenden diferentes patrones, que normalmente dependen de la geografía del terreno, obteniendo una mayor cantidad de información, desde zonas más elevadas, de lo que sucede alrededor.

El estudio de la cuenca visual no solo se utiliza para fines políticos o militares, si no que a día de hoy puede ser muy útil en diferentes ámbitos, tales como guardas forestales, impacto ambiental, restauración paisajística y estudios sobre el paisaje en general.

1.1. Funcionamiento de las Atalayas

Los torreros eran los encargados de la vigilancia, la guardia era normalmente de dos o tres hombres. En caso de avistar algún peligro se hacían señales de humo si era de día, llamadas “ahumada”, o se encendía una hoguera en la parte superior de la torre si era de noche. Así se informaban sucesivamente hasta que la alerta llegaba a la Fortaleza más cercana, en la que se daba la alerta y se reunían las tropas. Si el ataque era amplio el mensaje se seguía difundiendo hasta otra fortaleza mayor. A la vez que se daba la alarma, desde la torre salía un torrero en caballo, en dirección a la Fortaleza.



Ilustración 3 - Interior Atalaya

Para comunicarse se hacían señales de humo durante el día o fuego durante la noche. Con un sencillo código similar al del Morse, transmitían la información. Así, la noticia de algún peligro podría recorrer de atalaya en atalaya 200km en una hora.

Los centinelas eran soldados jóvenes con muy buena vista. Debían permanecer durante muchos días en el interior en retenes de dos para tumbarse. Debía ser gente de una pasta especial para sobrellevar el aislamiento. Muchas veces se destinaba aquí a soldados como castigo.

El interior de la atalaya tenía dos o tres plantas en las que se almacenaban víveres o leña para las fogatas.

El centinela accedía por una puerta a cierta altura, después guardaba la escalera, eso dificultaba el acceso a enemigos a la atalaya. Si algún intruso pretendía entrar le lanzaban piedras desde lo alto.

1.2. Descripción del problema

Este trabajo fin de grado, de ahora en adelante TFG, consiste en crear un **prototipo** aplicación para integrar los mapas de visibilidad calculados a partir de la ubicación de las atalayas en aplicaciones de mapas.

Este TFG se basa en un **rediseño** del TFG realizado por José Morón Rodriguez (Integración en aplicaciones de mapas de mapas de visibilidad, Septiembre 2015).

José Morón utilizó herramientas como OpenLayers, que se trata de una librería de JavaScript, para el manejo de capas de mapas. Esta librería permite la utilización de diferentes tipos de capas, como son: Google Maps, Bing, OSM, MapQuest, etc.

El trabajo de José Morón trata de superponer la capa de visibilidad de unos puntos predefinidos anteriormente, estos puntos son creados por el administrador. Para obtener la capa de visibilidad, el administrador debe ejecutar el programa “**gvSig**”, programa para el manejo de información geográfica, e introducir en él un modelo digital del terreno, mediante el cual el programa es capaz de obtener la capa de visibilidad.

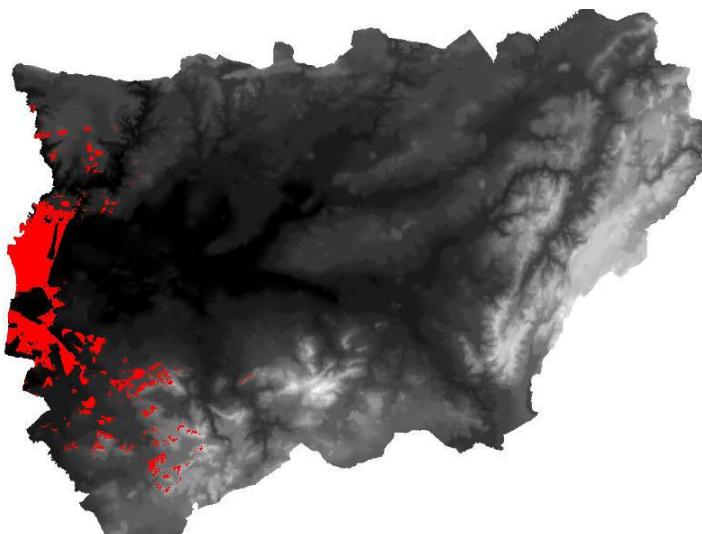


Ilustración 4 - Visibilidad obtenida mediante el TFG de José Morón

Una vez se obtiene la capa de visibilidad es necesario exportarla en un formato imagen, y añadirla al mapa manualmente.

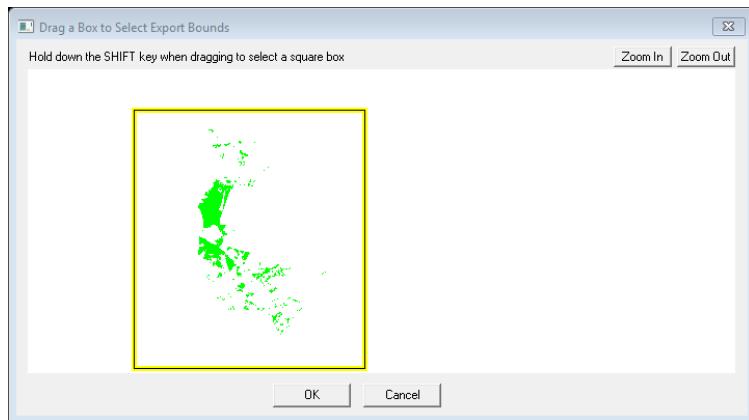


Ilustración 5 -- Visibilidad obtenida mediante el TFG de José Morón. Generación del mapa de bits

Quedando el resultado final así:

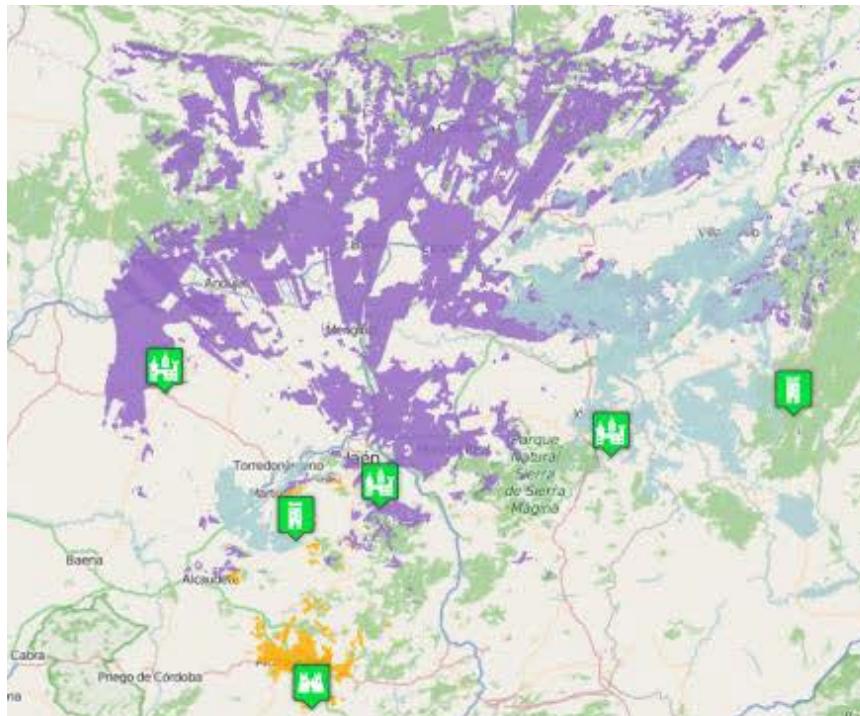


Ilustración 6 - - Visibilidad obtenida mediante el TFG de José Morón
: Integración en la aplicación de mapas.

Esta tarea de generar la capa de visibilidad es bastante compleja, ya que supone un uso avanzado del programa “**gvSig**”. Además, puede haber muchos puntos en los que calcular la visibilidad, llegando a ser un trabajo casi infinito para el administrador.

En este caso, con las herramientas utilizadas por José Morón, se hace muy difícil la unificación de las herramientas, ya que, el uso de OpenLayers, implica la utilización de capas de mapas de las que solo se pueden obtener datos como localizaciones en latitud y longitud, puntos de información como tiendas, restaurantes, gasolineras, carreteras, etc.

Por tanto no se dispone de ningún medio que nos pueda ayudar al cálculo de la visibilidad. Esto nos obliga a tener que buscar alternativas de mapas a los usados por José Morón.

En este trabajo se tratará de dar solución a este problema, se tratará de usar los recursos unificados en una única aplicación con la que poder obtener la visibilidad de una zona, o de diferentes puntos. Eliminando el arduo trabajo del administrador de crear cada una de las capas de visibilidad a mano para insertarlas en el mapa posteriormente.

Además, también se tratará de resolver la restricción que tiene el usuario a la hora de ver la visibilidad de un punto, ya que éste sólo tiene acceso a aquellos puntos creados con anterioridad por el administrador, no pudiendo, el mismo, elegir nuevos puntos.

1.3. Motivación

Este TFG está dentro de la rama Tecnologías de la Información, ya que se utilizan diferentes herramientas y servicios para el uso de los mapas mediante web, además está dentro de la mención de Sistemas Gráficos ya que se utilizan estructuras de datos de imágenes para almacenar la capa de visibilidad.

Además, otro motivo por el que he elegido este TFG es la abundancia de Atalayas por mi pueblo, Castillo de Locubín, donde desde pequeño he visitado y escuchado muchas historias acerca de ellas.

Este trabajo me ha permitido aprender más sobre ellas, además de localizar casi el total de las Atalayas de los alrededores, de las cuales he visitado algunas para comprobar la veracidad de los datos generados por la aplicación desarrollada.

1.4. Objetivos

El objetivo principal de este trabajo es llevar los mapas de visibilidad a una única aplicación, que unifique las tareas computacionales, haciendo así la experiencia del usuario final más gratificante. En este trabajo se generarán tanto capas de visibilidad, como la visibilidad de puntos concretos.

Para llevar a cabo este desarrollo será necesario analizar los requisitos e implementar una solución que los satisfaga.

Para desarrollar este proyecto se necesitará:

- Investigar acerca de donde obtener los datos para generar las capas de visibilidad.
- Diferentes mapas en los que integrar las capas de visibilidad, así como librerías.
- Lenguaje más cómodo en el que implementar el desarrollo basándonos en los datos obtenidos anteriormente.
- Diferentes tipos de servicio/servidores necesarios para el desarrollo.
- Implementación de la aplicación.

1.5. Estructura del documento

La realización de este trabajo se divide en varias partes:

Una primera parte sería la gestión y planificación del proyecto, en este apartado se tratará de documentar cómo se organizará el desarrollo de este proyecto, así como los costes de este.

En la parte del análisis hablaremos de los diferentes requisitos de nuestro proyecto, además estudiaremos los distintos tipos de mapas que podemos utilizar y las herramientas que se utilizarán para el desarrollo de este.

En la parte del diseño se desarrollarán los diferentes diagramas como de clases, casos de uso, actividad o de secuencia.

En la parte de la implementación se tratará de explicar las diferentes herramientas utilizadas, y una explicación más concisa de le desarrollo del proyecto. En este apartado se explicarán los códigos más relevantes.

En el apartado de instalación del servidor, se explicará paso a paso las pautas para la configuración de nuestra aplicación.

Por último, se darán unas conclusiones y mejoras acerca de este proyecto.

2. Gestión y Planificación

En este capítulo se tratará de planificar, localizar en el tiempo y documentar las diferentes tareas en las que se basa este proyecto, además se calculará el coste de este.

2.1. Planificación del Proyecto

Este proyecto se comenzó el día 5 de Septiembre de 2016 y se ha terminado el día 8 de Enero de 2017, por lo que se ha estado desarrollando durante 126 días, durante los cuales se han trabajado unas 3 horas diarias, alcanzando así y superando las entre 300 y 360 horas obligatorias de este Trabajo Fin de Grado

Para este proyecto se ha utilizado una metodología incremental, de este modo se han dividido los problemas en sub-problemas más sencillos y en los que cada uno forma parte de un incremento. Cada incremento es indispensable para poder continuar con el siguiente, una vez terminado un incremento se comprueba que todo funciona correctamente y se pasa al siguiente.

Por lo tanto se han dividido las tareas de la siguiente manera:

Sub-problemas	Duración (Horas)
Obtener Distancia	15
Obtener Altura	96
Obtener Visibilidad	51
Dibujar Visibilidad	18
POIS	18
Visibilidad POIS	15
Total	213

Tabla 1 - Subproblemas

En la siguiente tabla podemos ver más detalladamente cada uno de las actividades, incluyendo el tiempo de cada una:

Tarea	Duración (Días)	Horas diarias	Duración (Horas)	Fecha Inicio	Fecha Fin
Búsqueda Bibliográfica	5	2.5	12.5	5/09/16	9/09/16
Análisis	12	2.5	30	10/09/16	21/09/16
Elección de las Herramientas	23	2.5	57.5	22/09/16	14/10/16
Estudio de OpenLayers	15	2.5	37.5	22/09/16	6/10/16
Google Maps API	13	2.5	32.5	7/10/16	13/10/16
Instalación del Servidor	1	2.5	2.5	14/10/16	14/10/16
Desarrollo	72	2.5	180	15/10/16	25/12/16
Obtener Distancia	6	2.5	15	15/10/16	20/10/16
Análisis	3	2.5	7.5	15/10/16	17/10/16
Diseño	2	2.5	5	18/10/16	19/10/16
Implementación	1	2.5	2.5	20/10/16	20/10/16
Obtener Altura	32	2.5	80	21/10/16	21/11/16
Análisis	14	2.5	35	21/10/16	3/11/16
Diseño	14	2.5	35	4/11/16	17/11/16
Implementación	4	2.5	10	18/11/16	21/11/16
Obtener Visibilidad	17	2.5	42.5	22/11/16	8/12/16
Análisis	6	2.5	15	22/11/16	27/11/16
Diseño	6	2.5	15	28/11/16	3/12/16
Implementación	5	2.5	12.5	4/12/16	8/12/16
Dibujar Visibilidad	6	2.5	15	9/12/16	14/12/16
Análisis	2	2.5	5	9/12/16	10/12/16
Diseño	2	2.5	5	11/12/16	12/12/16
Implementación	2	2.5	5	13/12/16	14/12/16
POIS	6	2.5	15	15/12/16	20/12/16
Análisis	2	2.5	5	15/12/16	16/12/16
Diseño	2	2.5	5	17/12/16	18/12/16
Implementación	2	2.5	5	19/12/16	20/12/16
Visibilidad POIS	5	2.5	12.5	21/12/16	25/12/16
Análisis	2	2.5	5	21/12/16	22/12/16
Diseño	1	2.5	2.5	23/12/16	23/12/16
Implementación	2	2.5	5	24/12/16	25/12/16
Documentación	126	0.77	98	5/09/16	8/01/17
Total	126	3	378	5/09/16	8/01/17

Tabla 2 - Planificación

2.1.1. Diagrama de Grant

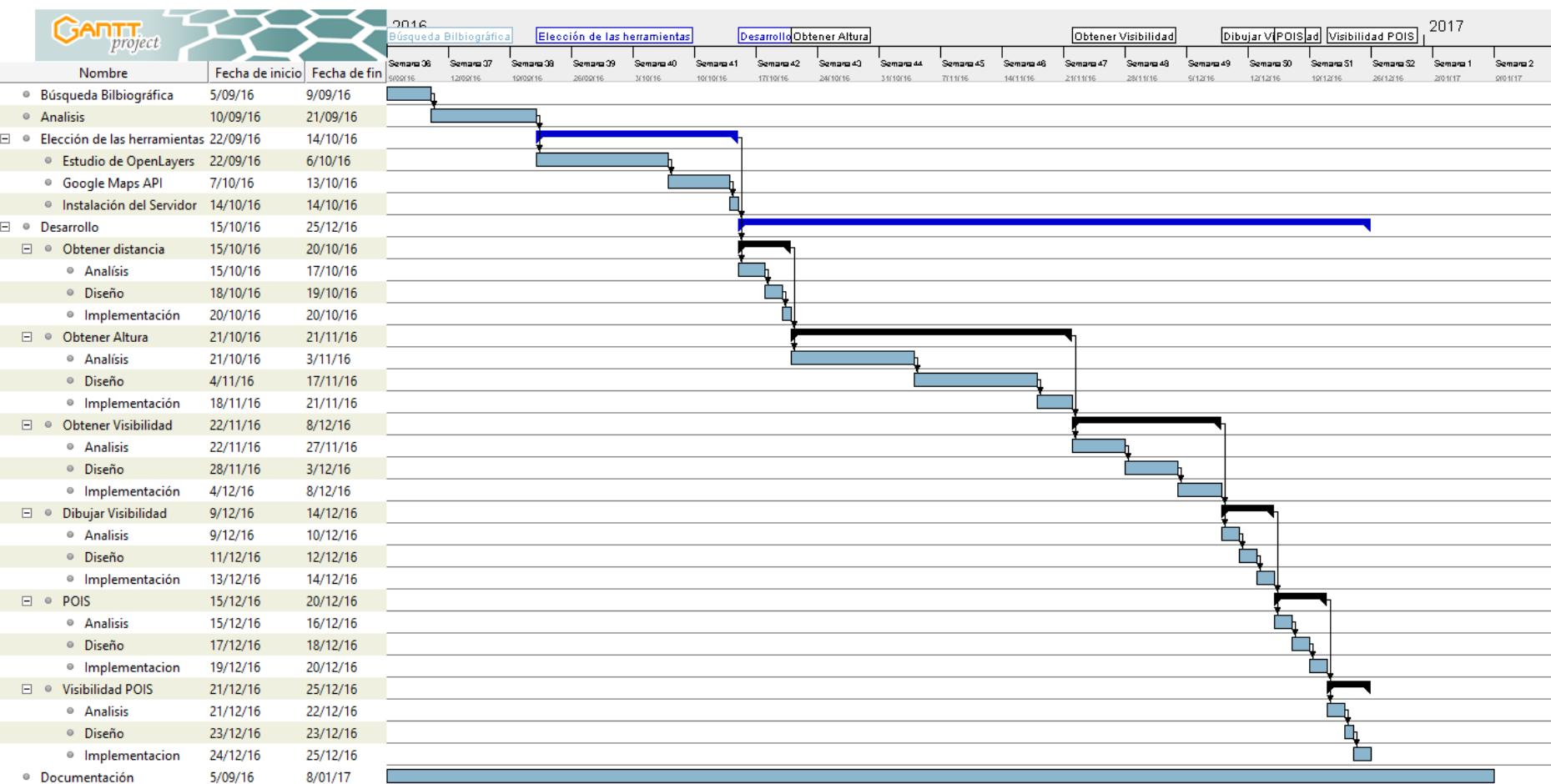


Ilustración 7 -Diagrama de Grant

2.1.2. Diagrama Pert

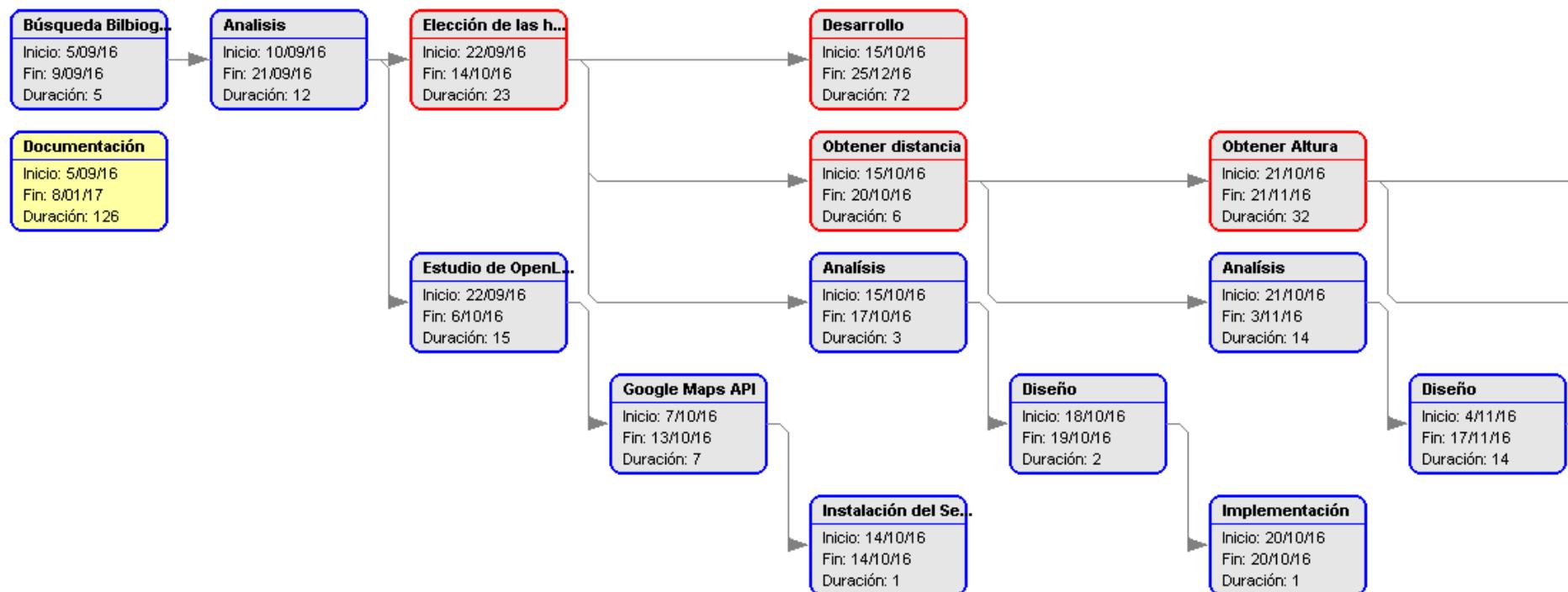
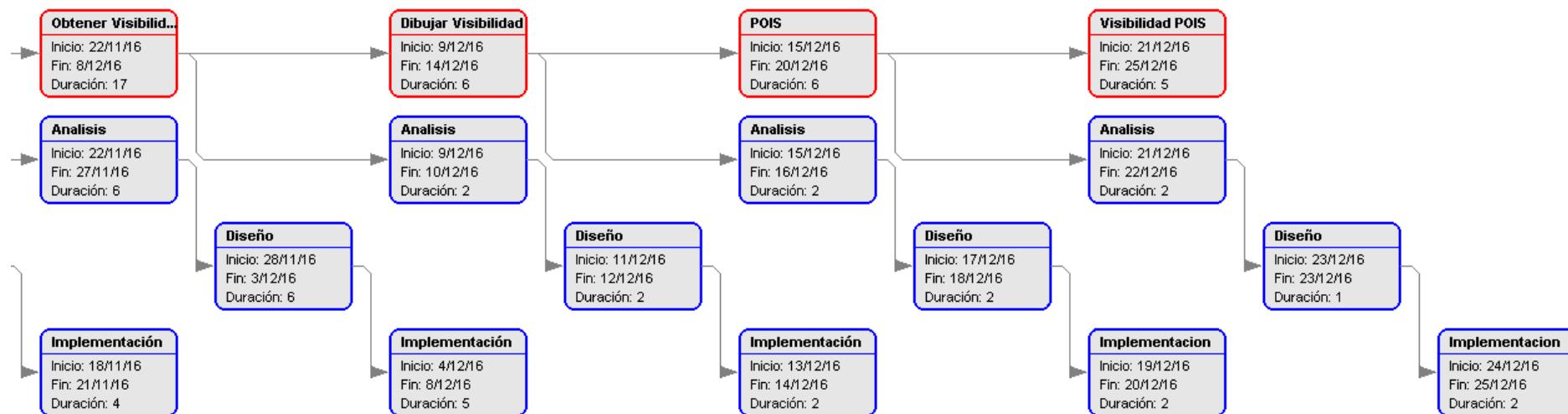


Ilustración 8 - Diagrama Pert 1



Nota: Se ha decidido dividir el diagrama para su correcta visualización.

2.2. Análisis de costes

En este capítulo se analizará una estimación de costes del proyecto.

Para esto, se tendrán en cuenta todos los gastos que este proyecto conlleva, como son gastos de software, hardware, trabajadores, conexión a internet, local, etc.

Respecto al software utilizado, se ha programará en el entorno de Notepad++, bajo el sistema operativo de Windows 10 Pro. Para el editado de imágenes, se usará la herramienta Gimp, para el desarrollo de los diferentes diagramas se utilizará Visual Paradigm, para la edición de la documentación se usará Microsoft Office 2016, además de la API de Google Maps con suscripción Premium. Por otro lado, el servidor en el que alojaremos la aplicación, usará Windows Server 2016. También se necesitará un nombre de dominio web.

Normalmente, no se suele desarrollar solo un proyecto a la vez, por tanto, tendremos en cuenta que estas herramientas están siendo utilizadas por 3 proyectos simultáneos, lo cual divide su precio entre estos. En el caso de la API de Google Maps, sí que es única para cada proyecto, por lo que el precio de esta se aplicará al completo.

Software	Precio Unidad (€)	Unidades	Precio Total (€)
Notepad++	0	1	0
Windows 10 Pro	279,00	1/3	93,00
Windows Server 2016	550,00	1/3	183,33
Gimp	0	1	0
Visual Paradigm	19,00	1/3	6,33
Microsoft Office 2016	149,00	1/3	49,66
Dominio	9	1	9
Premium Key de Google Maps Elevation	10345,5	1	10345,5
Total	-	-	10686,82

Tabla 3 - Software

Respecto al hardware, este proyecto se ha desarrollado en un equipo MSI GE20 2PC, el cual dispone de un procesador Intel Core I7 de 8 núcleos, una tarjeta gráfica dedicada de 2GB GDDR5, un disco duro de 256GB SSD, un disco duro de 1TB HDD, y 8GB de memoria RAM DDR3, con un precio de 980€. Podemos deducir que el equipo citado anteriormente tiene una vida útil de entre 5 y 7 años, lo que supone unas 61320 horas. Por lo tanto el precio del uso por hora de este equipo asciende a 0,015981€/Hora.

Además, se necesita un servidor en el que alojar nuestra aplicación, a la hora de elegir el servidor se va a tener en cuenta que puede albergar varios desarrollos, y que puede ser usado para diferentes actividades, por lo que las características serán superiores de las que, en un principio, este proyecto necesitaría. Por esto, vamos a adquirir un Servidor Dell Smart Value Poweredge T330, el cual dispone de un procesador Intel Xeon e3-1280 3.70 GHz, 16GB de RAM UDIMM, un disco duro de 1TB HDD, puerto SFP doble a 10Gb con un precio de 2354,82€. Este servidor, y debido a sus componentes, podemos predecir que tendrá una vida útil de unos 6 años, con lo que obtenemos unos 0,04480 €/Hora.

Se necesita una conexión a internet para esto hemos contratado una línea de Movistar con 300 Mb simétricos de fibra óptica con un precio mensual de 52,39 euros. Tomando el mes de 30 días, nos da un gasto de 0,0727 €/Hora.

También tenemos en cuenta el gasto eléctrico que tendrá nuestro servidor que estará conectado 24H. Procedemos a calcular el precio de la luz, que actualmente varía cada hora. Por lo que calcularemos el precio medio de un día, que equivale a 0,16859€/kWh.

El gasto del servidor es de 495W. Po lo que obtenemos un gasto de 0,08345 € por hora. Para el equipo de desarrollo, consume 280W, por lo que tiene un gasto de 0,04720 €/Hora.

Como bien hemos dicho antes, estos gastos estarían repartidos entre 3 proyectos, por lo que para este cálculo nos quedaremos con un tercio de los datos obtenidos. En el caso del gasto eléctrico del equipo de desarrollo, se calculará teniendo en cuenta las horas de desarrollo de este proyecto.

Por lo que el gasto hardware quedaría de la siguiente manera:

Hardware	Precio/1000 Horas(€)	Horas Totales	Precio Total (€)
Equipo de desarrollo	5,327	378	2,01
Servidor	14,93	8760	130,81
Conexión a Internet	24,23	8760	212,25
Consumo Servidor	27,816	8760	243,66
Consumo Equipo	15,73	378	5,94
Total	-	-	594,69

Tabla 4 - Gasto Hardware

A la hora de calcular el sueldo del personal del proyecto se va a tener en cuenta “XVI Convenio colectivo estatal de empresas de consultoría y estudios de mercados de la opinión pública” publicado en el BOE el 4 de abril de 2009. El sueldo anual de un titulado es de 20954,36 anuales. Si tomamos los meses de 30 días, de los cuales 22 son laborables y en los que cada día se trabaja 8 horas, podemos deducir que el precio por hora es de 9,92157 €/Hora. Este proyecto se ha desarrollado en un total de 378 horas por lo que tiene un coste de 3750,3542€. A esta cantidad se le añade el gasto de seguridad social, que suele suponer un 30% del sueldo, por lo que el gasto de personal asciende a 4875,46046 € en total.

Actividad	Precio
Software	10686,82
Hardware	594,69
Empleados	4875,46
Total	15156,98

Tabla 5 – Tabla de costes de desarrollo del proyecto.

Hasta ahora, tan solo hemos afrontado costes, pero debemos tener un beneficio, este beneficio representará un 20% de estos costes, además se introducirá un pequeño margen de error por los posibles imprevistos o modificaciones del usuario, para esto se va a aumentar en un 10% los costes. Por último, y no menos importante, debemos añadir el IVA, que en este caso supone un 21% del total.

Por tanto los costes totales serían los siguientes:

Actividad	%	Precio
Gastos		16156,97731
Beneficio	20	3231,3954
Imprevistos	10	1615,6977
-	-	21004,07044
IVA	21	4410,8547
Total		25414,92523

Tabla 6 – Coste Total del proyecto

Por tanto el coste de nuestro proyecto ascendería a un total de 25414,92523 € el primer año, como se ha explicado antes, en este precio está incluida la licencia Premium de Google Maps, así como la compra del dominio. El pago de estos servicios es anual, por lo que este precio sería el precio de inicialización del proyecto.

3. Análisis

En este capítulo se analizan los detalles del problema, los requisitos y la solución escogida. Esta es una de las fases más importantes a la hora del desarrollo de una aplicación, ya que aquí se crean las bases del proyecto.

3.1. Requisitos

Este subcapítulo se basará en los requisitos que tendrá nuestro sistema, ya sean requisitos funcionales o requisitos no funcionales. Los requisitos son las propiedades o restricciones definidas con precisión que un sistema debe satisfacer.

3.1.1. Requisitos Funcionales

Los requisitos funcionales son los diferentes servicios que el sistema debe proporcionar al usuario.

Los requisitos funcionales de esta aplicación son:

- La utilización de un mapa para la vista de la aplicación
- La posibilidad de aplicar capas de visibilidad a una zona determinada de un mapa
- Crear y eliminar puntos de referencia o POIS que quedarán almacenados
- Comprobar la visibilidad entre los distintos POIS
- Visualización de la capa de visibilidad de los POIS

3.1.2. Requisitos No Funcionales

Los requisitos no funcionales describen aspectos del sistema que están relacionados con el grado de cumplimiento de los requisitos funcionales.

Los requisitos funcionales de esta aplicación son:

- Se evitará tener que recargar la página para poder visualizar nuevos datos.
- La aplicación tendrá una interfaz intuitiva, no dando lugar a ambigüedades.

3.1.3. Requisitos del Cliente

Los requisitos hardware que se necesitan para esta aplicación son muy básicos, basta con un dispositivo con una conexión a internet.

Los requisitos software que necesitaran los usuarios de esta aplicación serán básicamente un navegador compatible con JavaScript y funciones asíncronas como son: Chrome, Firefox, Opera u Microsoft Edge entre otros.

3.1.4. Requisitos del Servidor

Los requisitos hardware de nuestro servidor no tienen por qué ser muy buenos, ya que como la aplicación se desarrolla en JavaScript el servidor tan solo se encargará de manejar las peticiones a la base de datos. En este caso se ha instalado la aplicación WAMP Server 2.2 que incluye un servidor Apache, MySQL y PHP. Los requisitos para esta aplicación son:

- Instalable en discos con formato NTFS
- Instalable en Windows Vista SP1, W7, W8, W8.1, W10 y Windows Server a partir de estas versiones.

Respecto a los requisitos de las aplicaciones que incluye, los requisitos mínimos de PHP son:

- Windows XP SP3

En cuanto a MySQL los requisitos mínimos son:

Hardware	Requisitos Mínimos	Requisitos Recomendados
Memoria	2GB	8GB o más
Procesador	2 Cores	4 Cores o más
Disco Duro	Intensidad de Escritura	RAID 10 o RAID 0+1

Tabla 7 - Requisitos MySQL

Por lo tanto los requisitos del servidor son:

Hardware	Especificaciones
Memoria	Se necesita una memoria para que nuestra base de datos consiga funcionar fluida, por tanto con 8GB sería más que suficiente.
Procesador	Para el procesador nos decantaremos por un Intel Core I3, ya que apenas deberá correr procesos.
Disco Duro	El disco duro será suficiente 256GB, con este espacio se prevé la falta de espacio que puede causar un aumento de la base de datos.
Tarjeta de Red	Se necesitará una tarjeta de red Ethernet gigabit de gran velocidad.

Tabla 8 - Requisitos Hardware

La conexión a internet debe ser permanentemente 24 horas durante 365 días al año, se intentará conseguir el máximo tiempo de actividad del servidor.

En cuanto a los requisitos software del servidor necesitaremos lo siguiente:

Software	Especificaciones
S.O.	Cualquier sistema operativo que pueda ejecutar un servidor web, en nuestro caso nos hemos decantado por Windows 10.
Base de datos	Se usará como gestor de base de datos MySQL.
Lenguajes	Se usará JavaScript para la API de Google Maps y PHP para la gestión de la base de datos.
Librerías	Se utilizarán como librerías auxiliares PNGLib para el manejo de las imágenes PNG, JQuery para las consultas a la Base de datos mediante PHP, además de la API de Google Maps.

Tabla 9 - Requisitos Software

3.2. Mapas

Para el desarrollo de este trabajo, un requisito fundamental es la obtención de un mapa que nos permita situarnos en una localización y desde esta representar los datos generados. A continuación veremos las diferentes alternativas a representaciones de mapas investigadas para este TFG:

- **OpenLayers:**



Ilustración 10 - Openlayers

Es una biblioteca de JavaScript de código abierto, que permite la utilización de diferentes capas de un mapa.

- **Google Maps API:**



Ilustración 11 - Google Maps API

Conjunto de reglas y especificaciones necesarias para utilizar los mapas de Google. Esta API te permite utilidades para manipular los mapas, añadir contenido o crear aplicaciones para mapas.

- **OpenStreetMap:**



Ilustración 12 - OpenstreetMap

Es un proyecto abierto y colaborativo para crear mapas modificables y libres. Los mapas se crean mediante la ayuda de los usuarios, usando la información capturada con los dispositivos GPS móviles.

- **Bing Maps:**



Ilustración 13 - Bing Maps

Desarrollada por Microsoft, es la base de mapas que soporta Windows. Dispone de vista satélite, mapas 3D, vista de pájaro o Streetside que dispone de una vista similar a Google Street View.

- **MapQuest:**



Ilustración 14 - MapQuest

Basada en una red de mapas propia desarrollada con la colaboración de Navteq. Dispone de capas para encontrar tiendas, hoteles, restaurantes, etc. Actualmente no dispone del servicio de satélite.

- **Apple Maps:**

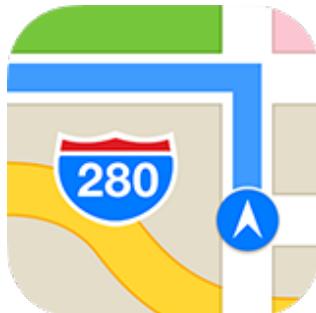


Ilustración 15 - Apple Maps

Desarrollado por Apple, dispone de mapas en 3D, vista satélite, o vista de pasos elevados.

Analizando a fondo las diferentes opciones, nos decantamos entre dos opciones OpenLayers y Google Maps API.

Encontramos que en OpenLayers no es posible la carga de una capa que tenga los valores de altura. Se puede cargar la capa de Google Maps de la cual lo único que podemos usar es la visualización del mapa. Sin embargo, si usamos el servicio de Google Maps disponemos de una capa llamada Google Maps Elevation, con la que podemos obtener los diferentes datos de altura necesarios para este proyecto. Por lo tanto se utilizará la API de **Google Maps Elevation**.

3.3. Lenguaje de Programación

Una vez elegido el tipo de mapa a utilizar, en este caso la API de Google Maps, se elegirá el lenguaje de programación JavaScript, ya que la API de Google Maps está desarrollada en este lenguaje. Además, se utilizarán algunos script PHP para la manipulación de la base de datos.

3.4. Servidores, Servicios y Otros

Para el desarrollo de este trabajo necesitaremos un servidor web que albergue la aplicación, además de una base de datos en la que guardaremos los puntos de interés de nuestro mapa, para estos servicios nos hemos decantado por la aplicación WAMP que incluye servidor web Apache, PHP y MySQL, además de una interfaz gráfica para la manipulación de la base de datos MySQL

3.5. Librerías

Para este proyecto hemos utilizado dos librerías de JavaScript:

- PNGLib: Es una librería para la creación de imágenes PNG en JavaScript, mediante esta librería crearemos la capa de visibilidad que posteriormente se le aplicará al mapa de Google Maps, facilitándonos el uso de imágenes en este proyecto.
- JQuery: Es una librería esencial para el desarrollo web, nos facilita el desarrollo de aplicaciones del lado cliente, siendo compatible con todos los navegadores. En este proyecto es usada creando peticiones Ajax a pequeños script PHP que manipulan la base de datos.

4. Diseño

En este capítulo se analizará más afondo cada una de las características que tendrá nuestro sistema, lo que nos permitirá implementarlo de forma efectiva. Se estudiarán los diferentes diagramas, un estudio del patrón modelo-vista-controlador y se mostrará una primera vista de lo que será este sistema.

4.1. Diagrama de clases

En este diagrama se representan las diferentes clases que forman este proyecto. En este caso al ser una aplicación web no son clases en sí, si no que se han diferenciado según el tipo de función que realizan, además están separadas en diferentes archivos en la práctica.

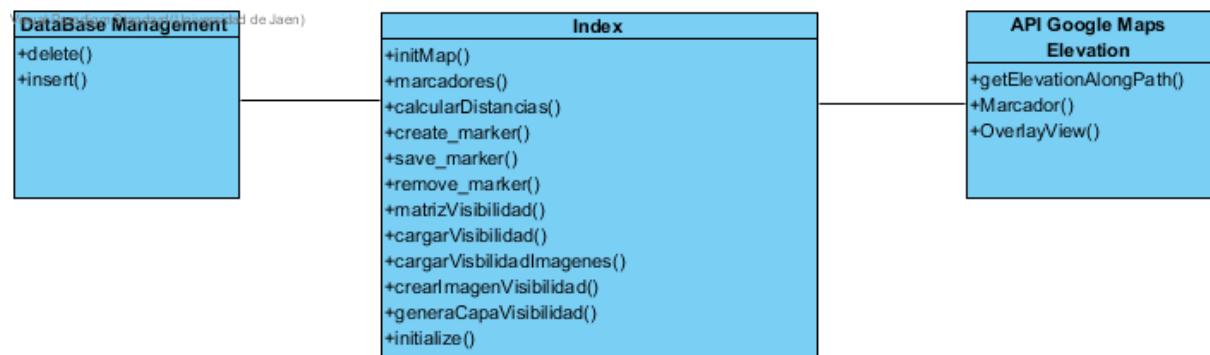


Ilustración 16 - Diagrama de Clases

Por tanto disponemos de una clase index, que sería la página web, y es la encargada de la interfaz y de los diferentes eventos. Esta clase se comunica con la API de Google Maps para recibir las alturas de los diferentes puntos, además de hacer uso de la superposición de imágenes. La clase index, por otro lado, se comunica con la clase, en este caso llamada, Database Management, que es la encargada de manipular los datos de la base de datos.

La clase Index dispone de los siguientes métodos:

- initMap: Este método se encarga de crear los eventos del mapa, además se crean los marcadores.

- Marcadores: Este método es el encargado de consultar la Base de Datos y añadir los marcadores al mapa.
- calcularDistancias: Calcula la longitud en metros entre dos localizaciones
- create_marker: Crea los POI
- save_marker: Guarda los POI en la base de datos.
- remove_marker: Eliminar un POI de la base de datos si existe, o del mapa si no se ha guardado.
- matrizVisibilidad: Se encarga de rellenar las casillas de la matriz
- cargarVisibilidad: Calcula la visibilidad de los puntos desde un punto referencia.
- cargarVisibilidadImagenes: Calcula la visibilidad de los POI, y modifica el icono del mismo según esta.
- creaImagenVisibilidad: Crea una imagen PNG que será aplicada al mapa.
- generaCapaVisibilidad: Prepara los datos necesarios para los cálculos de visibilidad, así como las alturas o distancias entre los puntos.

La clase Database Management, dispone de los métodos Insert(), para insertar un nuevo POI en la base de datos y el método delete() que elimina un POI de la base de datos.

Para la API de Google Maps en este caso solo se han usado dos funciones más representativas, como son: getElevationAlongPath, que nos devuelve diferentes alturas a lo largo de dos puntos y OverlayView que nos permite sobreponer una imagen en el mapa, facilitando así, la superposición de la capa de visibilidad calculada en otras operaciones.

4.2. Diagrama de Casos de Uso

Ahora vamos a mostrar los diagramas de casos de uso, en este caso tan solo se dispone de un diagrama de caso de uso, ya habrá un único usuario que utilizará la interfaz.

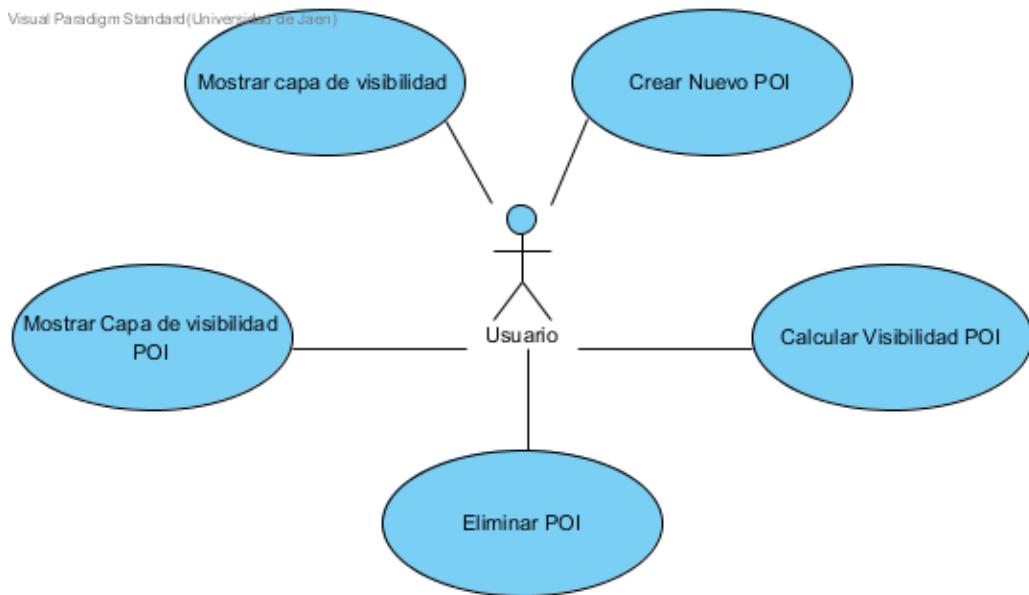


Ilustración 17 - Diagramas de casos de uso

El usuario puede:

- Mostrar la capa de visibilidad
- Crear un nuevo POI
- Eliminar un POI
- Mostrar la capa de visibilidad de un POI
- Calcular la visibilidad de un POI

4.3. Diagrama de Actividad

Ahora vamos a mostrar el diagrama de actividad, mostrando los diferentes estados que puede tener nuestra aplicación y explicando cada uno de ellos.

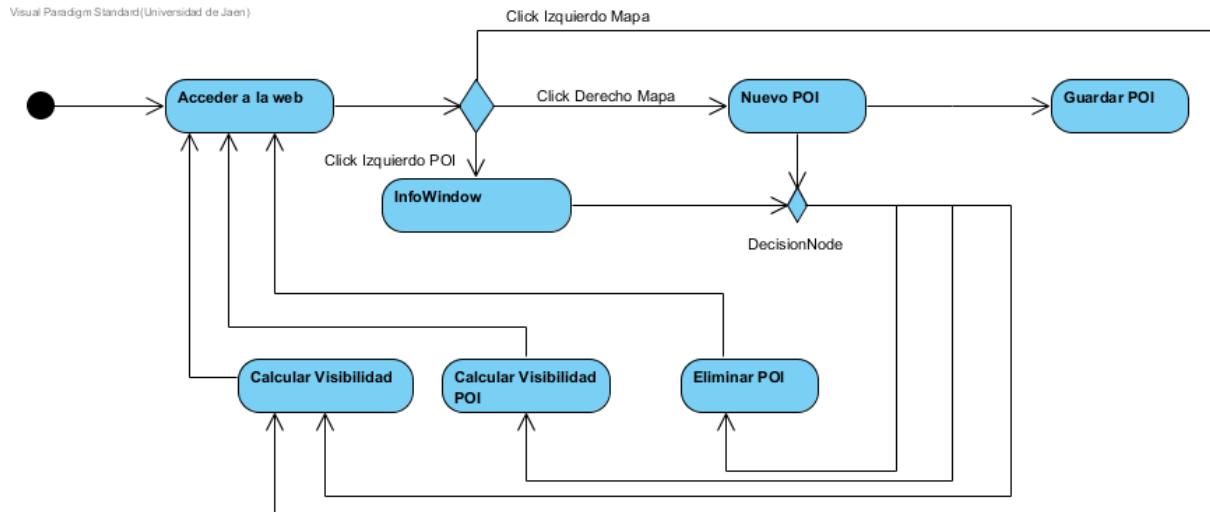


Ilustración 18 - Diagrama de Actividad

Como podemos ver al acceder a la web, tenemos tres posibles opciones, que son pulsar botón izquierdo sobre el mapa, botón derecho sobre el mapa o botón izquierdo sobre un POI.

Si se selecciona el botón izquierdo sobre el mapa calculará la capa de visibilidad. Si, por otro lado, se pulsa el botón derecho sobre el mapa, pasaremos al estado de crear un nuevo POI, donde podremos guardarlo, eliminarlo, calcular la visibilidad, o calcular la visibilidad de los POIs.

Por último, si hacemos clic izquierdo sobre un POI ya creado anteriormente, se podrá calcular la visibilidad, calcular la visibilidad de los POI o Eliminar el POI seleccionado.

4.4. Patrón Modelo-Vista-Controlador

El patrón modelo-vista-controlador (MVC) separa la representación de la información de la interacción con la misma. Es el patrón más extendido en el desarrollo web. Consta de tres capas, como su nombre indica, la capa del modelo, la capa de la vista y la capa del controlador.

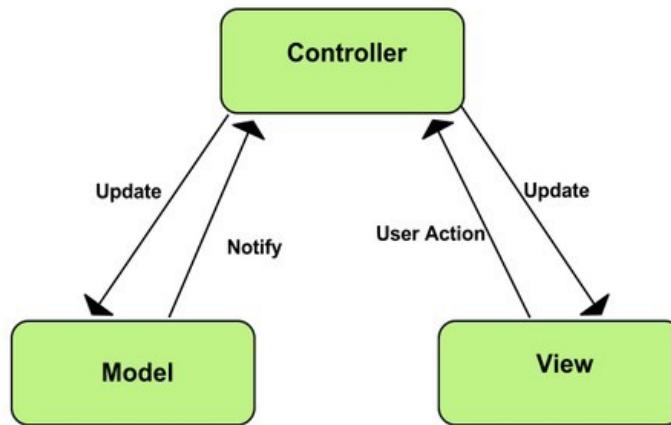


Ilustración 19 - Modelo Vista Controlador

Las ventajas de este tipo de patrones es su flexibilidad, ya que podemos modificar cada capa por separado. Este modelo se encarga de independizar la interfaz del usuario, la lógica y los datos.

- **El modelo:** Mantiene la gestión y estructura de los datos, permite su procesamiento (creación, eliminación, modificado, etc.), está basado normalmente en un Sistema Gestor de Base de Datos.
- **La vista:** Se encarga de la recolección de los datos que le envía el modelo, estos datos son adaptados para la correcta representación en la interfaz. Esta capa es la encargada de la interfaz del usuario.
- **El controlador:** Gestiona las peticiones del cliente en el lado servidor, este es el encargado de avisar al modelo de las acciones realizadas en la vista, de la misma forma el controlador avisa a la vista de nuevas modificaciones del modelo.

De este modo, nuestro proyecto se basa en este patrón. En este caso la capa de la vista estaría definida por el mapa de Google Maps. El mapa de Google Maps define nuestra interfaz y en ella están implementadas todas las interacciones del usuario, así, el usuario es capaz, mediante eventos del ratón, de informar a la vista de sus intenciones.

La vista, una vez obtiene las interacciones del usuario, informa al controlador, que en nuestro caso sería la aplicación web que hemos desarrollado. Estos eventos son manejados por el controlador, que según su función, se tratarán para ser enviados el modelo, o directamente se evaluarán en el propio controlador. En nuestro caso, la mayoría de los datos enviados al modelo son localizaciones formadas por longitud y latitud. Por ejemplo, cuando consultamos a la API de Google Maps por las alturas entre dos puntos, el controlador envía las localizaciones de ambos puntos, y el modelo nos responde con las alturas deseadas, o en otro caso, cuando eliminamos un marcador, avisamos al modelo de la localización de este marcador, y el modelo se encarga de eliminarlo, lo mismo pasa con la creación de nuevos marcadores, en los que enviamos la localización y una pequeña descripción.

Según el modo que se ha realizado esta aplicación, disponemos de dos capas de modelo. Una capa de modelo de nuestra aplicación sería la propia API de Google Maps, la cual utilizamos para consultar los datos de las alturas de nuestros puntos, haciendo así las veces de Base de datos.

Por otro lado, la otra capa de modelo de la cual se compone este proyecto, sería la parte web desarrollada en PHP, que se encarga del manejo de la base de datos. Este modelo se encarga de añadir nuevos POIs a la base de datos, o de eliminarlos en el caso de que el usuario acceda al evento

4.5. Diagramas de secuencia

En este capítulo se aborda la creación de los diagramas de secuencia. Estos diagramas muestran la interacción entre los objetos organizados en una secuencia de tiempo. Como hemos explicado anteriormente, se utiliza el patrón modelo-vista-controlador, por lo tanto se mostrará el intercambio de información entre las capas.

4.5.1. Carga del mapa

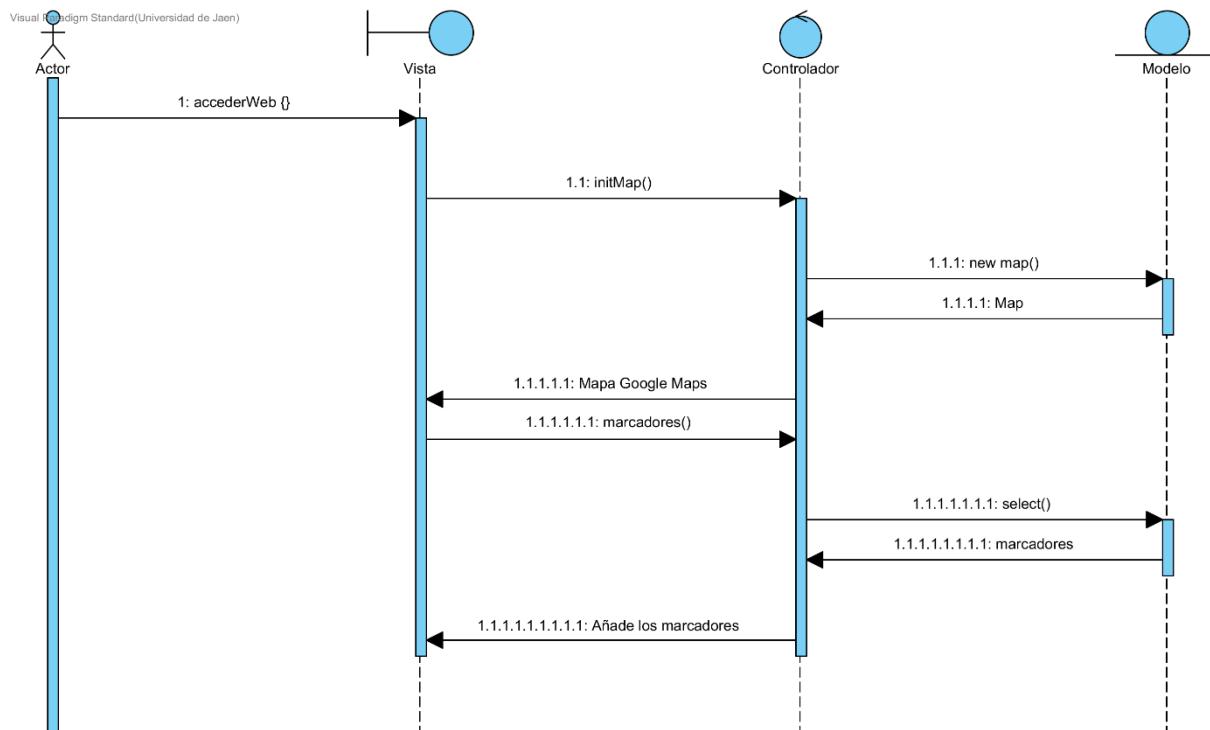


Ilustración 20 - Diagrama de Secuencia Carga del mapa

Al acceder a nuestra web, se lanza una llamada a la función `initMap()`, esta función se encarga de la creación de los diferentes eventos del mapa y se crea el mapa. Además, se consultan los marcadores guardados en la base de datos para añadirlos a la vista.

4.5.2. Generar capa de visibilidad en un punto

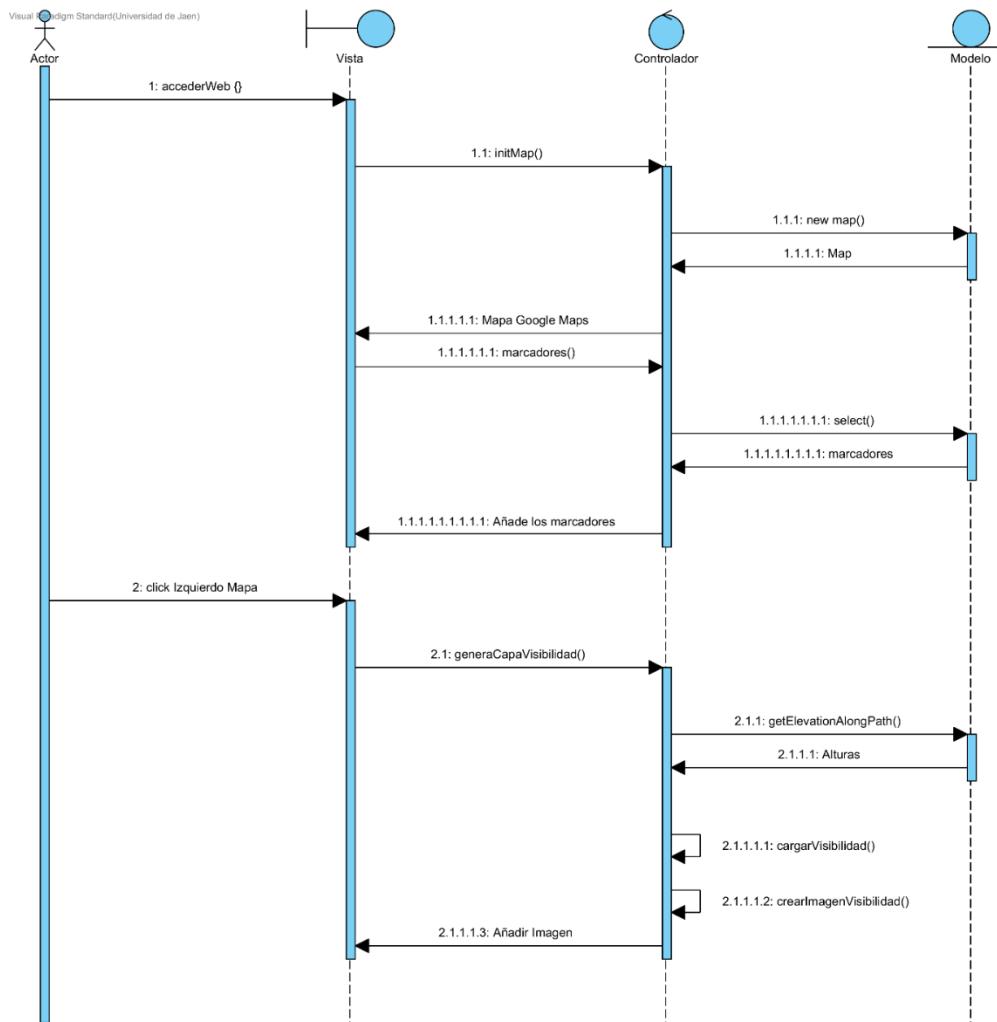


Ilustración 21 - Diagrama de Secuencia Generar capa de visibilidad en un punto

En este diagrama se representa el cálculo de la capa de visibilidad en un punto del mapa. El mapa es cargado al acceder a la web, y al pulsar el usuario con el botón izquierdo en una posición del mapa se consultan las alturas, se calcula la visibilidad, se crea la imagen y por último se añade al mapa.

4.5.3. Crear Nuevo Marcador

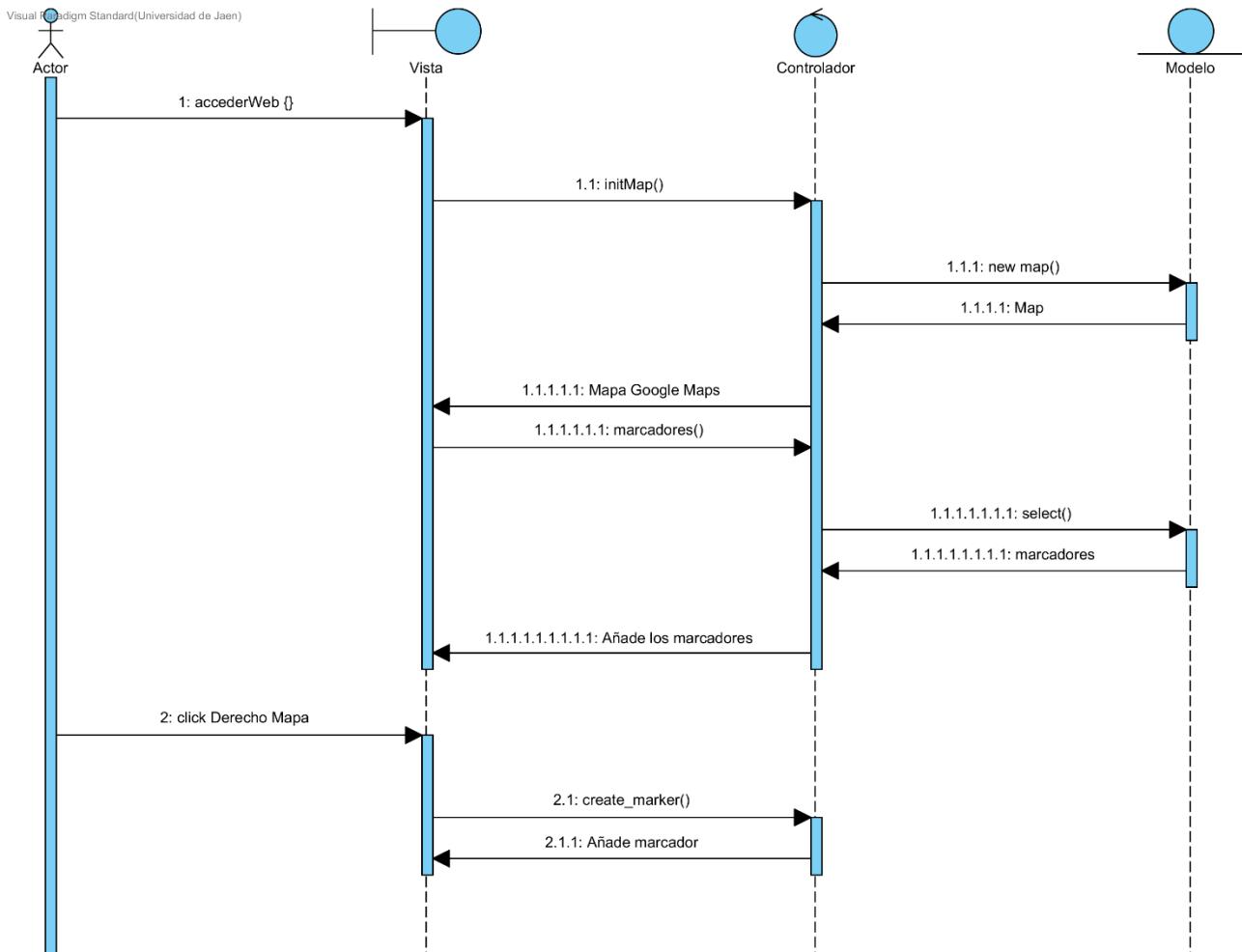


Ilustración 22 - Diagrama de Secuencia Crear nuevo marcador

Este diagrama muestra la creación de un nuevo marcador, para esto se carga el mapa al acceder a la web, el usuario pulsa el botón derecho en la posición donde desea crear el nuevo marcador, el marcador es creado con su ventana de información que es añadida al mapa. En este caso el marcador no se guarda en la base de datos, ya que para esto se dispone de un botón en la ventana de información del marcador, que será explicado más adelante.

4.5.4. Guardar marcador

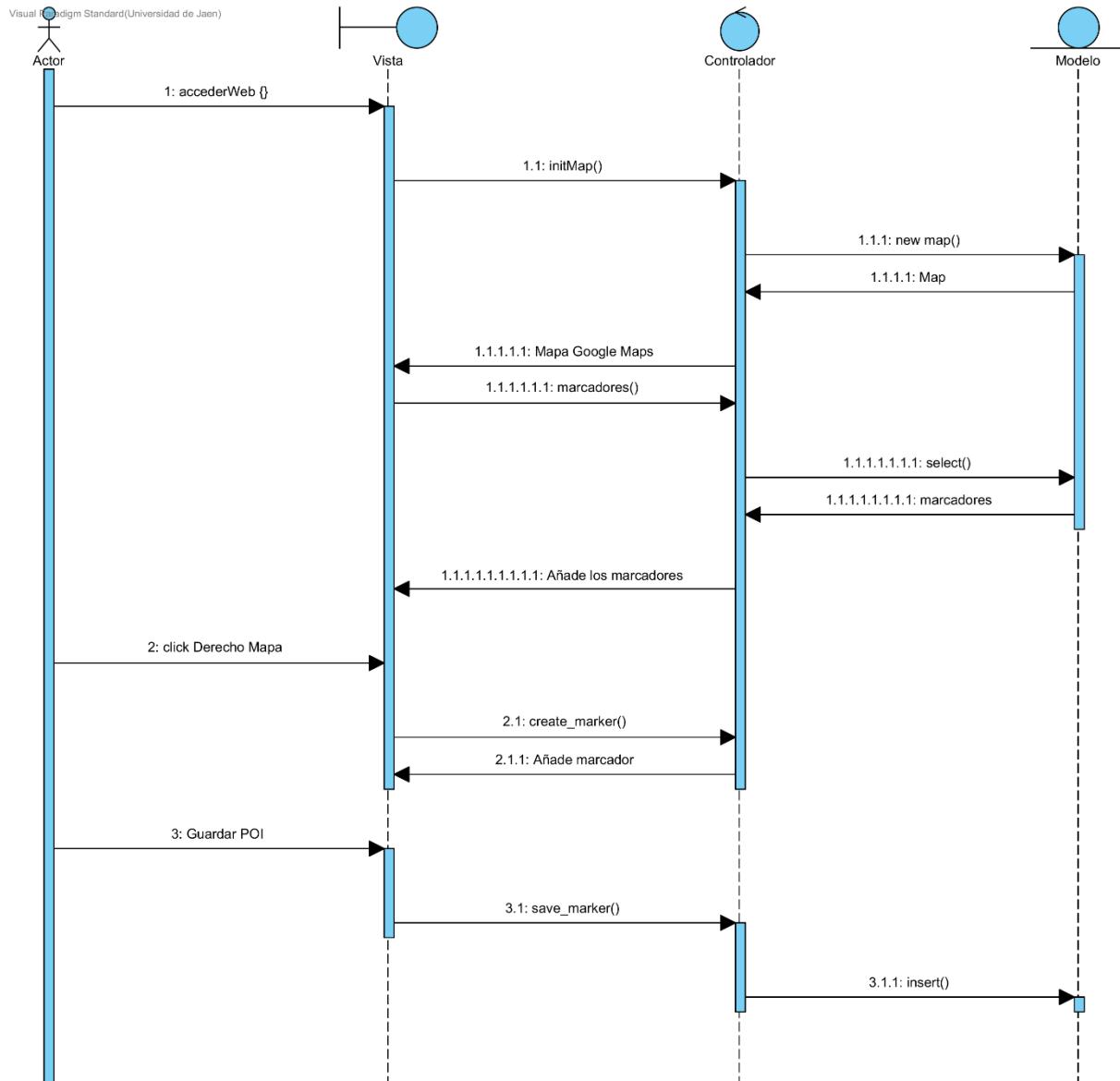


Ilustración 23 - Diagrama de Secuencia Guardar marcador

Este diagrama muestra el guardado de un nuevo marcador, para esto se carga el mapa al acceder a la web, el usuario pulsa el botón derecho en la posición donde desea crear el nuevo marcador, el marcador es creado con su ventana de información que es añadida al mapa, en esta ventana de información aparece el botón Guardar POI, que es seleccionado por el usuario y es añadido a la base de datos.

4.5.5. Calcular la capa de visibilidad desde un marcador no almacenado

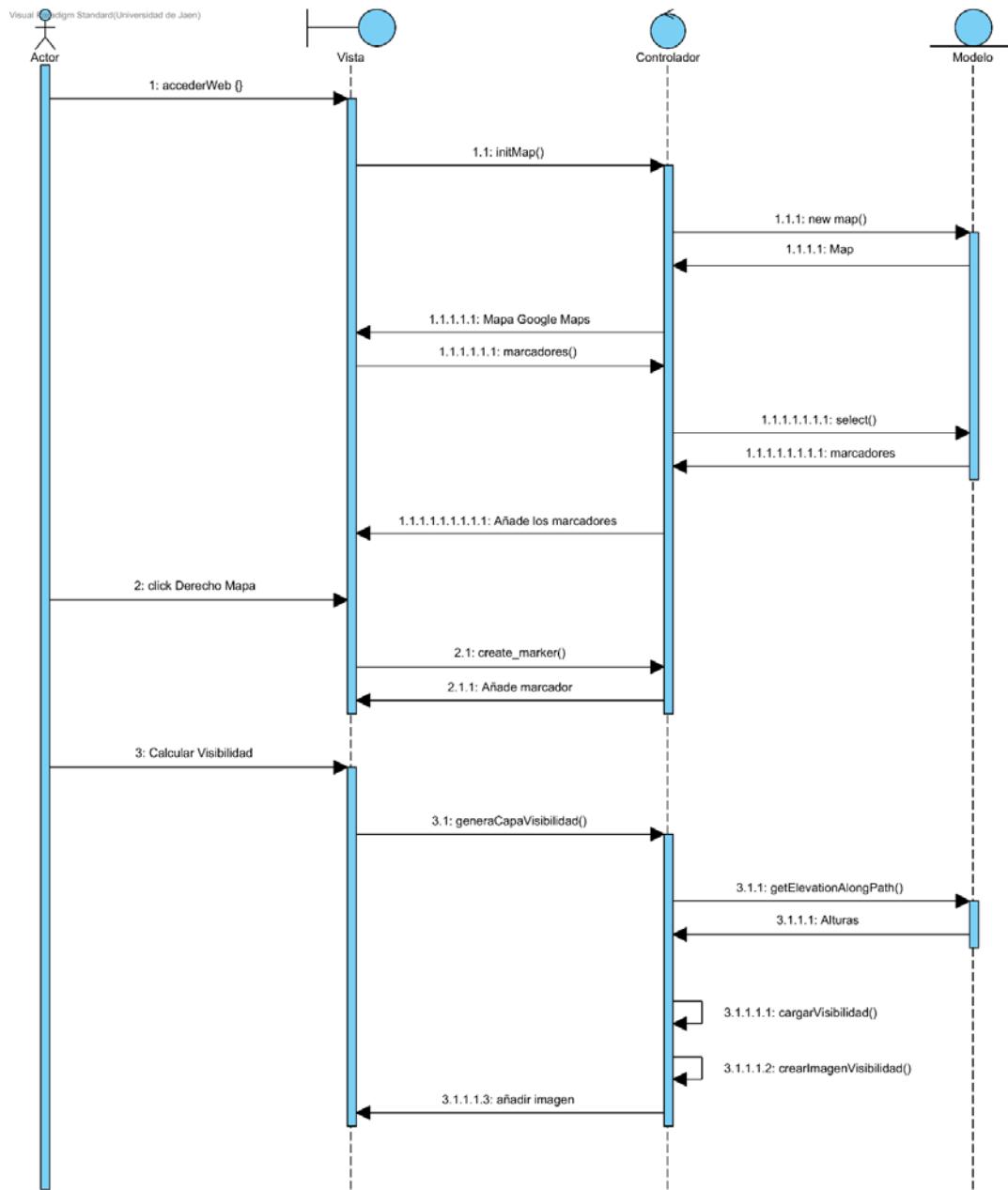


Ilustración 24 - Diagrama de Secuencia Calcular la capa de visibilidad desde un marcador no almacenado

Este diagrama muestra el cálculo de la capa de visibilidad de un marcador no guardado, para esto se carga el mapa al acceder a la web, el usuario pulsa el botón derecho en la posición donde desea crear el nuevo marcador, el marcador es creado con su ventana de información que es añadida al mapa, en esta ventana de información aparece el botón Calcular visibilidad, que es seleccionado por el usuario. En este caso, se consultan las alturas, se calcula la visibilidad, se crea la imagen y por último se añade al mapa.

4.5.6. Calcular visibilidad de los marcadores desde uno no almacenado

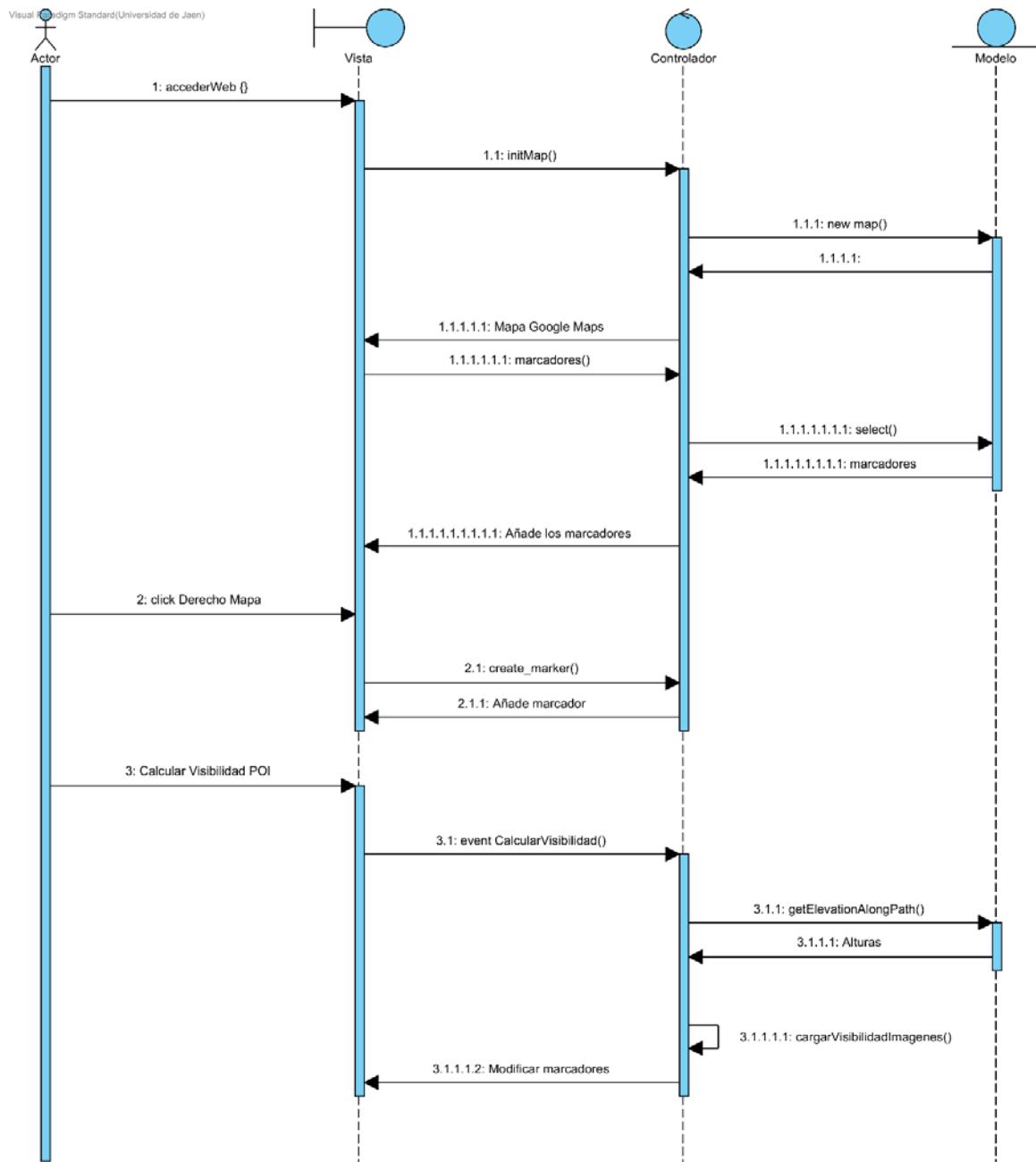


Ilustración 25 - Diagrama de Secuencia Calcular la visibilidad de los marcadores desde uno no almacenado

Este diagrama muestra el cálculo de la visibilidad de un marcado no guardado, para esto se carga el mapa al acceder a la web, el usuario pulsa el botón derecho en la posición donde desea crear el nuevo marcador, el marcador es creado con su ventana de información que es añadida al mapa, en esta ventana de información aparece el botón Calcular visibilidad POI, que es seleccionado por el usuario. En este caso, se consultan las alturas de los diferentes POIs, se carga la visibilidad y se modifica el icono de cada uno de los POIs.

4.5.7. Eliminar marcador no almacenado

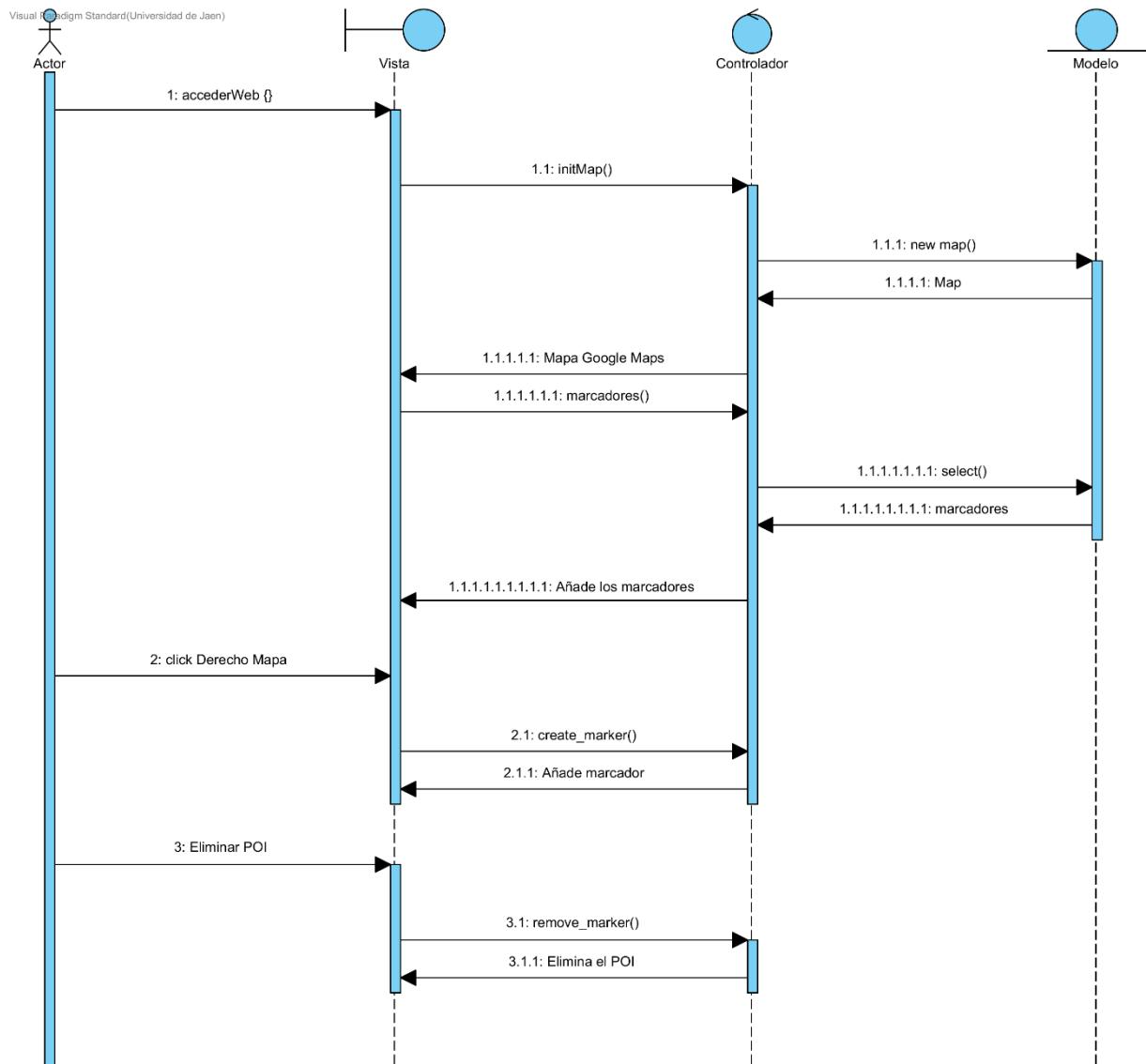


Ilustración 26 - Diagrama de Secuencia Eliminar marcador no almacenado

Este diagrama muestra la eliminación de un nuevo marcador que aún no se ha guardado en la base de datos, para esto se carga el mapa al acceder a la web, el usuario pulsa el botón derecho en la posición donde desea crear el nuevo marcador, el marcador es creado con su ventana de información que es añadida al mapa, en esta ventana de información aparece el botón Eliminar POI, que es seleccionado por el usuario. Ahora lo que se hace es eliminar ese marcador solo del mapa, ya que todavía no existe en la base de datos.

4.5.8. Abrir infoWindow marcador

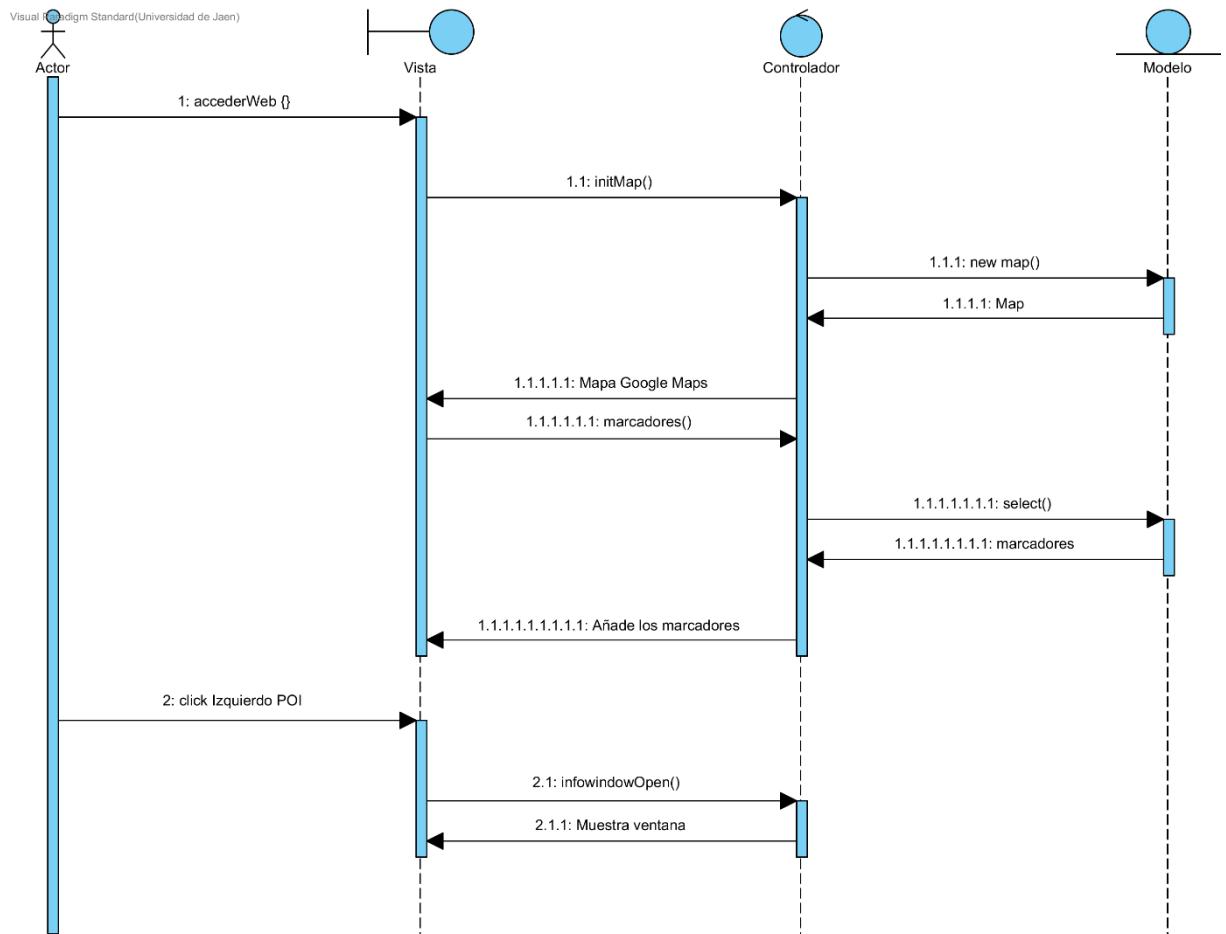


Ilustración 27 - Diagrama de Secuencia Abrir infowindow marcador

Este diagrama muestra la vista de la venta de información de un marcador, para esto se carga el mapa al acceder a la web, el usuario pulsa el botón izquierdo sobre el ícono de un marcador, esto hace que la ventana sea mostrada con la información de este marcador.

4.5.9. Calcular la visibilidad de un marcador almacenado

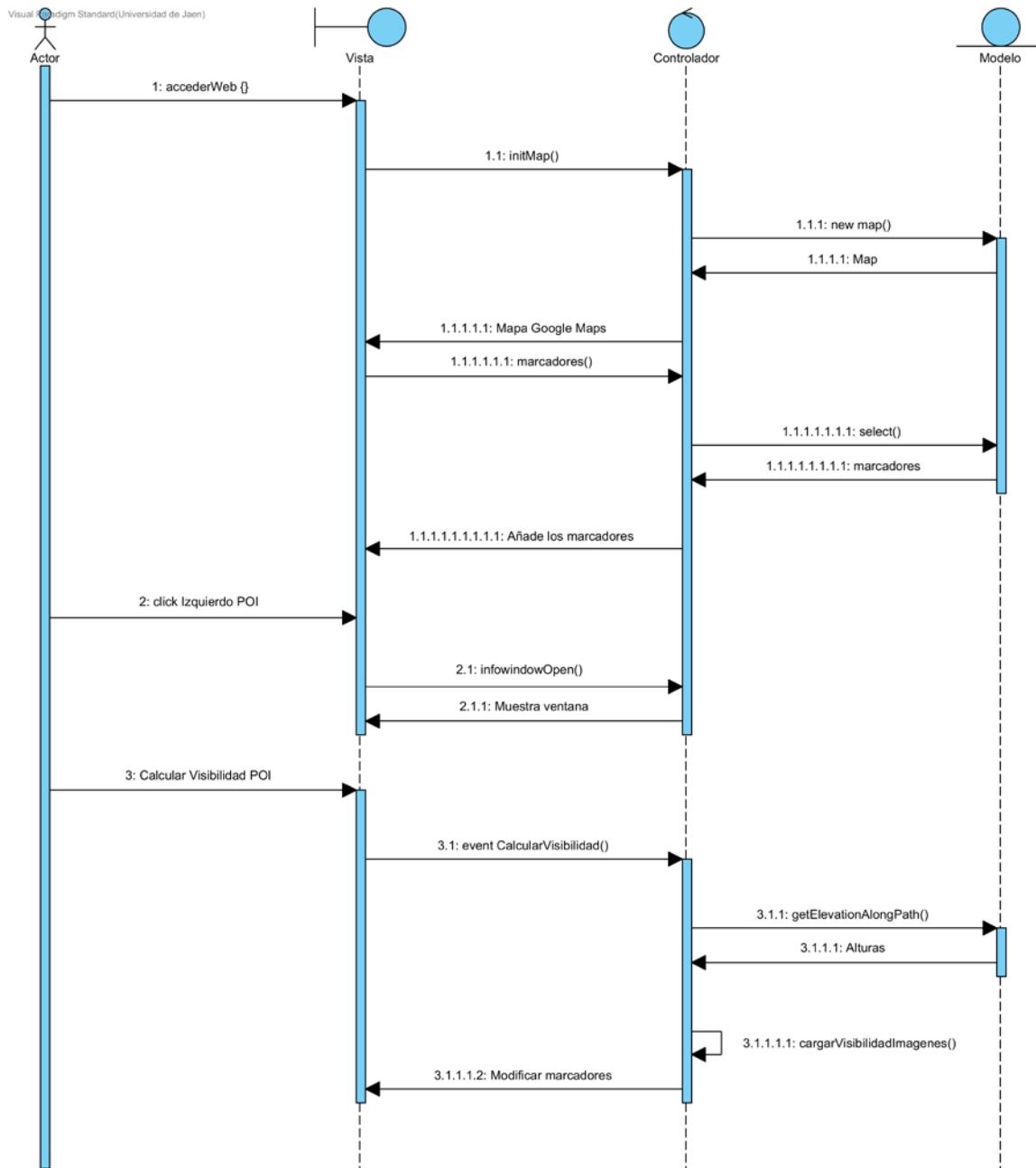


Ilustración 28 - Diagrama de Secuencia Calcular la visibilidad de un marcador almacenado

Este diagrama muestra el cálculo de la visibilidad de un marcador, para esto se carga el mapa al acceder a la web, el usuario pulsa el botón izquierdo sobre el icono de un marcador, esto hace que la ventana sea mostrada con la información de este marcador. En esta ventana de información aparece el botón Calcular visibilidad POI, que es seleccionado por el usuario. En este caso, se consultan las alturas de los diferentes POIs, se carga la visibilidad y se modifica el icono de cada uno de los POIs.

4.5.10. Calcular la capa de visibilidad desde un marcador almacenado

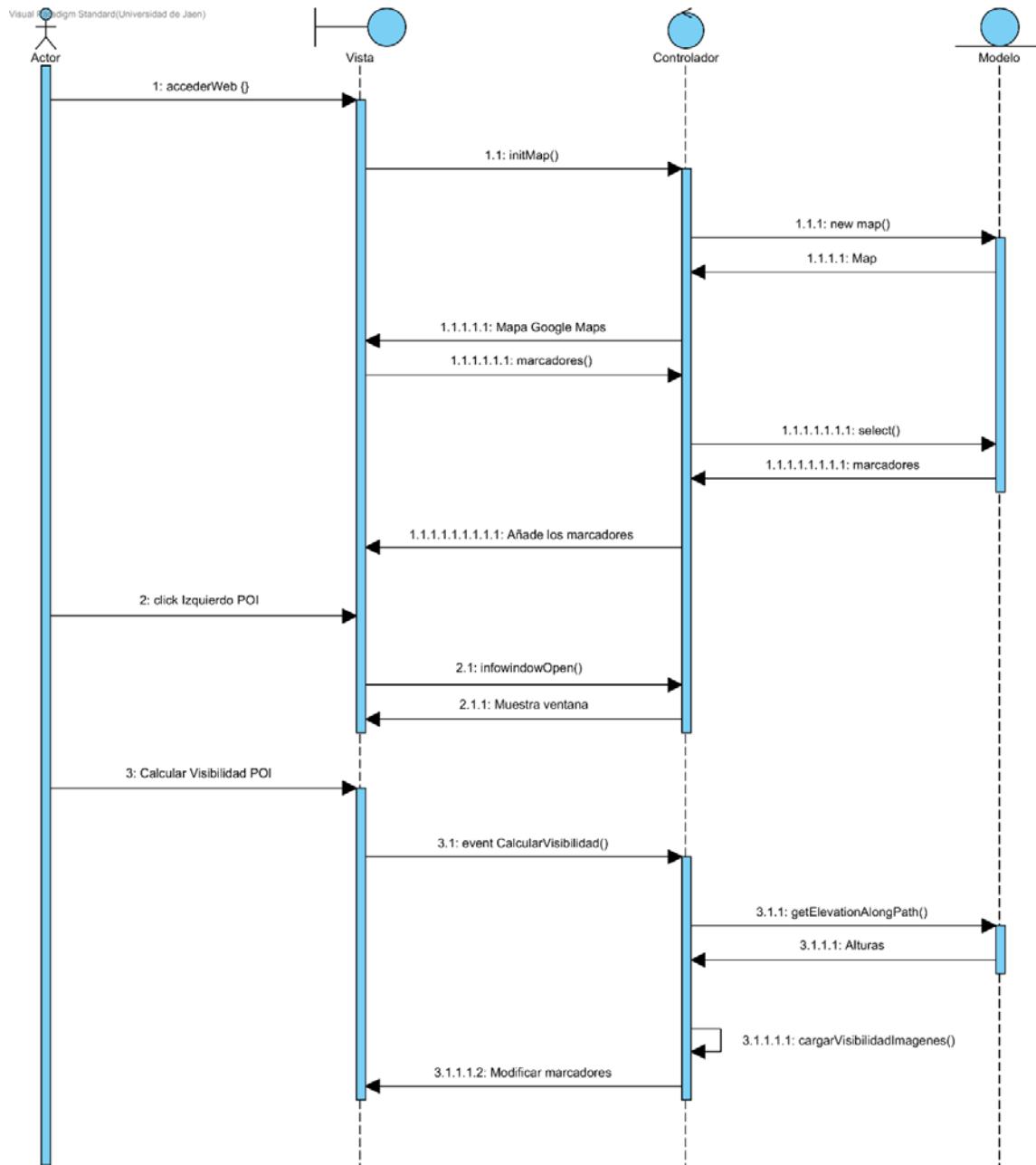


Ilustración 29 - Diagrama de Secuencia Calcular la capa de visibilidad desde un marcador almacenado

Este diagrama muestra el cálculo de la capa de visibilidad de un marcador almacenado, para esto se carga el mapa al acceder a la web, el usuario pulsa el botón izquierdo sobre el icono de un marcador, esto hace que la ventana sea mostrada con la información de este marcador. En esta ventana de información aparece el botón Calcular visibilidad, que es seleccionado por el usuario. En este caso, se consultan las alturas, se calcula la visibilidad, se crea la imagen y por último se añade al mapa.

4.5.11. Eliminar un marcador

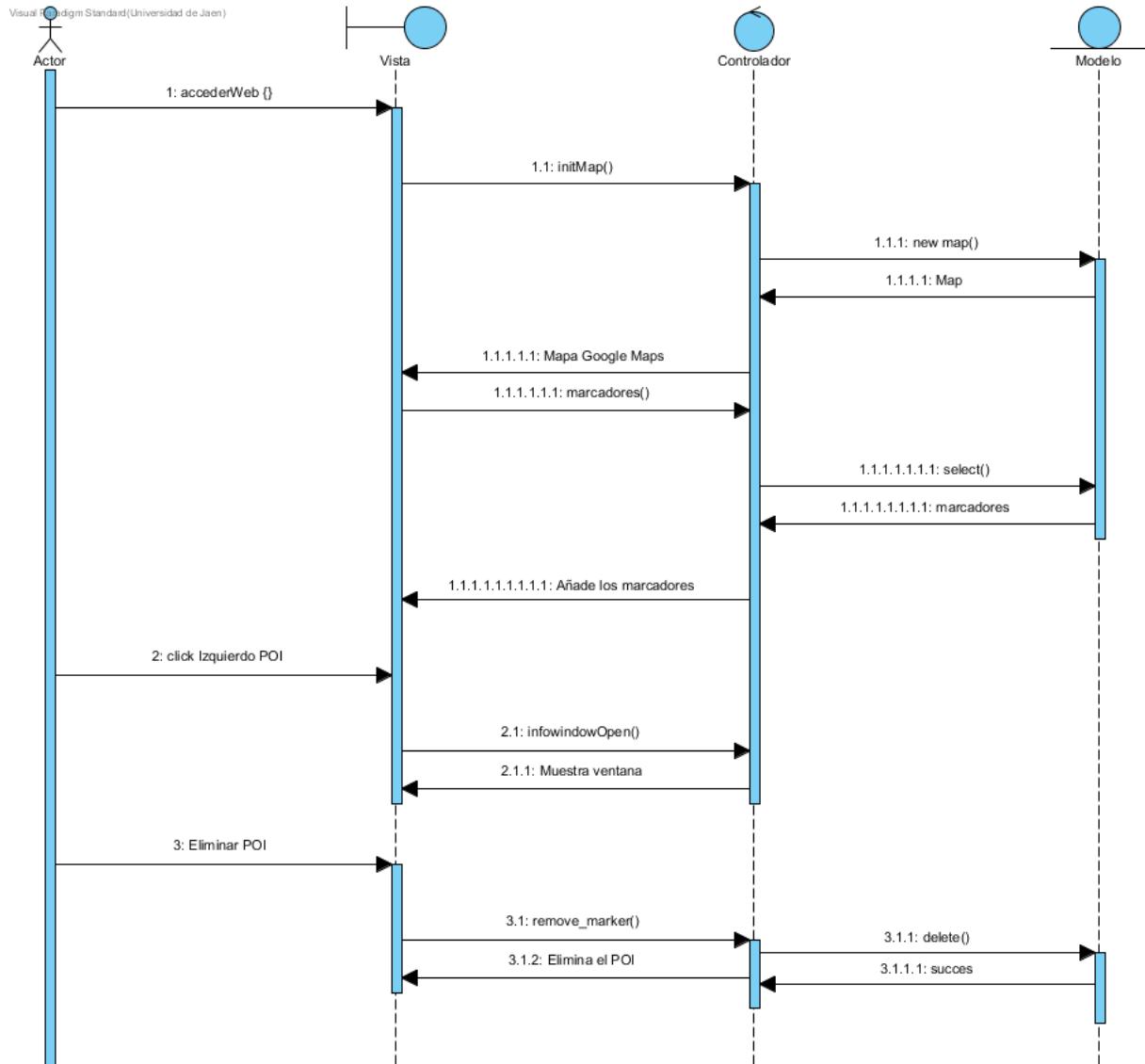


Ilustración 30 – Diagrama de Secuencia Eliminar marcador

Este diagrama muestra el cálculo de la capa de visibilidad de un marcador almacenado, para esto se carga el mapa al acceder a la web, el usuario pulsa el botón izquierdo sobre el ícono de un marcador, esto hace que la ventana sea mostrada con la información de este marcador. En esta ventana de información aparece el botón Eliminar POI, que es seleccionado por el usuario. Ahora se elimina el marcador de la base de datos y del mapa.

4.6. StoryBoards

En este apartado se diseñará un prototipo de lo que será la interfaz del usuario, aquí mostraremos las diferentes opciones que podrá desarrollar el usuario en el uso de la aplicación.

La utilización de Storyboard nos dará una primera visión de cómo será el producto final, a partir de aquí podemos desarrollar la interfaz basándonos en estos datos.

Para su realización se ha utilizado una tableta digital, en la que se pueden modificar posteriormente cualquier nueva adaptación.

Ahora vamos a mostrar en unas imágenes de cómo sería nuestra interfaz.

En la página principal se mostrará un mapa:

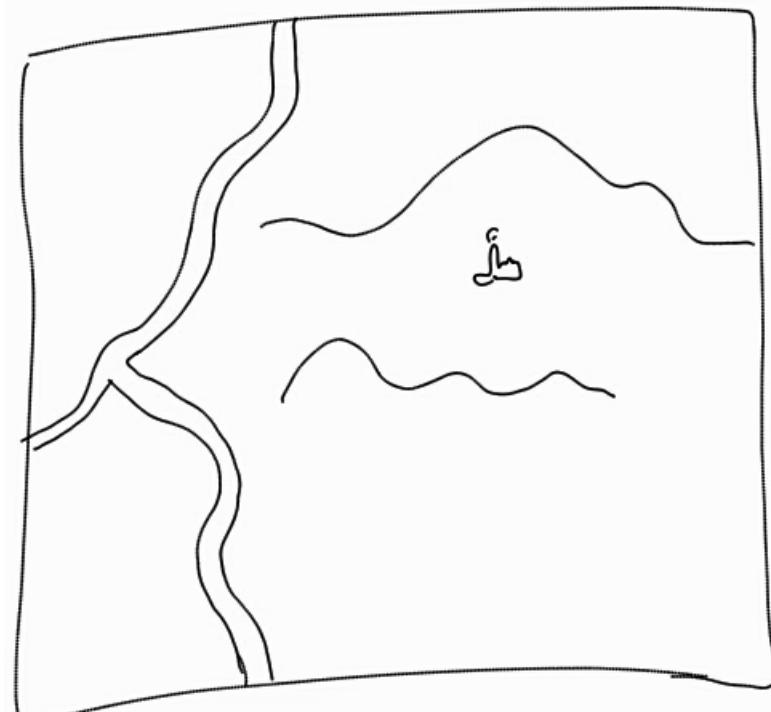


Ilustración 31 - Storyboard Mapa principal

En este mapa el usuario tendrá dos opciones, podrá pulsar el botón derecho o izquierdo del ratón.

Si pulsa el botón izquierdo, se mostrará la capa de visibilidad en ese punto:

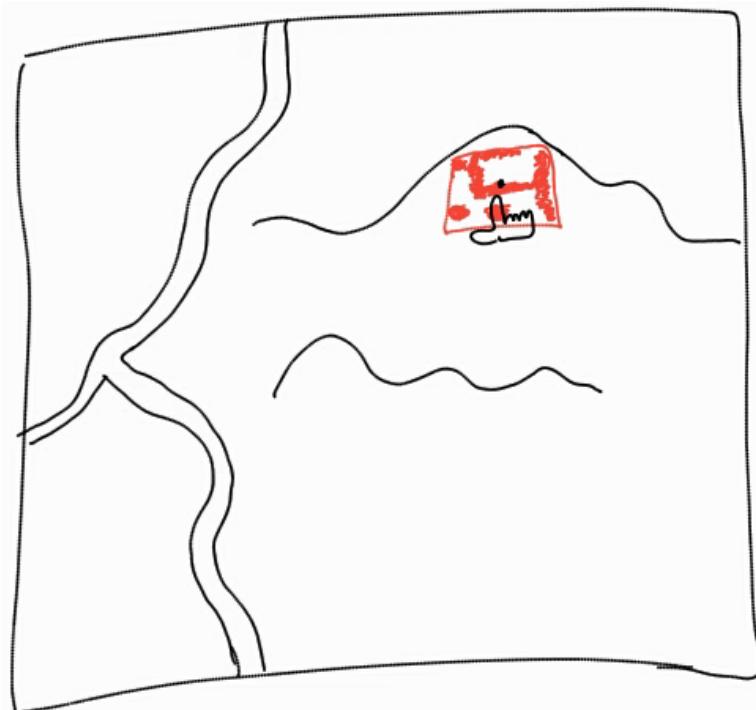


Ilustración 32 - Storyboard Capa de Visibilidad

Si utiliza el botón derecho, se mostrará un nuevo marcador y aparecerá su ventana de información, con la que el usuario puede volver a interactuar.

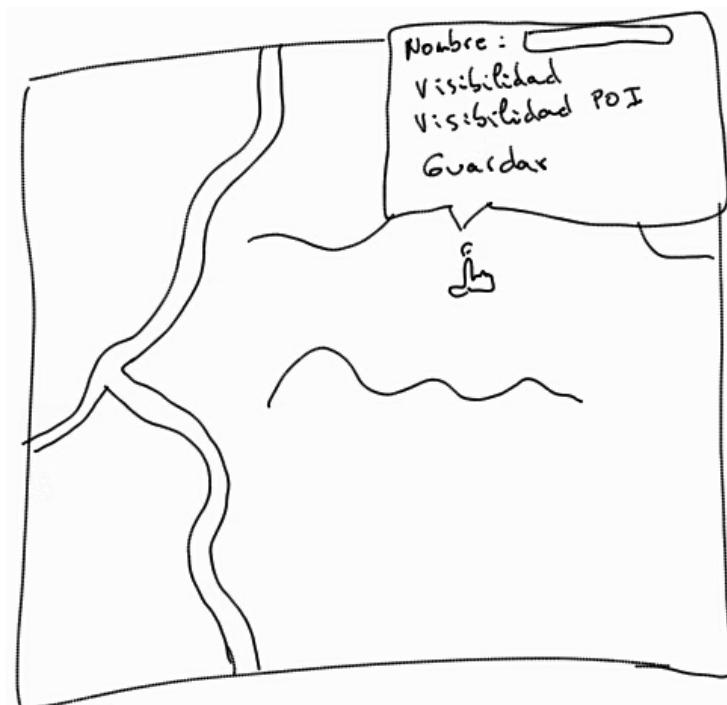


Ilustración 33 - Storyboard Nuevo Marcador

Al aparecer la ventana de información el usuario podrá calcular la visibilidad de ese marcador, calcular la visibilidad respecto a los otros marcadores o guardar el marcador, además podrá elegir si quiere que sea una torre o un castillo.

Una vez guardado el marcador, podemos pulsar sobre su ícono y nos volverá a aparecer la ventana de información.

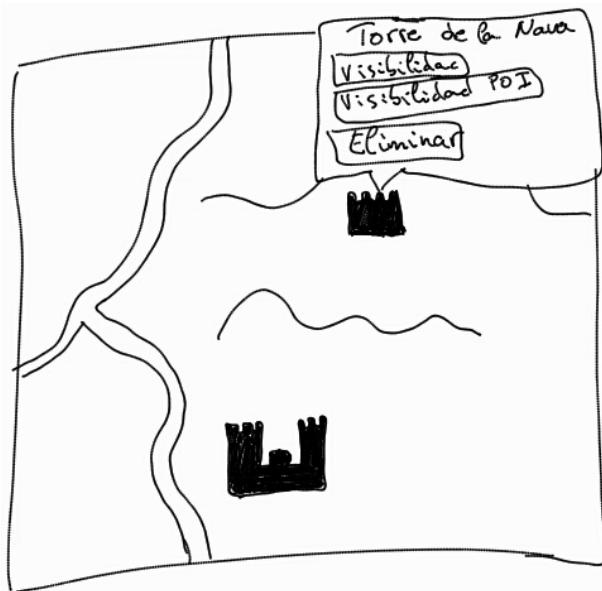


Ilustración 34 - Storyboard Infowindow

Desde aquí, podremos calcular la visibilidad del marcador que nos dará el siguiente resultado:

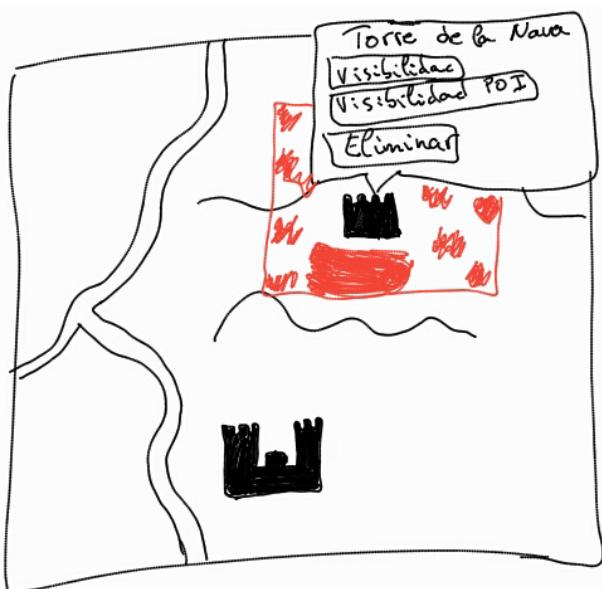


Ilustración 35 - Storyboard Visibilidad

O podremos calcular la visibilidad de los otros marcadores respecto al seleccionado:

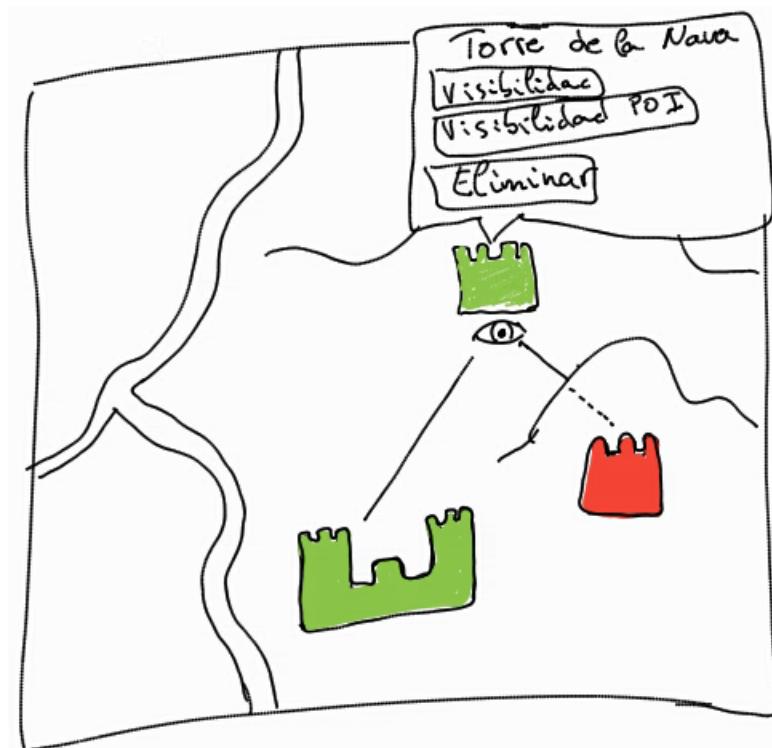


Ilustración 36 - Storyboard Visibilidad POIs

5. Implementación

En este capítulo hablaremos sobre los recursos utilizados, explicando detalladamente el uso de cada uno, así como de su función en este trabajo.

5.1. Servidor WAMP



Ilustración 37 - WAMP Server

WampServer es un entorno de desarrollo web de Windows. Permite crear aplicaciones web con Apache, PHP, MySQL y administrar la base de datos mediante PhpMyAdmin.

Para este proyecto se ha usado WampServer 2.2 que incluye Apache 2.4.2, PHP 5.4.3, MySQL 5.5.24 y 24 y PhpMyAdmin 3.5.1.



• Apache

Ilustración 38 - Apache

Apache es un servidor web HTTP de código abierto mantenido por una comunidad organizada bajo la Apache software Foundation, es muy modular y multiplataforma.



• PHP

Ilustración 39 - PHP

PHP (Hypertext Preprocessor) es un lenguaje de programación interpretado del lado de servidor que se utiliza para la generación de páginas web de forma dinámica. Este lenguaje es de código abierto, gratuito y multiplataforma.



• MySQL

Ilustración 40 - MySQL

MySQL es un sistema de gestión de bases de datos relacional, fue creada por la empresa sueca MySQL Ab, licenciado bajo la GPL de la GNU

5.2. Google Maps JavaScript API



Ilustración 41 - Google Maps JavaScript API

La API de Google Maps es un conjunto de reglas y especificaciones que nos proporciona Google para poder trabajar con sus aplicaciones.

Estas bibliotecas amplían la funcionalidad de las aplicaciones añadiendo nuevas funciones, implementando patrones de diseño o haciendo algunas tareas un poco más fáciles.

5.2.1. Clave de Google Maps

Google Maps dispone de un sistema de autenticación mediante clave para usar su API, esta clave se puede obtener mediante una cuenta de Gmail.

En este caso el plan para la API Key usada ha sido la estándar, ya que es gratuita, aunque tiene algunas limitaciones.

A screenshot of a web page titled "Standard Usage Limits". It contains a list of usage limits for users of the standard API:

- 2,500 free requests per day, calculated as the sum of [client-side](#) and server-side queries.
- 512 locations per request.
- 50 requests per second, calculated as the sum of [client-side](#) and server-side queries.

Ilustración 42 - Google Maps Standard Usage Limits

Aunque estas limitaciones pueden ser eliminadas si se contrata un plan Premium.

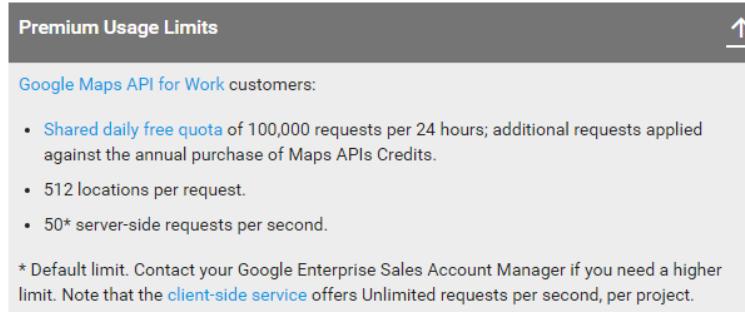


Ilustración 43 - Google Maps Premium Usage Limits

Para crear una nueva clave es necesario acceder a la API Google Console:

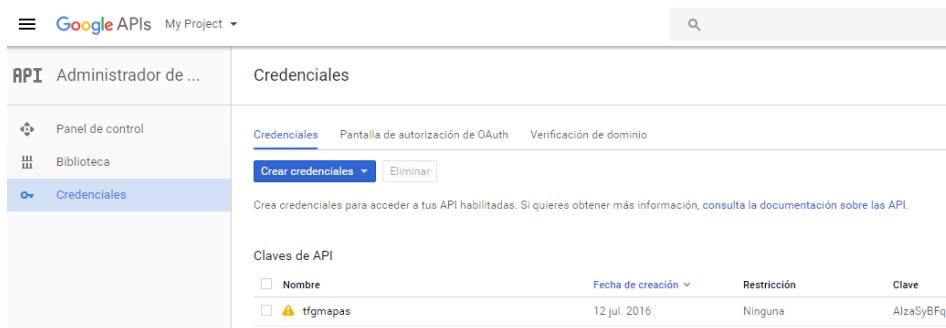


Ilustración 44 - Google Maps API Console

Una vez creada esta clave nos permitirá usar la API de Google Maps añadiéndola a nuestro proyecto usando la siguiente URL.

```
<script src="https://maps.googleapis.com/maps/api/js?key=YOUR_API_KEY" type="text/javascript"></script>
```

Ilustración 45 - Google Maps Key

5.2.2. Creación del Mapa

Basándonos en las especificaciones dadas en la API de Google podemos crear un mapa especificando el zoom, el centro del mapa y el tipo de mapa.

```
var map = new google.maps.Map(document.getElementById('map'), {  
    zoom: 13,  
    center: {lat: localizacion.lat(), lng: localizacion.lng()},  
    mapTypeId: 'terrain'  
});
```

Ilustración 46 - Nuevo Mapa

5.2.3. Google Maps Elevation

Google Maps dispone de dos funciones para obtener la altura, ambas son asíncronas por lo que hay que tratarlas debidamente:

- `elevator.getElevationForLocations`: Devuelve la altura de una determinada localización.
- `elevator.getElevationalongPath`: Devuelve un vector con las alturas de un número de distancias intermedias especificado en la función de entre dos puntos.

Esta función debe conocer el punto inicial, el punto final y el número intermedio de alturas a obtener.

Hay que saber que en el número intermedio de alturas entra también el punto de visión y el punto objetivo, por tanto si a esta función le pasamos como parámetro 8 resultados el resultado real será:

- **Objeto [0]=Punto de visión**
- **Objeto [1...6]=Puntos intermedios**
- **Objeto [7]=Punto Objetivo**

En este caso, esta función de Google es llamada desde otra propia nuestra, a la que le pasamos los datos deseados. Se ha introducido dentro de una función propia ya que como se ha dicho antes, es una función asíncrona, con lo cual debemos hacer uso de la promesa de JavaScript (promise).

Esto nos permite poder recoger los datos una vez se hayan terminado de ejecutar las diferentes llamadas que se le hacen a esta función. Así aseguramos que todos los datos pedidos estén correctamente ejecutados.

```
function initialize(situaciones, localizacionInicial, num_divisiones) {
    var elevator = new google.maps.ElevationService; //Iniciamos el servicio de alturas de google
    var listElevations=new Array(2); //Vector con la posicion inicial y final
    listElevations[0]=localizacionInicial; //posicion inicial
    listElevations[1]=situaciones; //posicion final que se pasa como argumento
    return new Promise(function(resolve,reject) {

        elevator.getElevationAlongPath({
            'path': listElevations,
            'samples': num_divisiones
        },function(results, status) {
            if (status === google.maps.ElevationStatus.OK) {
                if (results) resolve(results);
                else console.log('No results found');

            } else {
                console.log('Elevation service failed due to: ' + status);
            }
        });
    });
}
```

Ilustración 47 - Google Maps getElevationAlongPath

Con los datos obtenidos de esta función posteriormente podemos calcular si un punto es visible o no desde el punto de visión, ya que tenemos los puntos intermedios entre ellos, calculando con estos su visibilidad.

5.2.4. Google Maps Markers

Google Maps dispone del uso de marcadores, un marcador tan solo identifica una ubicación en el mapa. Los marcadores pueden ser modificados y editados, como por ejemplo su posición, ícono o la ventana de información.

```
var marker = new google.maps.Marker({
  position: MapPos,
  map: map,
  draggable:DragAble,
  animation: google.maps.Animation.DROP,
  title:MapTitle,
  icon: iconPath
});
```

Ilustración 48 - Nuevo Marcador

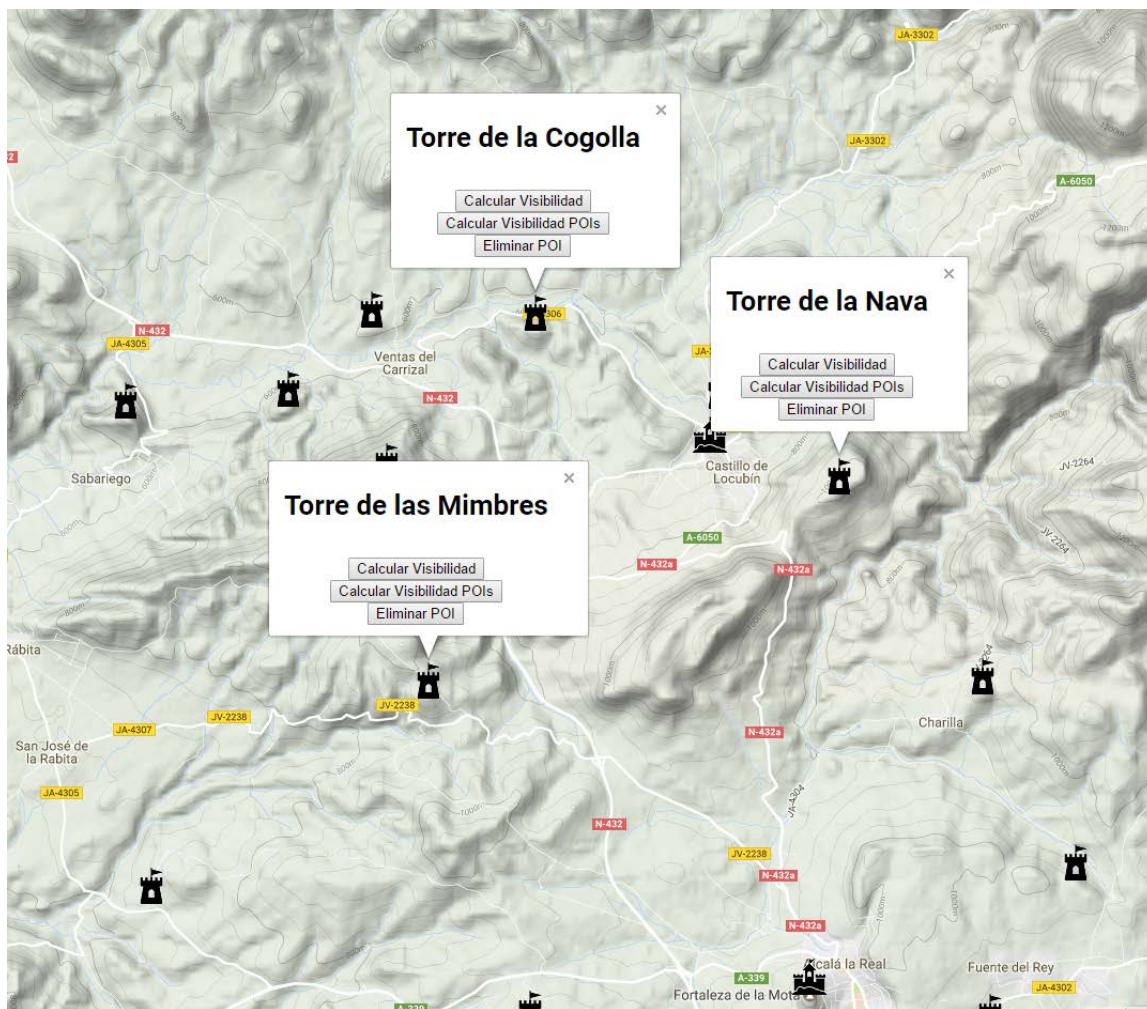


Ilustración 49 - Ejemplo Marcadores

5.2.5. Superposiciones de Imágenes

Google Maps permite la superposición de imágenes en el mapa, estas imágenes usan unos parámetros de longitud y latitud, por lo que se mueven cuando arrastras el mapa o cuando aplicas un zoom.

Se usa mediante la clase OverlayView, esta clase dispone de varios métodos que son necesarios incluir en el proyecto para su uso.

Funciones incluidas: USGSOverlay, onAdd, draw, onRemove, hide, show, toggle, toggleDom.

Ejemplo de uso:

```
var bounds = new google.maps.LatLngBounds(
  new google.maps.LatLng(situations[10][0].lat(),situations[10][0].lng()), //Coordenadas de la imagen
  new google.maps.LatLng(situations[0][10].lat(),situations[0][10].lng()));
overlay = new USGSOverlay(bounds, 'data:image/png;base64,'+capa.getBase64(), map); //Añado la imagen al mapa
```

Ilustración 50 - USGSOverlay

Quedando un resultado como este:

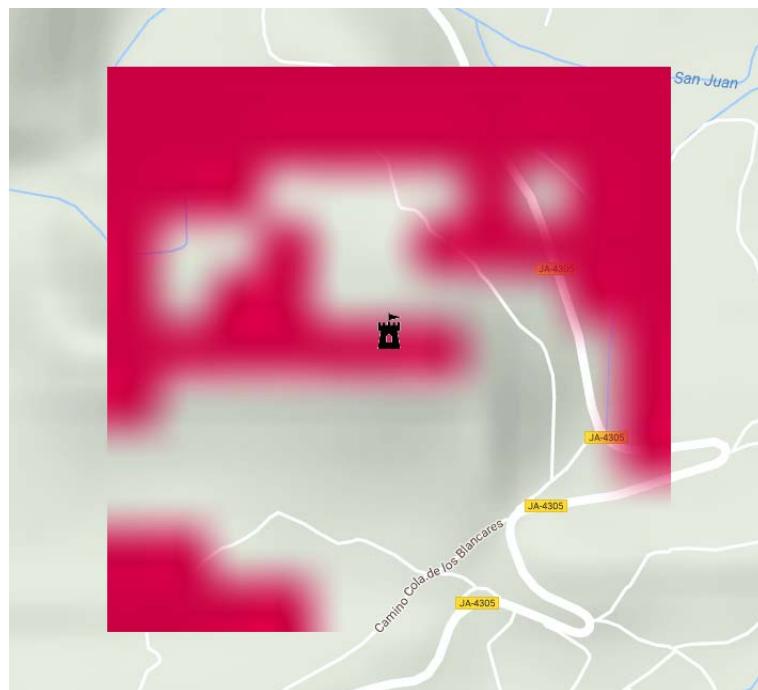


Ilustración 51 - Visibilidad

5.3. Funciones

Este subcapítulo explicarán las principales funciones que se han desarrollado para este TFG.

5.3.1. Cálculo de la capa de visibilidad

Tomando el mapa con la orientación de la imagen, se obtienen que los valores de latitud y longitud varían de la siguiente manera:

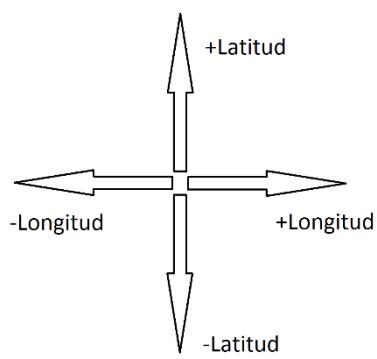


Ilustración 52 - Longitud y Latitud Referencia

Como podemos ver en la imagen, la latitud aumenta al norte de España, mientras disminuye al sur y aumenta al este mientras disminuye al oeste. Esto si nos basamos en los puntos cardinales reales sería un aumento de la latitud al noroeste y aumento de la longitud al sureste.

Por tanto, partimos del punto pulsado por el usuario y obtenemos la latitud y longitud de ese punto, ahora suponemos que superponemos una matriz sobre el punto elegido, donde el centro de la matriz coincide con el punto pulsado por el usuario, con lo que mediante unos cálculos de longitud se calculan todas las localizaciones de cada una de las casillas de la matriz superpuesta.

Cada uno de los puntos de la matriz se convertirá en un **Punto Objetivo**, para el cual se obtendrán diferentes alturas entre este punto y el **Punto de Visión**.

Como he dicho anteriormente para calcular la capa de visibilidad se parte de la localización del **Punto de Visión** y se obtiene cada uno de los **Puntos Objetivo** de nuestra matriz de visibilidad.

Estos datos son obtenidos mediante la distancia de cada una de nuestras casillas, de esta manera se obtiene una matriz que contendrá en cada casilla diferentes localizaciones que serán **Puntos Objetivo**. Además se calcularán los datos de distancia para cada uno de estos puntos.

Pseudocódigo:

```

VARIABLES
Localizacion: Punto de vision
LocalizacionAux: Punto objetivo
Casillas: Numero de filas/columnas de la matriz de la capa de visibilidad
mLat: Grados de Latitud que equivalen X metros
mLng: Grados a de Longitud que equivalen X metros
Situations: Matriz de coordenadas de tamaño Casillas X Casillas

INICIO
    PARA i=0 HASTA Casillas CON INCREMENTO +1
        Situations.latitud+=mLat;
        INICIO
            PARA i=0 HASTA Casillas CON INCREMENTO +1
                Situations.longitud+=mLng;
                Consultar y guardar datos de altura y distancia de Situations.
            FIN_PARA
        FIN
    FIN_PARA
FIN

Funcion cargarVisibilidad

Aplicar capa al mapa

```

Ilustración 53 – Pseudocódigo Cálculo Coordenadas

Con estos datos es el momento de obtener las alturas. Las alturas son obtenidas dentro de la función “initialize” explicada en el apartado **4.2.3 Google Maps Elevation**, en la que se llama a la función de google “getElevationAlongPath”, esta función es asíncrona, por lo que a la hora de obtener los datos se deben utilizar las herramientas de JavaScript para datos asíncronos.

Con esto ya se tendría la altura del punto de visión, las alturas en los puntos objetivo y las alturas intermedias entre el punto de visión y los puntos objetivo, por lo que se puede empezar a calcular la visibilidad.

5.3.2. Función cargarVisibilidad

Esta función es la encargada de calcular la visibilidad de la matriz de visibilidad, esto se consigue mediante un bucle en el que se obtienen los datos de las alturas para cada punto, como se dijo anteriormente se tiene para cada punto de la matriz la altura en el punto de visión, la altura de seis puntos intermedios y la altura del punto final.

Ahora para cada punto, primero se comprueba si el punto de visión es mayor o menor al punto final basándonos en el siguiente razonamiento:

En el caso de que el punto de visión sea menor que el punto objetivo, para que el punto objetivo sea visible el ángulo formado entre ambos será el mayor a todos los ángulos de los puntos intermedios.

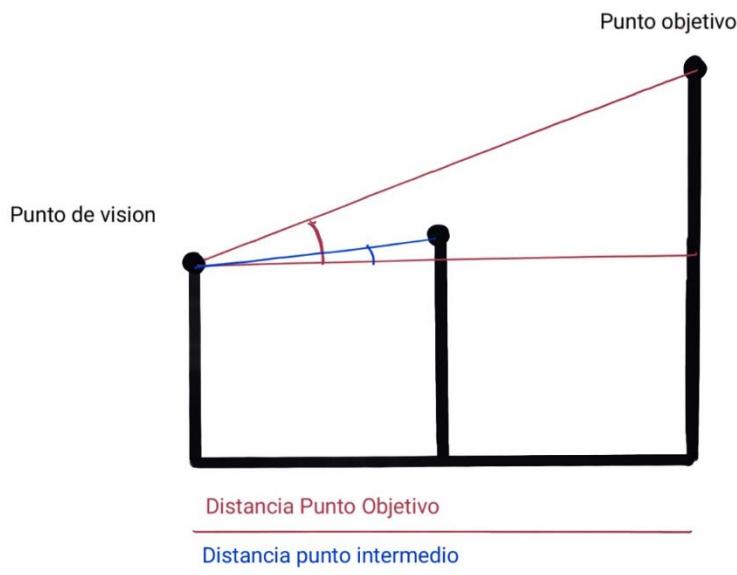


Ilustración 54 - Visibilidad Punto de Visión Inferior

Sin embargo, en el caso de que el punto de visión sea mayor al punto objetivo, el punto objetivo será visible si el ángulo formado entre ambos es el menor de todos los ángulos que forman los puntos intermedios con el punto de visión.

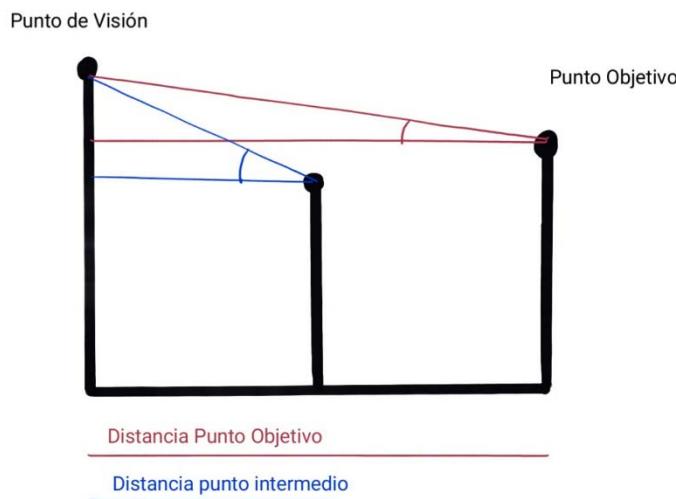


Ilustración 55 - Visibilidad Punto Visión Superior

Mediante este razonamiento el Pseudocódigo de la función sería el siguiente:

```

vAlturas: Vector de alturas entre el Punto de vision y el objetivo actual
distancia: Distancia desde el punto de vision al objetivo actual

INICIO
    PARA i=0 HASTA Casillas*Casillas CON INCREMENTO +1
        vAlturas: nuevas alturas
        distancia: nueva distancia
        INICIO
            PARA i=0 HASTA vAlturas CON INCREMENTO +1
                Calcular angulos entre Punto de vision y vAlturas
                Si Punto de vision MENOR Punto Objetivo
                    Buscar el anguloMayor
                SI NO
                    Buscar el anguloMenor
                FIN_PARA
            FIN
            SI anguloMayor IGUAL anguloObjetivo
                VISIBLE
            SI NO
                NO VISIBLE
            FIN_PARA
        FIN
    FIN

```

Ilustración 56 - Pseudocódigo Calcular Visibilidad

5.3.3. Visibilidad de los POI

Para la visibilidad de los POI el procedimiento es el mismo que para el cálculo de la capa de visibilidad, las únicas modificaciones son, que como tenemos menos puntos objetivos que al calcular la capa de visibilidad, podemos permitirnos aumentar el número intermedio de datos obtenidos de Google Maps Elevation.

En este caso lo hemos aumentado de 6 puntos intermedios a 18, obteniendo así un resultado mucho más real.

Otra de las modificaciones es que en este caso en vez de pintar la capa de visibilidad lo que hacemos es modificar el color del Marcador estableciendo un color verde si el POI es visible y un color rojo si el POI no es visible.

El procedimiento anterior da el siguiente resultado de la imagen.

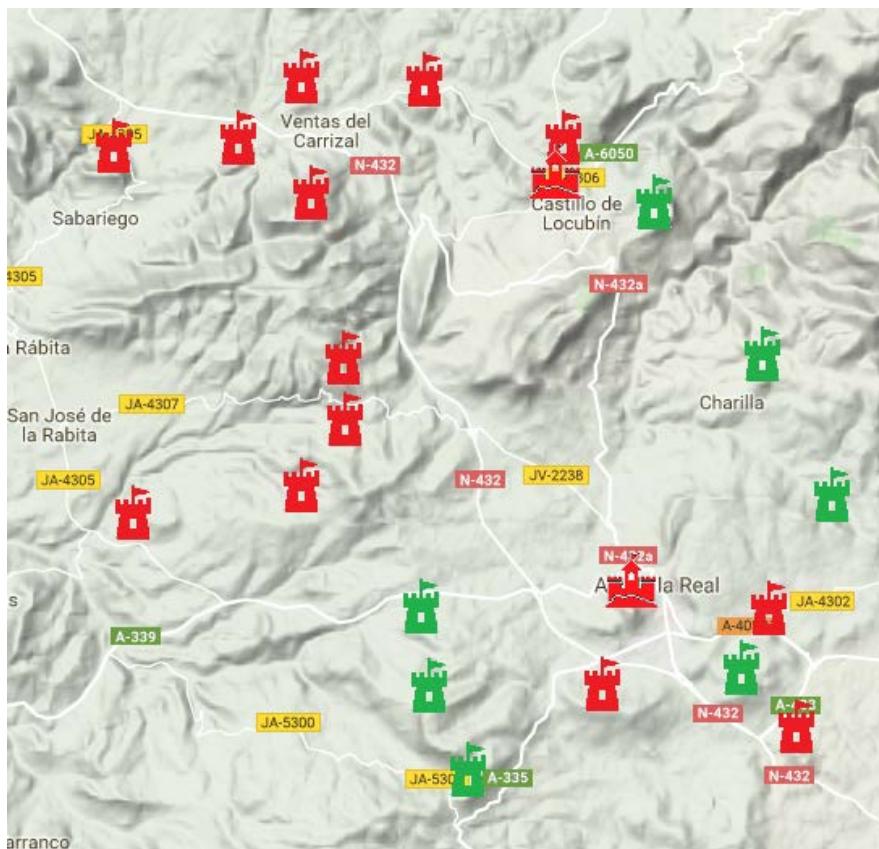


Ilustración 57 - Visibilidad POIs

5.3.4. Otras Funciones y Características

En este pequeño apartado se va a dar una explicación general del código, explicando algunas de las características más relevantes de su funcionamiento.

Cuando el usuario accede a la web, se procesa el mapa, esto lo que hace es una llamada a la API de Google Maps, en la que mediante la nuestra clave nos devuelve el mapa mostrado en la aplicación.

Posteriormente el usuario puede seleccionar un punto del mapa, si fuese con el botón izquierdo, se pasaría a calcular la matriz de puntos y la capa de visibilidad.

La matriz de puntos está formada por las localizaciones, que son los puntos objetivos finales y que representan la localización en la superposición de la matriz en el mapa. Esta matriz está formada por 11 casillas, ya que debido al problema de la clave Standard de Google Maps no nos permite hacer más peticiones al servidor en un tiempo determinado, y cada una de estas casillas tiene aproximadamente unos 100 metros, por lo que nuestra matriz sería de aproximadamente 1Km. Se ha decidido dar estos datos, para que la aplicación devuelva resultados lo más reales posible. Además, se ha implementado un botón en la parte izquierda superior, que cambia el área de visibilidad a 50Km. Este botón sólo se ha implementado para ver cómo serían los datos que se obtendrían en caso de tener la clave de Google Maps Premium, ya que los datos que se obtienen son los resultantes de obtener una altura cada 8 Km aproximadamente, por tanto no se toman los valores necesarios para que los datos se acoplen a la realidad.

Una vez tenemos nuestra matriz con las localizaciones, podemos pasar a preguntar a Google Maps la altura de estas posiciones. Como se ha dicho anteriormente, Google Maps utiliza funciones asíncronas por tanto hay que recoger los datos mediante el uso de Promise de JavaScript.

En este caso recogemos los datos de las alturas de tanto el punto de visión, el punto objetivo y los puntos intermedios y los almacenamos en el LocalStorage de JavaScript. Esto nos permite almacenar los datos en el navegador del cliente, eliminando del servidor la escritura y lectura de datos.

En el localStorage son almacenados mediante una clave de tipo texto que es identificativa para cada valor almacenado, lo que se hace es almacenar cada uno de los datos con un contador para luego ser recogidos y poder ser tratados correctamente.

Ahora vamos a tratar los datos almacenados, lo que se hace es extraer el número de datos que corresponden a cada posición de la matriz, y se calcula la visibilidad del punto objetivo, en función de si es mayor o menor que el punto de visión se buscará el ángulo mayor o menor. Con esto obtenemos la visibilidad de cada una de las casillas de la matriz, ahora es el paso de crear la capa de visibilidad.

Esta capa es creada mediante una librería llamada PNGlib, lo que se hace es darle a la imagen un fondo transparente, ahora se comprueba si la casilla de la matriz es visible o no, y en función de esto se colorea la casilla de la imagen o no.

El último paso sería superponer la imagen creada anteriormente al mapa, para esto se hace uso de la librería USGSOverlay, disponible en la API de Google Maps.

Mediante los datos de la posición de la esquina superior derecha y la esquina inferior izquierda se superpone la imagen al mapa.

Por otro lado, el usuario puede pulsar con el botón derecho, esto llamará al evento en el que se creará un nuevo marcador en el mapa y desde el cual se podrá lanzar la capa de visibilidad, o calcular la visibilidad de los demás marcadores respecto del seleccionado. Este marcador puede también ser almacenado en la base de datos.

Cuando se calcula la visibilidad de los demás marcadores, el proceso es muy similar al de la capa de visibilidad, pero como se tienen menos marcadores que casillas de la matriz podemos permitirnos aumentar el número de datos intermedios entre dos puntos a obtener, creando así resultados más reales.

Para esto lo que hacemos es aumentar el número de datos intermedios de 6 a 18 para cada uno de los marcadores, y según si es visible o no, mediante el método de la capa de visibilidad, modificamos el icono del marcador, obteniendo un color rojo si no es visible y un color verde cuando si es visible.

Estos marcadores pueden ser almacenados y eliminados de la base de datos, para lo que se ha utilizado JQuery y PHP, lo que se hace es mandar un petición http al script PHP, el script recoge los datos enviados y los procesa, ejecutando la orden necesaria.

6. Instalación y Configuración del Servidor

En este capítulo se detallarán los pasos a seguir para la instalación y configuración del servidor para un correcto funcionamiento de este.

6.1. Instalación

Lo primero que se hará es instalar una maquina con Windows 10 64 bits.

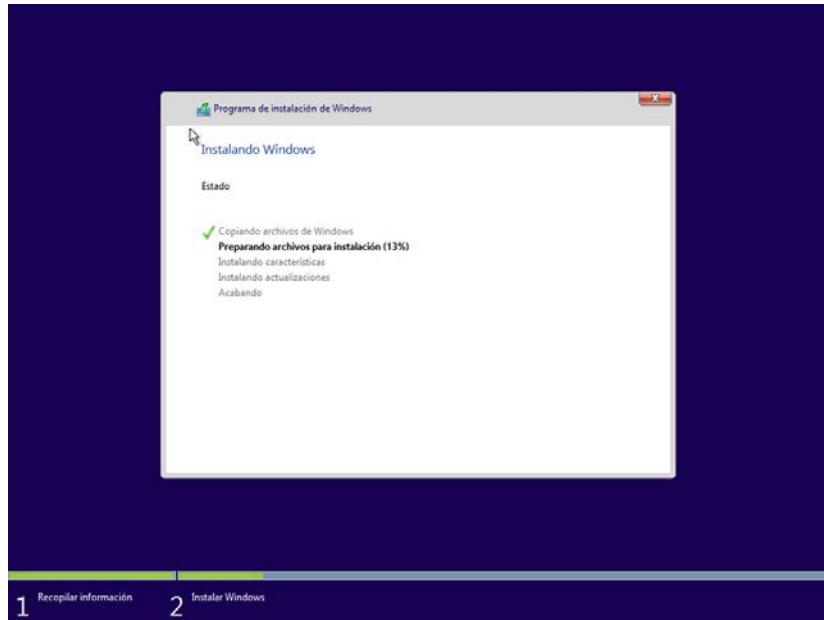


Ilustración 58 - Instalación Windows

Una vez instalado el sistema operativo vamos a proceder a la descarga del software necesario.

En este caso necesitaremos el servidor WAMP, que puede ser descargado de la página <http://wampserver.es>

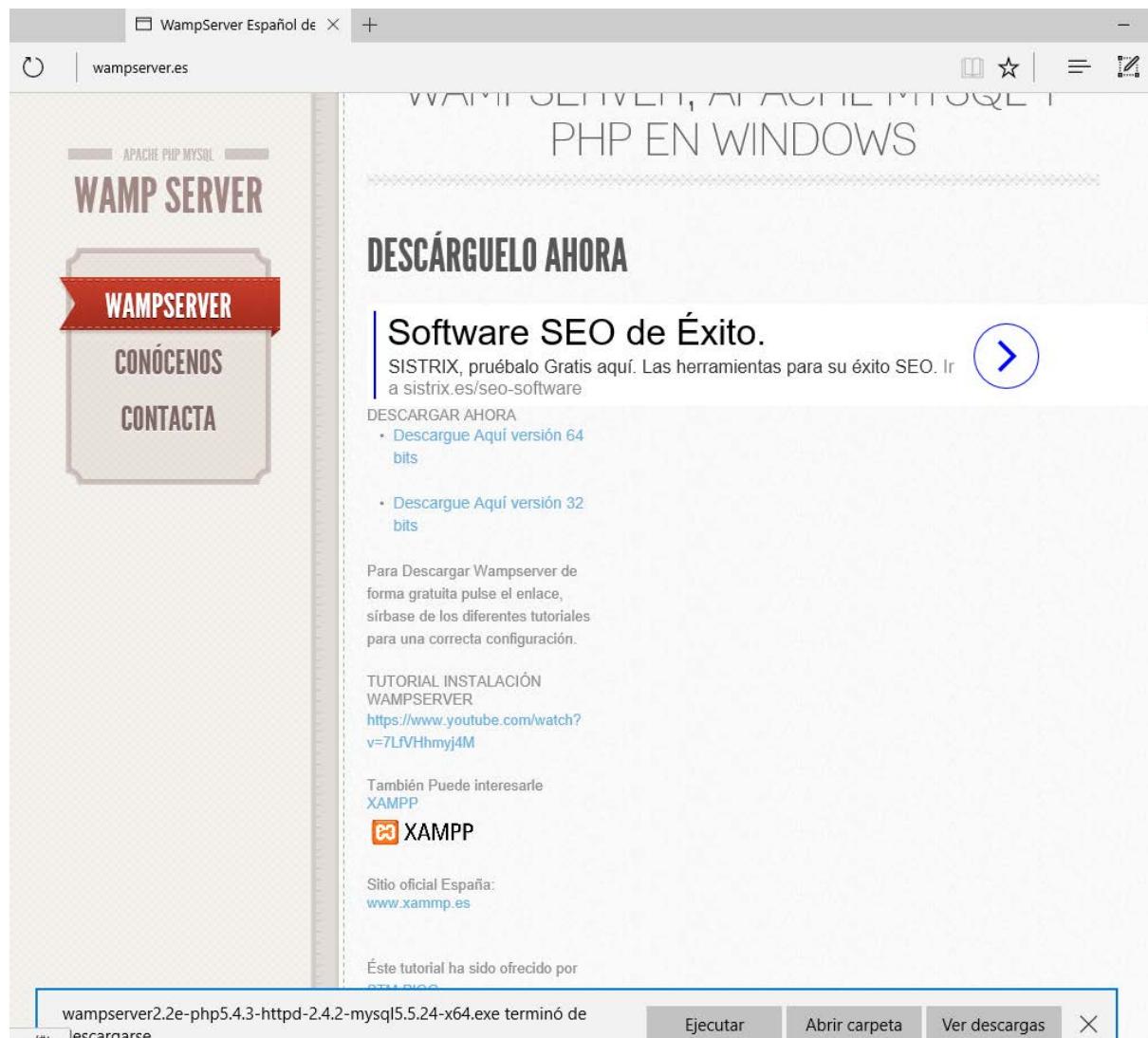


Ilustración 59 - Descarga WAMP

Se ha descargado la versión de 64 bits, ya que el sistema operativo instalado tenía esta configuración.

Procedemos a su instalación, que es bastante sencilla e intuitiva.

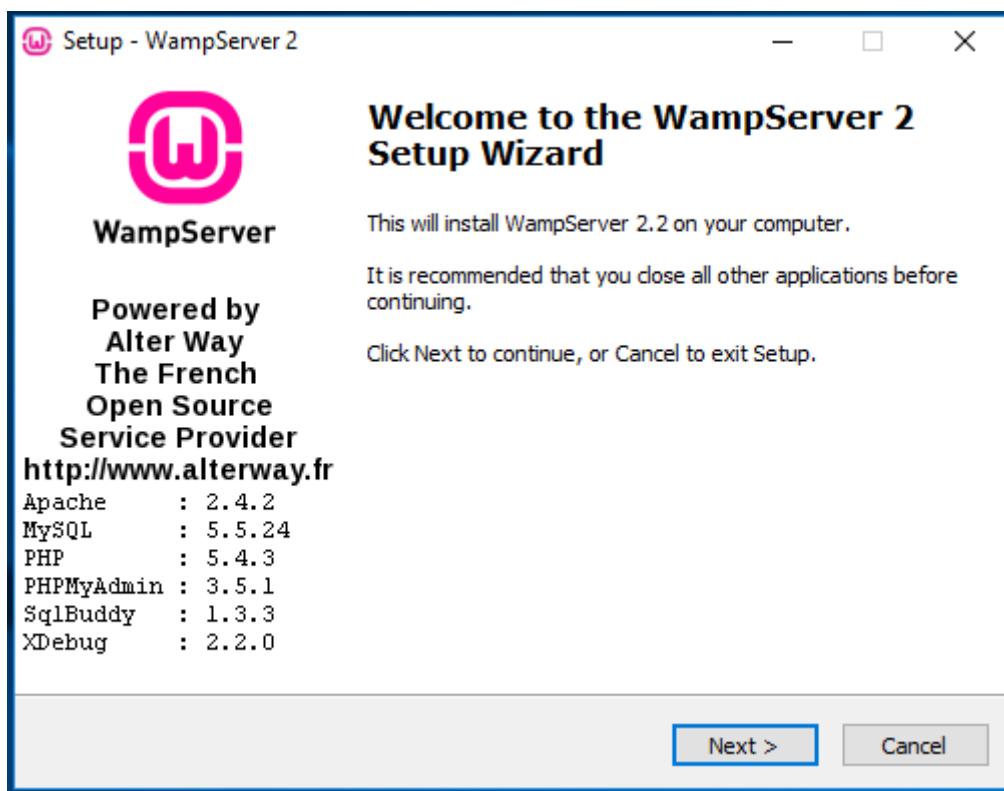


Ilustración 60 - Instalación WAMP

Una vez instalado podemos acceder al servidor web mediante la dirección de nuestro localhost.

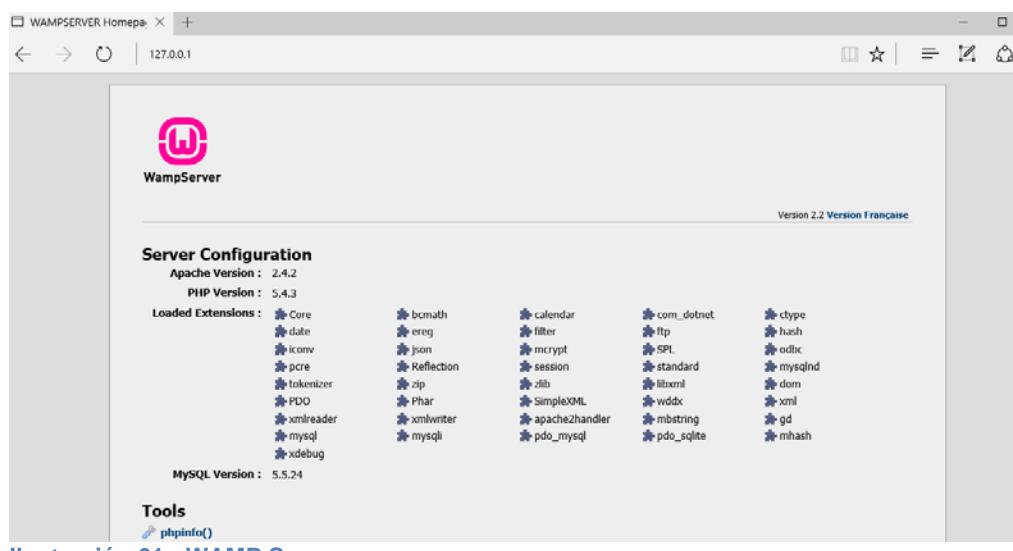


Ilustración 61 - WAMP Server

Ahora copiaremos el contenido del proyecto en la ruta de nuestro servidor web, que en este caso está en C:\wamp\www\mapasdevisibilidad

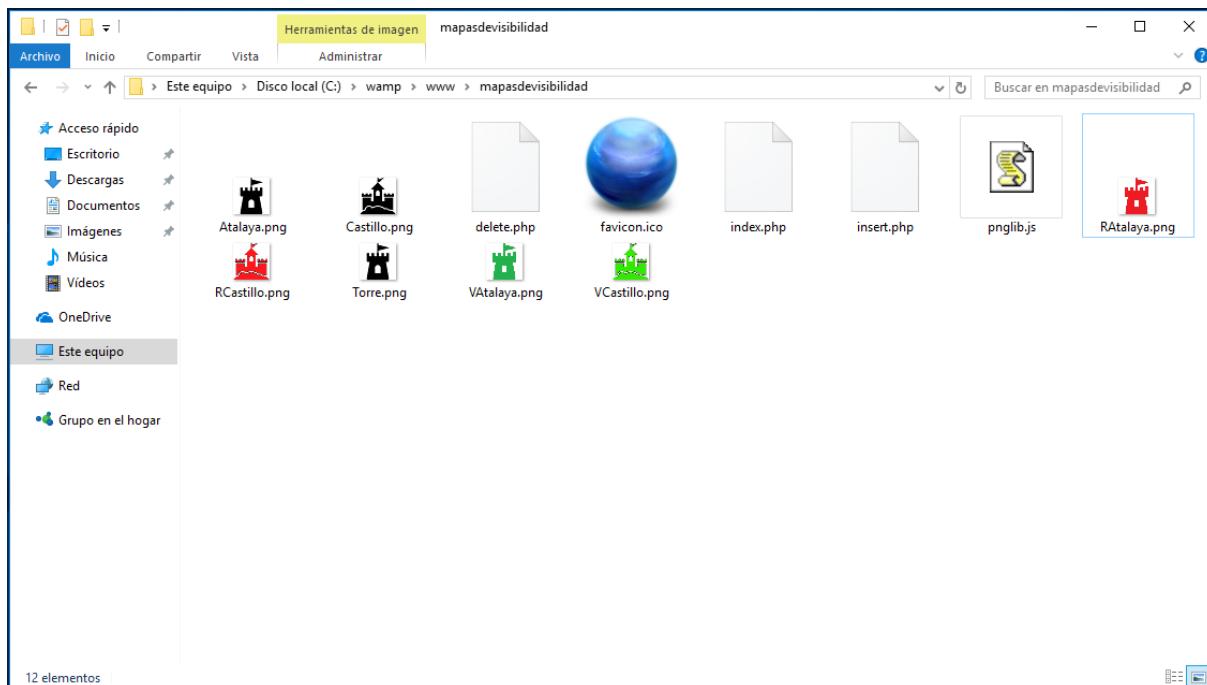


Ilustración 62 - Carpeta Web

Ahora lo siguiente será la configuración de la base de datos, para esto nos serviremos de la herramienta PhpMyAdmin incluida en WAMP server, instalado anteriormente.

Para acceder a esta herramienta accedemos a la URL: <http://127.0.0.1/phpmyadmin>

The screenshot shows the PhpMyAdmin interface at the URL 127.0.0.1/phpmyadmin. The top navigation bar shows 'localhost' and various menu options like 'Bases de datos', 'SQL', 'Estado actual', 'Usuarios', 'Exportar', 'Importar', and 'Más'. The main area is titled 'Bases de datos' and shows the following table:

Base de datos	Replicación maestra
information_schema	Replicado/a
mysql	Replicado/a
performance_schema	Replicado/a
test	Replicado/a

Total: 4

Below the table, there are buttons for 'Activar las estadísticas' and a note: 'Nota: Activar aquí las estadísticas de la base de datos podría causar tráfico pesado entre el servidor web y el servidor MySQL.'

Ilustración 63 - Base de Datos

Lo siguiente será crear un usuario con permisos para nuestra base de datos, para llevar a cabo esta función accedemos a la pestaña de usuarios y pulsamos en agregar usuario.

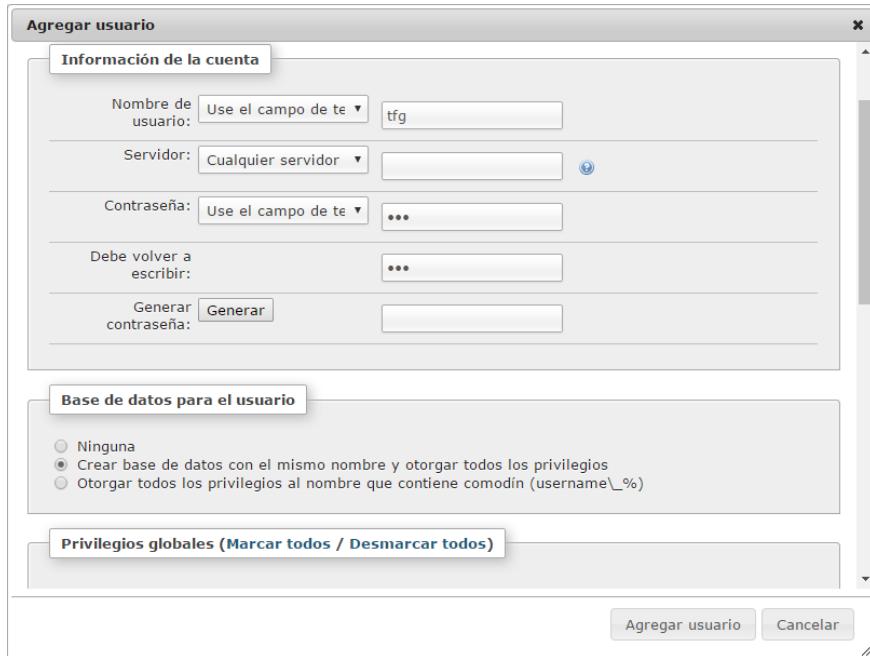


Ilustración 64 - Usuario Base de Datos

Ahora importaremos nuestro fichero de la base de datos adjuntado con el proyecto, así creará las tablas necesarias y algunos de los datos ya precargados.

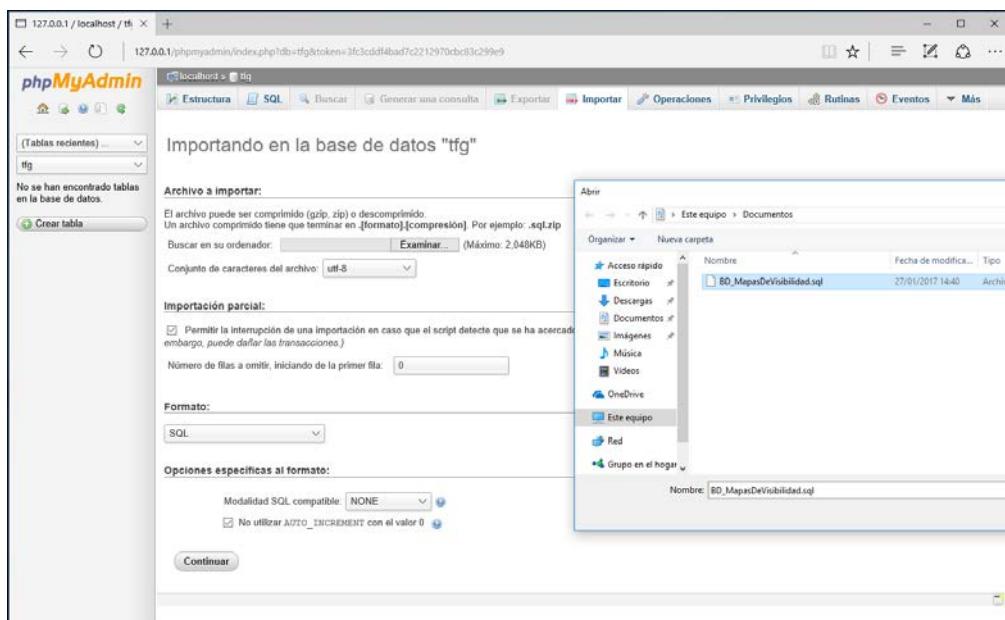


Ilustración 65 - Carga Base De Datos

Como podemos comprobar los datos se han cargado correctamente, así ya tendremos algunos marcadores creados al iniciar nuestra aplicación.

	<input type="checkbox"/> Editar	<input type="checkbox"/> Copiar	<input type="checkbox"/> Borrar	id	longitud	latitud	nombre	imagen
<input type="checkbox"/>	<input type="checkbox"/> Editar	<input type="checkbox"/> Copiar	<input type="checkbox"/> Borrar	17	-3.94565150141716	37.5296668208324	Castillo de la Villeta	Castillo.png
<input type="checkbox"/>	<input type="checkbox"/> Editar	<input type="checkbox"/> Copiar	<input type="checkbox"/> Borrar	15	-3.9440153539180756	37.535251296388694	Torre de Triana	Atalaya.png
<input type="checkbox"/>	<input type="checkbox"/> Editar	<input type="checkbox"/> Copiar	<input type="checkbox"/> Borrar	19	-4.01324853010483	37.53543208560044	Torre de los Ajos	Atalaya.png
<input type="checkbox"/>	<input type="checkbox"/> Editar	<input type="checkbox"/> Copiar	<input type="checkbox"/> Borrar	16	-3.9736711978912354	37.54514194919502	Torre de la Cogolla	Atalaya.png
<input type="checkbox"/>	<input type="checkbox"/> Editar	<input type="checkbox"/> Copiar	<input type="checkbox"/> Borrar	18	-3.99980403280258	37.545570468413466	Torre de Encima Hermosa	Atalaya.png
<input type="checkbox"/>	<input type="checkbox"/> Editar	<input type="checkbox"/> Copiar	<input type="checkbox"/> Borrar	14	-3.9296749234199524	37.460580891635644	Fortaleza de la Mota	Castillo.png
<input type="checkbox"/>	<input type="checkbox"/> Editar	<input type="checkbox"/> Copiar	<input type="checkbox"/> Borrar	13	-3.9248161017894745	37.5243010387458	Torre de la Nava	Atalaya.png
<input type="checkbox"/>	<input type="checkbox"/> Editar	<input type="checkbox"/> Copiar	<input type="checkbox"/> Borrar	22	-3.9907340705394745	37.49830793401747	Torre de las Mimbres	Atalaya.png
<input type="checkbox"/>	<input type="checkbox"/> Editar	<input type="checkbox"/> Copiar	<input type="checkbox"/> Borrar	72	-3.592003583908081	37.17697075362282	Torre de la Vela (Alhambra)	Castillo.png
<input type="checkbox"/>	<input type="checkbox"/> Editar	<input type="checkbox"/> Copiar	<input type="checkbox"/> Borrar	25	-3.96083950996389893	37.71784348533012	Castillo de la Peña de Martos	Castillo.png
<input type="checkbox"/>	<input type="checkbox"/> Editar	<input type="checkbox"/> Copiar	<input type="checkbox"/> Borrar	27	-4.088518023490906	37.590526834792904	Castillo de Alcaudete	Castillo.png
<input type="checkbox"/>	<input type="checkbox"/> Editar	<input type="checkbox"/> Copiar	<input type="checkbox"/> Borrar	63	-3.747245818376541	37.671643577610084	Castillo de Otiásar	Castillo.png
<input type="checkbox"/>	<input type="checkbox"/> Editar	<input type="checkbox"/> Copiar	<input type="checkbox"/> Borrar	50	-4.039444327354431	37.53397299781009	Torre de la Harina	Atalaya.png
<input type="checkbox"/>	<input type="checkbox"/> Editar	<input type="checkbox"/> Copiar	<input type="checkbox"/> Borrar	35	-3.937387615442276	37.64707383437435	Torre del Algarrobo	Atalaya.png
<input type="checkbox"/>	<input type="checkbox"/> Editar	<input type="checkbox"/> Copiar	<input type="checkbox"/> Borrar	36	-3.997562974691391	37.5261985138487	Torre de la Sierra	Atalaya.png
<input type="checkbox"/>	<input type="checkbox"/> Editar	<input type="checkbox"/> Copiar	<input type="checkbox"/> Borrar	37	-3.8869795203208923	37.47519866911902	Torre de la Boca de Charilla	Atalaya.png
<input type="checkbox"/>	<input type="checkbox"/> Editar	<input type="checkbox"/> Copiar	<input type="checkbox"/> Borrar	38	-3.900635987520218	37.45608098296875	Torre de Cascante	Atalaya.png
<input type="checkbox"/>	<input type="checkbox"/> Editar	<input type="checkbox"/> Copiar	<input type="checkbox"/> Borrar	39	-3.901877850294113	37.49887184862901	Torre de Charilla	Atalaya.png
<input type="checkbox"/>	<input type="checkbox"/> Editar	<input type="checkbox"/> Copiar	<input type="checkbox"/> Borrar	40	-3.935755491256714	37.4431982713073	Torre de la Dehesilla	Atalaya.png
<input type="checkbox"/>	<input type="checkbox"/> Editar	<input type="checkbox"/> Copiar	<input type="checkbox"/> Borrar	41	-4.03530165553093	37.47210355363936	Torre de Fuente Alamo	Atalaya.png
<input type="checkbox"/>	<input type="checkbox"/> Editar	<input type="checkbox"/> Copiar	<input type="checkbox"/> Borrar	42	-3.9064349234104156	37.44599965133462	Torre de la Moraleja	Atalaya.png
<input type="checkbox"/>	<input type="checkbox"/> Editar	<input type="checkbox"/> Copiar	<input type="checkbox"/> Borrar	43	-3.9744342863559723	37.45643868611195	Torre de los Pedregales	Atalaya.png
<input type="checkbox"/>	<input type="checkbox"/> Editar	<input type="checkbox"/> Copiar	<input type="checkbox"/> Borrar	44	-3.8449063897132874	37.39326906494865	Torre del Quejigal	Atalaya.png

Ilustración 66 - Datos Base de Datos

Si accedemos a la aplicación podemos ver que funciona correctamente.

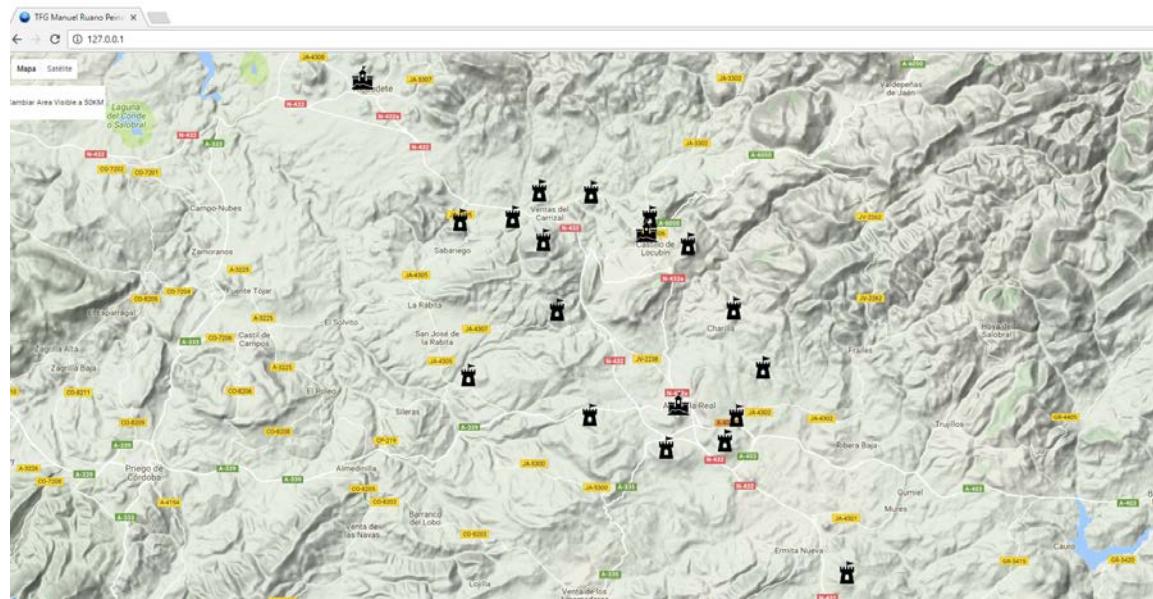


Ilustración 67 - Aplicación

7. Conclusiones y Posibles Mejoras

Uno de los problemas de este proyecto, es la necesidad del pago de la API de Google Maps, este problema podría solucionarse de diferentes formas.

Po un lado, una solución sería consultar las alturas mediante un modelo digital del terreno, precargado con anterioridad, en nuestra aplicación. Esto supone también la limitación de las consultas, ya que sólo se podrían consultar aquellas zonas incluidas en los datos del modelo digital cargado.

Por otro lado, otra solución sería ir almacenando las consultas que se le hacen al servidor de Google en nuestra base de datos, así si consultamos datos que ya han sido procesados anteriormente, tan solo tendríamos que hacer la consulta a nuestra base de datos.

Para ambas soluciones, deberíamos hacer uso de la interpolación, ya que raramente se utilizarían las mismas coordenadas, por lo tanto se haría una interpolación de entre las más cercanas, para obtener una solución lo más acertada posible.

También sería interesante, que a la hora de seleccionar un punto del mapa, no solo se trate ese punto, ya que por ejemplo un castillo, está construido a lo largo de un área, y por lo tanto si tan solo valoramos un punto no da un dato totalmente real. Sería conveniente lanzar, desde varios puntos cercanos al elegido, el cálculo de la visibilidad. Aquí el problema encontrado es el número de peticiones que Google Maps nos permite lanzar, por lo que para esto se necesita una clave Premium.

Otra mejora para este proyecto, sería su implementación en una aplicación para móvil, dado que para Android se puede programar en Java y este proyecto se ha programado en JavaScript, los cambios necesarios para esta adaptación sería bastante sencilla, además Google Maps también dispone de una api para los mapas de Android, por lo que disminuye la complejidad de esta opción.

Mientras se desarrollaba, se pensó modificar la capa de visibilidad para que en vez de ser un cuadrado tan estricto, tan solo se pintaran los píxeles en un radio, formado así una circunferencia. El problema fue que el área de la capa de visibilidad ya era demasiado pequeña como para reducirla mediante este método aún más.

Desde un principio, el objetivo principal de este proyecto fue el de obtener la capa de visibilidad desde un punto, pero una vez obtenido esto, era muy sencillo aplicarlo a los marcadores, por lo que se hizo una ampliación de lo inicialmente perseguido.

Como posibles alternativas de uso de este proyecto, podría ser para la colocación de las torres de vigilancia de los guardas forestales. Con esto y basándose un poco en las Atalayas se podría calcular las mejores zonas de visión a la hora de construir una nueva torre. Esta aplicación supondría un ahorro de estos cálculos, ya que no sería necesario desplazarse hasta los diferentes lugares para estudiar cada uno de ellos.

Como conclusión personal, este proyecto me ha ayudado a conocer la cantidad de Atalayas y Castillos de nuestra provincia, que atesora la mayor concentración de fortalezas y castillos de toda Europa, a darle importancia a la visibilidad de las zonas, la cual desconocía totalmente.

Además, nunca había utilizado el lenguaje JavaScript, así como ninguna API, lo que me ha ayudado a fortalecer mi formación a cerca de ambos.

Índice de Ilustraciones

Ilustración 1- Torre de la Nava	5
Ilustración 2 - Torre de la Aldehuela	6
Ilustración 3 - Interior Atalaya	7
Ilustración 4 - Visibilidad obtenida mediante el TFG de José Morón.....	8
Ilustración 5 -- Visibilidad obtenida mediante el TFG de José Morón. Generación del mapa de bits	9
Ilustración 6 - - Visibilidad obtenida mediante el TFG de José Morón.....	9
Ilustración 7 -Diagrama de Grant.....	15
Ilustración 8 - Diagrama Pert 1	16
Ilustración 9 - Diagrama Pert 2	17
Ilustración 10 - Openlayers.....	25
Ilustración 11 - Google Maps API	25
Ilustración 12 - OpenstreetMap	26
Ilustración 13 - Bing Maps	26
Ilustración 14 - MapQuest	27
Ilustración 15 - Apple Maps	27
Ilustración 16 - Diagrama de Clases.....	29
Ilustración 17 - Diagramas de casos de uso	31
Ilustración 18 - Diagrama de Actividad	32
Ilustración 19 - Modelo Vista Controlador.....	33
Ilustración 20 - Diagrama de Secuencia Carga del mapa	35
Ilustración 21 - Diagrama de Secuencia Generar capa de visibilidad en un punto.....	36
Ilustración 22 - Diagrama de Secuencia Crear nuevo marcador.....	37
Ilustración 23 - Diagrama de Secuencia Guardar marcador	38
Ilustración 24 - Diagrama de Secuencia Calcular la capa de visibilidad desde un marcador no almacenado.....	39
Ilustración 25 - Diagrama de Secuencia Calcular la visibilidad de los marcadores desde uno no almacenado.....	40
Ilustración 26 - Diagrama de Secuencia Eliminar marcador no almacenado	41
Ilustración 27 - Diagrama de Secuencia Abrir infowindow marcador	42
Ilustración 28 - Diagrama de Secuencia Calcular la visibilidad de un marcador almacenado	43
Ilustración 29 - Diagrama de Secuencia Calcular la capa de visibilidad desde un marcador almacenado.....	44
Ilustración 30 – Diagrama de Secuencia Eliminar marcador.....	45
Ilustración 31 - Storyboard Mapa principal.....	46
Ilustración 32 - Storyboard Capa de Visibilidad	47
Ilustración 33 - Storyboard Nuevo Marcador	47
Ilustración 34 - Storyboard Infowindow	48
Ilustración 35 - Storyboard Visibilidad.....	48
Ilustración 36 - Storyboard Visibilidad POIs.....	49
Ilustración 37 - WAMP Server	50
Ilustración 38 - Apache.....	50
Ilustración 39 - PHP	50
Ilustración 40 - MySQL.....	50
Ilustración 41 - Google Maps JavaScript API.....	51

Ilustración 42 - Google Maps Standard Usage Limits	51
Ilustración 43 - Google Maps Premium Usage Limits	52
Ilustración 44 - Google Maps API Console	52
Ilustración 45 - Google Maps Key.....	52
Ilustración 46 - Nuevo Mapa.....	53
Ilustración 47 - Google Maps getElevationAlongPath	54
Ilustración 48 - Nuevo Marcador.....	55
Ilustración 49 - Ejemplo Marcadores	55
Ilustración 50 - USGSOverlay.....	56
Ilustración 51 - Visibilidad.....	56
Ilustración 52 - Longitud y Latitud Referencia.....	57
Ilustración 53 – Pseudocódigo Cálculo Coordenadas.....	58
Ilustración 54 - Visibilidad Punto de Visión Inferior	59
Ilustración 55 - Visibilidad Punto Visión Superior.....	60
Ilustración 56 - Pseudocódigo Calcular Visibilidad.....	60
Ilustración 57 - Visibilidad POIs	61
Ilustración 58 - Instalación Windows.....	65
Ilustración 59 - Descarga WAMP.....	66
Ilustración 60 - Instalación WAMP	67
Ilustración 61 - WAMP Server	67
Ilustración 62 - Carpeta Web	68
Ilustración 63 - Base de Datos.....	68
Ilustración 64 - Usuario Base de Datos	69
Ilustración 65 - Carga Base De Datos	69
Ilustración 66 - Datos Base de Datos	70
Ilustración 67 - Aplicación.....	70

Índice de Tablas

Tabla 1 - Subproblemas	13
Tabla 2 - Planificación	14
Tabla 3 - Software	18
Tabla 4 - Gasto Hardware	20
Tabla 5 – Tabla de costes de desarrollo del proyecto.....	20
Tabla 6 – Coste Total del proyecto.....	21
Tabla 7 - Requisitos MySQL.....	23
Tabla 8 - Requisitos Hardware	24
Tabla 9 - Requisitos Software.....	24

Bibliografía

Google Maps API Disponible en: <https://developers.google.com/maps/?hl=es-419>

[Consulta: 7-10-2016]

OpenLayers Disponible en: <https://openlayers.org/>

[Consulta: 6-10-2016]

StackOverflow Disponible en: <http://es.stackoverflow.com/>

[Consulta: 3-12-2016]

GeoDataSource Disponible en: <http://www.geodatasource.com/developers/javascript>

[Consulta: 15-10-2016]

Ajax y JavaScript Disponible en:

http://www.aprenderaprogramar.es/index.php?option=com_content&view=article&id=959:ajax-y-javascript-con-consulta-a-base-de-datos-recuperar-informacion-en-tiempo-real-desde-servidor-cu01216f&catid=83:tutorial-basico-programador-web-ajax-desde-cero&Itemid=212

[Consulta: 25-12-2016]

Atalayas o torres de vigilancia Disponible en: <http://www.alcaudete.es/index.php/es/2015-06-29-09-43-29/atalayas-o-torres-de-vigilancia>

[Consulta: 28-12-2016]

Marcadores Google Maps Disponible en: <https://www.sanwebe.com/2013/10/google-map-v3-editing-saving-marker-in-database>

[Consulta: 20-12-2016]

Servidor WAMP Disponible en: <http://www.wampserver.com/en/>

[Consulta: 14-10-2016]

Promesa JavaScript Disponible en:

https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Objetos_globales/Promesa

[Consulta: 21-11-2016]

Sistema de Visibilidad de Andalucía Disponible en:

http://www.juntadeandalucia.es/medioambiente/portal_web/web/temas_ambientales/paisaje/percepcion_paisajes/sva_mapas_suelo2.pdf

[Consulta: 5-1-2017]

Torres y Atalayas de la Sierra Sur de Jaén Disponible en:

http://www.adsur.es/contenidos/baul/turismo/turismo_torres_atalayas.php

[Consulta: 24-12-2016]

JQuery Ajax Disponible en: <http://api.jquery.com/jquery.ajax/>

[Consulta: 18-12-2016]

PHP Disponible en: <http://php.net/manual/es/index.php>

[Consulta: 17-12-2016]

Introduction to Google Maps API Featuring ES6 Promises

Disponible en: <http://jamesbedont.com/2016/03/23/GoogleMapsApiIntroduction.html>

[Consulta: 21-11-2016]

Adding a Custom Overlay

Disponible en:

<https://developers.google.com/maps/documentation/javascript/examples/overlay-simple>

[Consulta: 12-12-2016]

JavaScript Disponible en: <http://www.w3schools.com/js/>

[Consulta: 13-10-2016]

Modelo Digital de Elevaciones Disponible en:
<https://www.ign.es/ign/layoutIn/modeloDigitalTerreno.do>

[Consulta: 12-10-2016]

Creando y utilizando callbacks Disponible en: <https://fernetjs.com/2011/12/creando-y-utilizando-callbacks/>

[Consulta: 3-11-2016]

Herramienta móvil sobre arqueología defensiva en 4D: diseño conceptual,

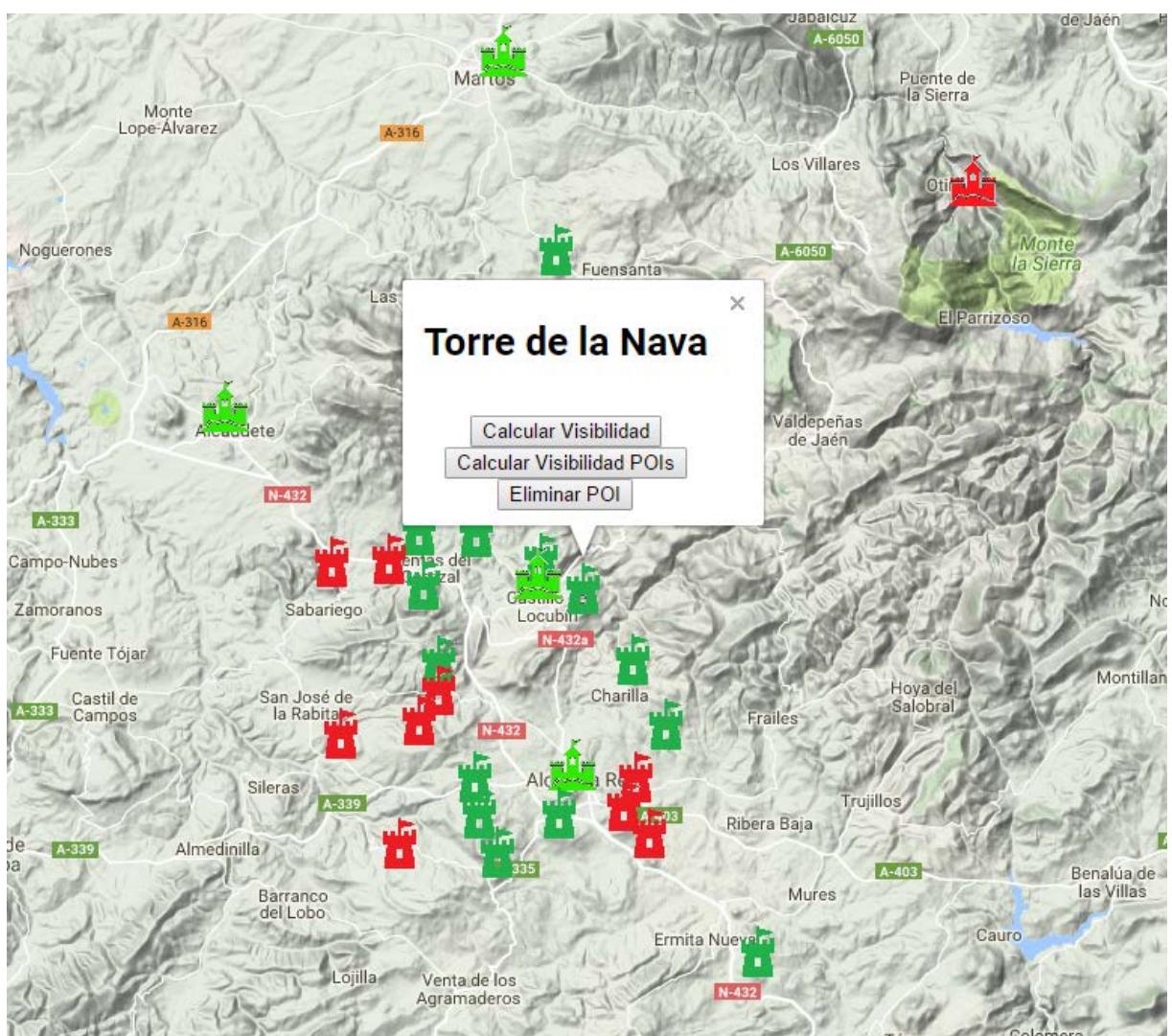
Autores: José M. Noguera, Rafael J. Segura, María V. Gutiérrez, Juan C. Castillo,

Universidad de Jaén, 2015

Anexo 1. Comprobación de los datos obtenidos mediante la aplicación.

Para la comprobación de los resultados obtenidos mediante la aplicación se decidió desplazarse físicamente hasta algunas de las Atalayas y evaluar los datos obtenidos mediante la visibilidad de los diferentes marcadores.

Para esto, se eligió la Torre de la Nava, ya que era de las Atalayas con mayor visibilidad en cuanto a otras Atalayas cercanas.

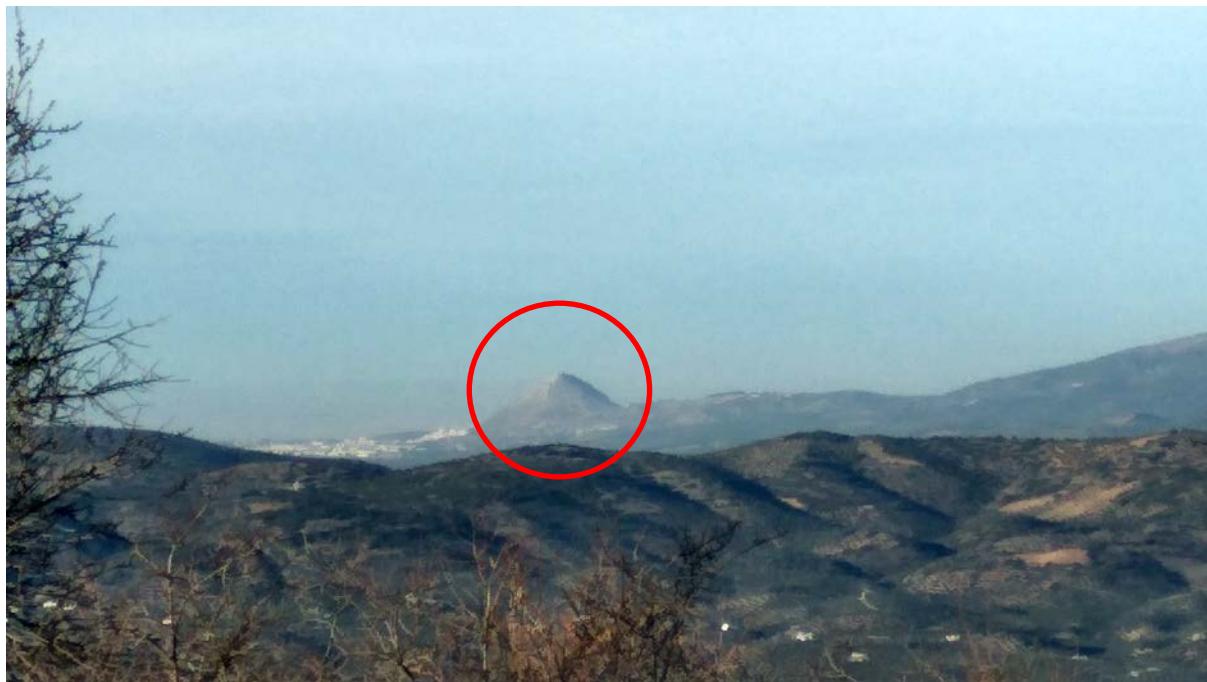


Si lanzamos la aplicación, obtenemos que desde esta Atalaya podemos ver las siguientes:

- Castillo de Alcaudete
- Torre de Encina Hermosa
- Torre de la Cogolla
- Castillo de la Villeta
- Torre de Triana
- Fortaleza de la Mota
- Torre de la Sierra
- Torre de los Pedregales
- Torre de la Solana
- Torre de las Mimbres
- Torre de Guadalquita
- Torre de la Dehesilla
- Torre del Algarrobo
- Torre de Charilla
- Torre de la Boca de Charilla
- Torre del Quejigal
- Castillo de la Peña de Martos

Una vez en la Atalaya se fotografiaron algunas de estas torres, ya que algunas no eran visibles debido a que las fotos fueron tomadas desde la base de la torre y otras no eran visibles a causa de árboles o algunas construcciones.

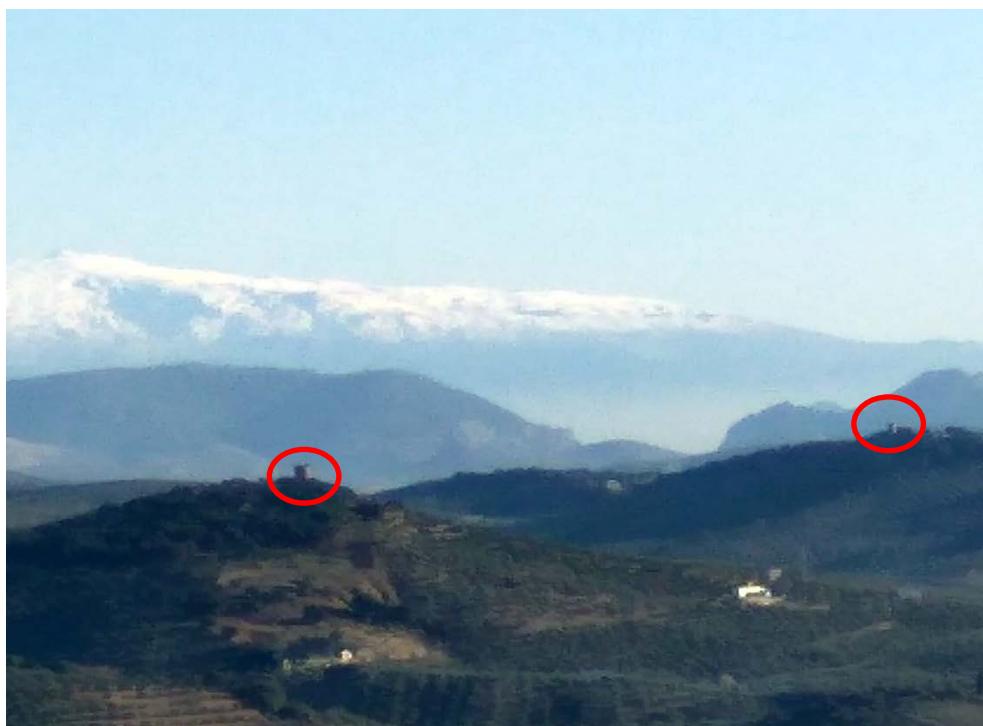
- Castillo de la Peña de Martos



- Fortaleza de la Mota y Torre de la Dehesilla



- Torre de Charilla y Torre de la Boca de Charilla



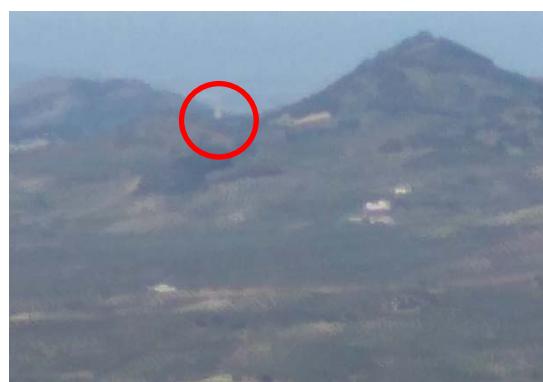
- Torre de Encina Hermosa



- Torre de los Pedregales

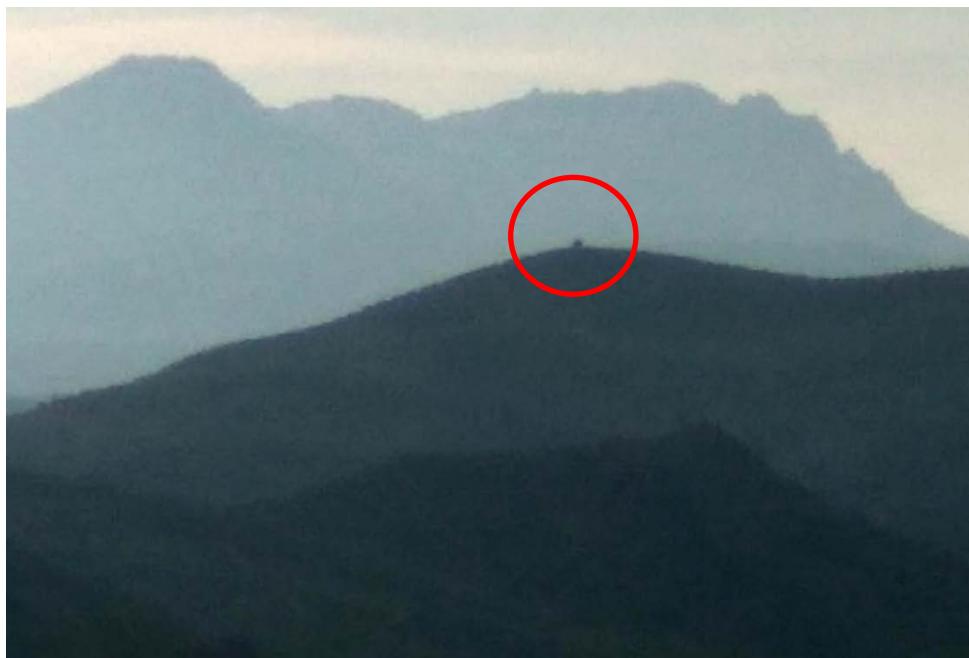


- Castillo de Alcaudete

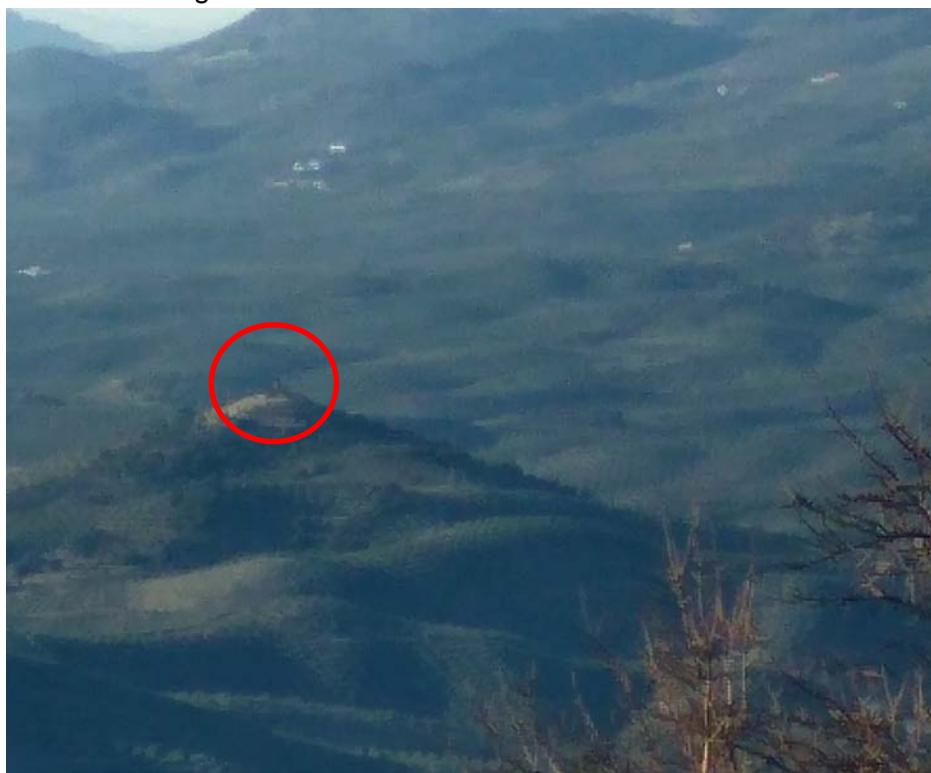


Del Castillo de Alcaudete tan sólo vemos la punta de la torre.

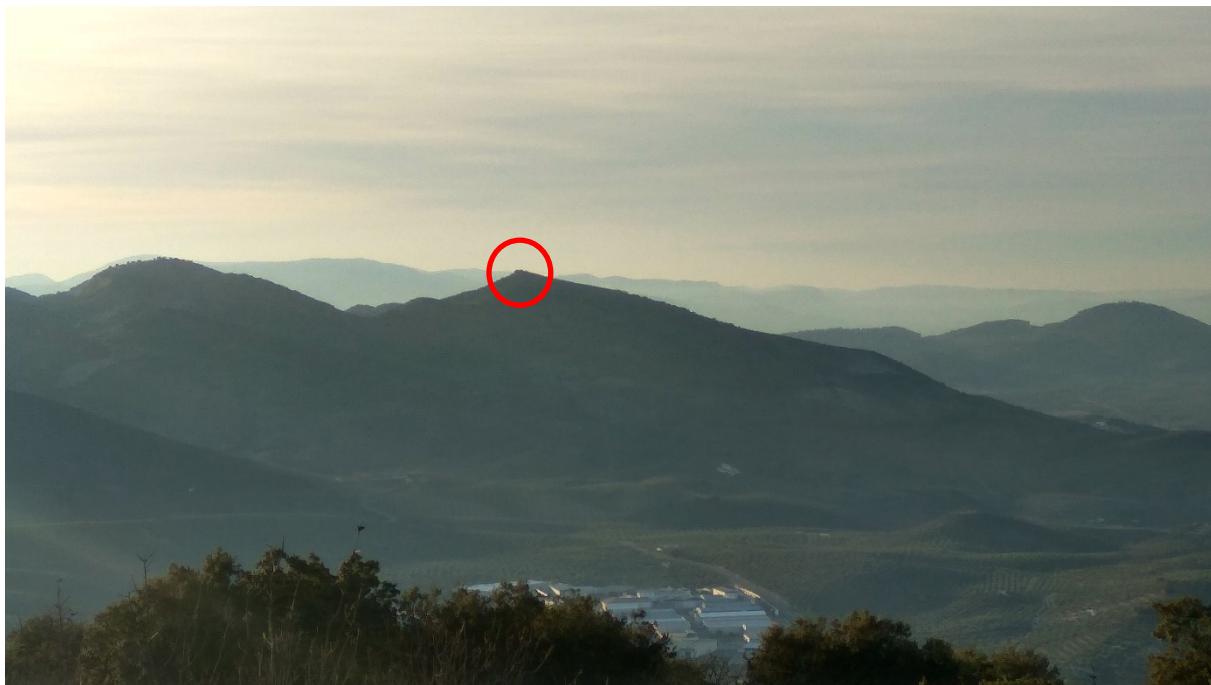
- Torre de las Mimbres



- Torre de la Cogolla



- Torre de la Sierra



La torre de la Sierra no es posible su visión, aparte de que esta casi destruida, hay bastantes árboles que dificultan su visión.

- Torre del Algarrobo



- Torre de Triana



- Castillo de la Villeta

