Universidad de Valladolid

E.U. DE INFORMÁTICA (SEGOVIA)

Grado en Ingeniería Informática de Servicios y Aplicaciones

Aplicación Android para un recorrido monumental virtual por Segovia

Alumno: Ángel Barroso Sanz

Tutor: Jesús Cordobés Puertas

Tutor: Diego Martín de Andrés

Índice de contenido

Índice de tablas

Índice de Figuras

CAPÍTULO 1

PLANTEAMIENTO INICIAL

1.1 Introducción

"Viajamos para cambiar, no de lugar, sino de ideas"

Hippolyte Taine(1828-1893) Filósofo francés.

Segovia está situada en un punto estratégico de nuestra geografía, hecho que ha provocado que prácticamente todas las culturas que han habitado la Península Ibérica desde los albores de la Humanidad hayan dejado su particular impronta en la ciudad y la provincia. Los primitivos habitantes celtas de Segovia dejaron paso a la ocupación romana, cuyas edificaciones civiles, como el célebre Acueducto, se conservan intactas para admiración de los turistas nacionales e internacionales. Pero en Segovia no sólo podemos encontrar testimonios de los antiguos habitantes; la ciudad también posee un magnífico y recoleto casco antiguo por el que poder pasear, tapear y contemplar edificios tan impresionantes como la Catedral.

Además de su vasta oferta histórica, artística y cultural, Segovia es la capital perfecta para disfrutar de las delicias de la mejor cocina. Sus asados son antológicos y tienen fama mundial, gracias a personajes tan conocidos como el famoso Cándido y sus cochinillos. La profusión de restaurantes, asadores y mesones que encontramos por sus calles y plazas ofrecen deliciosas especialidades a precios más que asequibles, lo que supone el complemento perfecto para cualquier jornada turística por la ciudad.

Segovia cuenta además con la ventaja de encontrarse muy cerca de Madrid, la capital de España y urbe cosmopolita por excelencia. Las comunicaciones con la ciudad son magníficas y acercarse a ella para pasar un día de turismo es tan sencillo como recomendable.

Pero, ¿qué pretende este Trabajo de Fin de Grado? Pues pretende que cualquier persona que visite la ciudad de Segovia tenga acceso a toda la información citada anteriormente basándose en la ubicación de su dispositivo móvil.

1.2 Motivación

La geolocalización consiste en conocer automáticamente la ubicación geográfica en la que nos encontramos. Una funcionalidad que va cobrando cada vez más importancia, desde que allá por 2009 se acuñara este término, utilizada sobre todo en el mundo móvil y en el desarrollo de aplicaciones móviles.

Es por esta importancia en auge que cada vez más *smartphones*, y no sólo los de gama alta, incluyen un GPS para poder determinar la posición localizada sobre el mapa. Precisamente es el sector del desarrollo de aplicaciones móviles el que está sabiendo ver la amplia variedad de posibilidades que ofrece la geolocalización. De hecho, hay muchas maneras en las que puede ayudar esta funcionalidad desde cualquier dispositivo móvil.

Por ejemplo, utilizando los datos de localización del dispositivo móvil, se pueden encontrar comercios cercanos, cafeterías, cines, ferreterías, etc, o algo tan simple como establecer la franja horaria en la que el usuario se encuentra simplemente obteniendo latitud y longitud mediante el GPS, información que obtiene de los diversos satélites que orbitan la Tierra.

Aunque también puede recibir información a través del *wi-fi* y *bluetooth* además del *GPS*[1], la localización final es una combinación de los datos obtenidos con esas herramientas. Es por eso que la ubicación puede que no siempre sea exacta, sino que será una ubicación aproximada.

¿Cómo aplicamos esto a nuestro proyecto? Dado que Segovia no posee ninguna aplicación que sirva de guía a un turista, con este trabajo se pretende que un usuario, en función de la localización en la que se encuentre, obtenga información sobre los monumentos y lugares de interés cercanos a su posición, mostrandoles una ficha detallada con información sobre ese lugar en concreto.

Esta posibilidad se encuentra en la aplicación que se presenta, la cual está dirigida a cualquier persona que disponga de un dispositivo móvil inteligente y quiera conocer más a fondo la ciudad de Segovia.

[1] GPS: Global Positioning System (Sistema de Posicionamiento Global)

1.3 Objetivos y alcance del sistema

1.3.1 Alcance

Esta aplicación está dirigida principalmente a toda persona que quiera conocer más la ciudad de Segovia. Se podrá acceder a esta aplicación a través de un dispositivo móvil Android desde la versión 4.1 (API 16 - *Jelly Bean*) hasta la versión más actual, la versión 6.0 (API 23 - *Marshmallow*). Además se puede tener completa disponibilidad de la aplicación sin la necesidad de estar conectado a Internet.

1.3.2 Objetivos

- **OB-01 :** Mejorar la sensación del turista que visite la ciudad de Segovia.
- **OB-02 :** Hacer accesible la información sobre la ciudad de Segovia y sus monumentos a través del móvil en función de la localización.
- **OB-03 :** Posicionar al turista en todo momento para que sepa en qué punto de la ciudad se encuentra exactamente.

1.3.3 Reglas de Negocio

- **RN-01 :** El usuario deberá disponer de una conexión a internet.
- **RN-02 :** El usuario sólo recibirá una notificación de monumento cercano por cada arranque de la aplicación.
- **RN-03 :** El usuario podrá acceder a la información del monumento independientemente de su posición o de su conexión a internet.

1.4 ¿Por qué Apache Cordova?

Apache Cordova es un *framework* de licencia libre que cuenta con muchas Apis de diversos dispositivos móviles para desarrollar aplicaciones nativas dentro de un *smartphone*. Cada vez está tomando más énfasis en el mundo de los programadores, ya que para el desarrollo de las aplicaciones se utilizan las tecnologías web como HTML, CSS y JavaScript, resultando aplicaciones híbridas.

Una de las grandes peculiaridades de este entorno de trabajo es la posibilidad de desarrollar aplicaciones móviles para iOS, Android y demás sistemas operativos sin la necesidad de programar en sus lenguajes nativos (Java, Objetive-C, etc.), dado que las aplicaciones programadas en Cordova comparten la misma interfaz mientras que a más bajo nivel sigue estando el código nativo de cada sistema operativo, de ahí que se consideren híbridas. Cordova proporciona, además, una serie de bibliotecas de código abierto que permiten la comunicación entre esta interfaz común y las distintas funciones nativas del dispositivo.

Las aplicaciones Cordova se basan en un archivo común config.xml , que proporciona información acerca de la aplicación y especifican los parámetros que afectan a su funcionamiento. Este archivo se adhiere a la especificación de *Empaquetado de aplicaciones Web* o *Widget*, de la W3C[2].

La misma aplicación se implementa como una página web, que contiene todos los archivos HTML, CSS y JavaScript para que se ejecute. Cordova genera un WebView[3] dentro de la envoltura de una aplicación nativa a través del SDK de la plataforma deseada, que proporciona una aplicación con interfaz de usuario completa y que interactúa con varias características del dispositivo de forma nativa. Para ello, Cordova dispone de una amplia gama de plugins para que la aplicación tenga acceso a los distintos componentes.

Gracias a las herramientas de este *framework* se pueden desarrollar aplicaciones móviles para las siguientes plataformas:

[2] W3C: son las siglas de World Wide Web Consortium, un consorcio fundado en 1994 para dirigir a la Web hacia su pleno potencial mediante el desarrollo de protocolos comunes que promuevan su evolución y aseguren su interoperabilidad.
[3] WebView: es un navegador integrado dentro de una aplicación.

- Amazon Fire OS.
- Android.
- BlackBerry 10.
- Firefox OS.
- iOS.
- Ubuntu.
- Windows Phone.
- Windows 8.
- Tizen.

1.4.1 Razones para su elección

En definitiva, ¿por qué desarrollar una aplicación Android basada en Cordova? Además de porque Android es el sistema operativo más extendido entre los usuarios de dispositivos móviles, apache Cordova nos permite desarrollar la aplicación en un tiempo bastante menor de lo que supondría desarrollar esa misma aplicación en lenguaje Android nativo. Y también porque en un momento dado, si la aplicación en Android tiene éxito, sería mucho menos costoso migrar la misma aplicación a los diferentes sistemas operativos móviles, ya que con Cordova, una sola programación de la aplicación permite disfrutar de la aplicación en todos los sistemas operativos mencionados anteriormente.

En las figuras 1.1 y 1.2, se muestran una serie de gráficos indicando el porcentaje de uso (en el año 2015) de todos los sistemas operativos para dispositivos móviles:

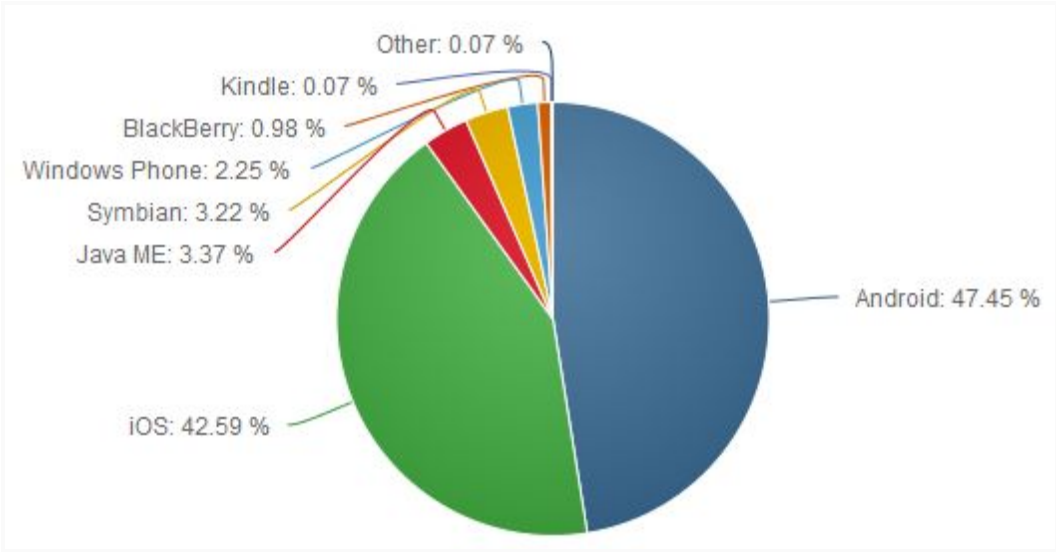


Figura 1.1 Sistemas operativos móviles más utilizados en 2015.

Además el rango de versiones elegido para que esté disponible la aplicación es usado por más del 91% de los dispositivos Android actuales.

Version	Codename	API	Distribution
2.2	Froyo	8	0.2%
2.3.3 - 2.3.7	Gingerbread	10	3.0%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	2.7%
4.1.x	Jelly Bean	16	9.0%
4.2.x		17	12.2%
4.3		18	3.5%
4.4	KitKat	19	36.1%
5.0	Lollipop	21	16.9%
5.1		22	15.7%
6.0	Marshmallow	23	0.7%

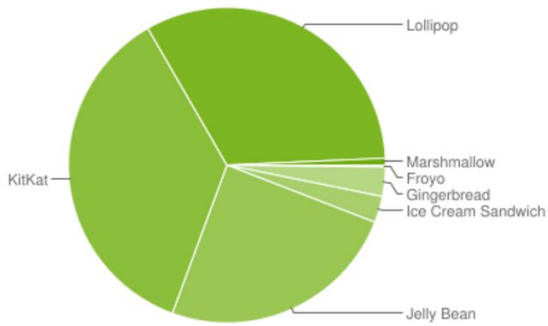


Figura 1.2 Versiones Android más utilizadas a Enero de 2016.

1.4.2 Ventajas de una aplicación híbrida

- Minimizamos el código específico: La mayor parte del código puede utilizarse para el resto de plataformas. Solo se utiliza código nativo para aquellos aspectos que lo requieran.
- Menor coste de desarrollo sobre todo si se requiere la aplicación en varias plataformas.
- Menor coste de mantenimiento al ser la mayor parte del código común a todas las plataformas.
- Una aplicación Híbrida puede acceder a los recursos del dispositivo móvil prácticamente como una nativa.
- Se distribuye mediante los respectivos “stores”.

1.4.3 Desventajas de una aplicación híbrida

- Rendimiento: El rendimiento y la experiencia de usuario no pueden alcanzar los niveles de la aplicación nativa.
- Habitualmente los procesos de aprobación en los correspondientes "stores" para aplicaciones híbridas son más estrictos y pueden llegar a ser rechazadas si no queda clara la funcionalidad proporcionada mediante la carga de código remoto.
- El diseño visual se corresponde más con una aplicación web que con el del sistema operativo en el que se muestra.

1.5· Estado del arte

En los últimos años ha surgido un gran número de aplicaciones comparables a la desarrollada en este proyecto que, sin embargo, ya sea por su funcionalidad, por su estética o por su abandono por parte de los desarrolladores no cumplen las expectativas. Las siguientes apps que se muestran se pueden conseguir en el Android Market.

- TourKhana Segovia

Figura 1.3 Logo aplicación Tourkhana.

Tiene por protagonistas a dos cochinillos y se sirve de la realidad aumentada para mostrar información de la ciudad, retos, recompensas, pasatiempos y juegos, todos ellos relacionados con Segovia. Invita a descubrir el gran secreto de la ciudad, el "código gorrinchi". Pig y Pog, los cochinillos, cuentan la historia de la ciudad para que el usuario pueda jugar con ellos y descubrir el secreto. Si se pulsa en "Ayuda", la aplicación dirá al usuario lo que tiene que hacer: hay que partir del Acueducto y completar un recorrido que termina en la Iglesia de la Vera Cruz. En cada parada se ve un vídeo y se realiza una prueba; todas ellas juntas y superadas desvelarán el secreto… Para poder disfrutar de la realidad aumentada el usuario debe tener cargado el Layar[4].Ambas aplicaciones se pueden descargar gratuitamente de la tienda virtual de Android o Apple.

[4] Layar: aplicación móvil que sirve para dotar de realidad aumentada e interactividad a cualquier material impreso.

- App Segovia Guía Segovia

Figura 1.4 App Segovia Guía Segovia

Esta aplicación incluye información sobre promociones, eventos, agenda cultural, ocio, restauración y tiendas en la ciudad de Segovia de una forma fácil e intuitiva. En resumen, no sólo proporciona información sobre monumentos, sino que se centra en toda la oferta de la ciudad. Tiene bastante buena opinión por parte de los usuarios, pero no tiene suficientes descargas, no es fácil encontrar información sobre ella y sobre todo lleva sin actualizarse desde enero de 2014.

- Segovia para todos

Figura 1.5 Aplicación Segovia para todos.

La app "Segovia para Todos" es una audioguía interactiva con cuatro recorridos temáticos, una ruta con información sobre los monumentos accesibles para personas con movilidad reducida y cuatro opciones para descargar o disfrutar online de todo el contenido de la guía. Cuenta con mala opinión por parte de los usuarios y además también está desactualizada.

- Segovia Guía de Turismo - S2go

Figura 1.6 Aplicación S2go.

Sights 2Go Segovia es la guía turística más completa de la ciudad de Segovia que se puede llevar en su teléfono o tablet. Esta aplicación de turismo le ayuda a descubrir los más importantes monumentos, museos, hoteles, restaurantes y otras importantes atracciones de la ciudad, como el Acueducto de Segovia, El Alcázar, La Catedral, etc.

Incluye un mapa detallado de la ciudad y de la provincia, donde poder visualizar, por medio de GPS, donde se encuentra el visitante en cada momento, así como su posición respecto a las distintos puntos claves de la ciudad. Funciona totalmente *offline*, sin necesidad de ninguna conexión de datos.

CAPÍTULO 2

PLANIFICACIÓN Y PRESUPUESTO

2.1 Planificación

A la hora de llevar a cabo correctamente la planificación de un proyecto se debe tener en cuenta la metodología que se va a utilizar, el tiempo disponible y los recursos asignados, los objetivos y tener también presente las restricciones de presupuesto.

2.2 Metodología

Rompiendo con los esquemas tradicionales, he aplicado la metodología de trabajo *Scrum* para la realización del proyecto.

Scrum es un proceso en el que se aplican de manera regular un conjunto de buenas prácticas para trabajar colaborativamente, en equipo, y obtener el mejor resultado posible de un proyecto.

En *Scrum* se realizan entregas parciales y regulares del producto final, priorizadas por el beneficio que aportan al receptor del proyecto. Por ello, *Scrum* está especialmente indicado para proyectos en entornos complejos, donde se necesita obtener resultados pronto, donde los requisitos son cambiantes o poco definidos, donde la innovación, la competitividad, la flexibilidad y la productividad son fundamentales. Los principales beneficios que proporciona *Scrum* son:

- *Cumplimento de expectativas*: El cliente establece sus expectativas indicando el valor que le aporta cada requisito / historia del proyecto. El equipo los estima y con esta información el *Product Owner*[5] (tutor) establece su prioridad.
- *Flexibilidad a cambios*: Alta capacidad de reacción ante los cambios de requerimientos generados por necesidades del cliente o evoluciones del mercado. La metodología está diseñada para adaptarse a los cambios de requerimientos que conllevan los proyectos complejos.

[5] Product owner: rol que representa al cliente.

- *Reducción del tiempo de venta*: El cliente puede empezar a utilizar las funcionalidades más importantes del proyecto antes de que esté finalizado por completo.
- *Mayor calidad del software*: El método de trabajo y la necesidad de obtener una versión funcional después de cada iteración, ayuda a la obtención de un software de calidad superior.
- *Mayor productividad*: Se consigue entre otras razones, gracias a la eliminación de la burocracia y a la motivación del equipo que proporciona el hecho de que sean autónomos para organizarse.
- *Maximiza el retorno de la inversión (ROI[6])*: Producción de software únicamente con las prestaciones que aportan mayor valor de negocio gracias a la priorización por retorno de inversión.
- *Predicciones de tiempos*: Mediante esta metodología se conoce la velocidad media del equipo por *sprint* (los llamados puntos historia), con lo que consecuentemente, es posible estimar fácilmente para cuándo se dispondrá de una determinada funcionalidad que todavía está pendiente de realizar.
- *Reducción de riesgos*: El hecho de llevar a cabo las funcionalidades de más valor en primer lugar y de conocer la velocidad con que el equipo avanza en el proyecto, permite despejar riesgos eficazmente de manera anticipada.

Para ello, dividimos el desarrollo de la aplicación en diferentes fases a las que llamaremos "sprint" y que tendrán una media de 15 días de duración . En cada *sprint*, se marcan como objetivos una serie de tareas a realizar. Además, cada sprint cuenta con 4 fases propias de modelo de desarrollo incremental, que son :

- Análisis: es el proceso en el que se intenta descubrir qué es lo que realmente se necesita y se llega a una comprensión adecuada de los requerimientos del sistema (las características que el sistema debe poseer).

[6] ROI : El retorno sobre la inversión (RSI o ROI, por las siglas en inglés de *return on investment*) es una razón financiera que compara el beneficio o la utilidad obtenida en relación a la inversión realizada

- Diseño: fase en la que se describen requisitos de la arquitectura de la aplicación y la definición precisa de cada subconjunto de la aplicación, como la estructura del software, los datos y las interfaces entre componentes.
- Implementación: es la fase que se realiza una vez se conoce qué funciones debe desempeñar el sistema de información (análisis) y se ha decidido cómo se van a organizar sus distintos componentes (diseño). En ella se procede a desarrollar el sistema diseñado mediante el uso de las herramientas adecuadas.
- Pruebas: se prueba cada subconjunto de la aplicación creado para garantizar que el desarrollo ha sido realizado de acuerdo con las especificaciones.
- Despliegue: la etapa de despliegue implica instalación, personalización, y testeo.

Por tanto, cada tarea del *sprint* tendrá a su vez todas esas tareas asociadas. Tras finalizar el plazo de entrega de cada tarea, se realiza una retrospectiva, es decir, se comprueba que las tareas realizadas cumplen los requisitos estipulados y que cubren las necesidades de la aplicación. En caso de que esos requisitos no se cumplan, se volverán a definir.

A continuación, dejamos una estimación de la planificación de tareas y tiempo sobre el desarrollo de la aplicación :

Sprint 1		**15 días**
	Crear estructura de la aplicación	2 días
	Implementar estructura de la aplicación	5 días
	Introducir Api de Google Maps	3 días
	Definir área circular para un monumento	5 días
Sprint 2		**12 días**
	Detectar posición mientras el usuario se mueve	4 días
	Definir intervalo de tiempo de ejecución del script localizador	1 día
	Comprobar si la posición está dentro de un área	7 días
Sprint 3		**15 días**
	Enviar notificación cuando la posición esté dentro de un área	5 días
	Inserción de datos de todos los monumentos	5 días
	Generar script de localización para todos los monumentos	5 días
Sprint 4		**15 días**
	Testeo del correcto funcionamiento de la app	8 días
	Corrección de errores	3 días
	Optimización de código	4 días

Tabla 2.1 Organización de tareas a realizar

En total salen **57 días** de desarrollo de la aplicación. Teniendo en cuenta que cada día tiene 8 horas laborables, el desarrollo de la aplicación supone un total de **456 horas** de desarrollo.

Sin embargo, con este modelo planteado no es posible obtener una aplicación con una buena escalabilidad, y los costes de mantenimiento serían demasiado elevados, por lo que al modelo planteado se le añade un *sprint* más:

Sprint 5		**11 días**
	Creación de una base de datos externa	1 días
	Creación de una API Rest	4 días
	Enlazar aplicación con API Rest externa	4 días
	Testeo del buen funcionamiento de la app	2 días

Tabla 2.2 Sprint extra

Esto supone que a los 57 días anteriormente estimados habría que añadir otros 11 días, obteniendo un total de **68 días**, con el correspondiente incremento de horas que ello supone, siendo un total de **544 horas**.

2.3 Presupuesto

Las siguientes estimaciones se realizan a través de varios métodos con el objetivo de acotar los límites entre los que se va a encontrar el coste final y que se acerquen lo máximo al resultado final.

2.3.1 Estimación de costes

Para desarrollar el proyecto se necesitarán medios de Hardware y Software, cuyo coste proporcional al uso que se le dará en el proyecto hay que introducirlo en el presupuesto. Naturalmente, también se deberá incluir el coste de los recursos humanos utilizados.

Como desarrollador, dado que se van a realizar las tareas de un analista, un diseñador y un programador, se utilizará el perfil de un programador *Fullstack*. Los desarrolladores Fullstack ofrecen el paquete completo. Su gran experiencia les ha dado la capacidad de hacer de Backend[7], Frontend[8], y a veces también DevOps[9] en una misma persona. Es decir, se

[7] Backend: toda la tecnología relacionada con el lado servidor
[8] Frontend: toda la tecnología relacionada con el lado cliente
[9] DevOps: Su objetivo es ayudar a una organización a producir productos y servicios software rápidamente.

encargará tanto del análisis, diseño y desarrollo que conlleva la aplicación, por lo que no nos hace falta desglosar el número de horas que se dedican a cada parte.

- ➔ Un desarrollador Fullstack cobra 26.000 € brutos al año, lo que significa que a la empresa ese trabajador le cuesta 33.800 € al año(incluyendo el pago a la seguridad social). Si dividimos ese salario entre 14 pagas y calculamos el precio por hora de ese trabajador, el coste aproximado por hora nos sale a unos **15,08 € / hora**.
- ➔ La estimación de horas del proyecto es de **544 horas.**

	Tiempo	**Coste**
Desarrollador Fullstack	544 horas	15,08 € / hora
TOTAL	544 horas	**8203,52 €**

Tabla 2.3 Estimación de RRHH

Por otro lado, se debe disponer también de una serie de herramientas para poder realizar el trabajo. A continuación, se muestran dos tablas que indican los costes de *software* y *hardware* del proyecto.

Hardware	**Uso (%)**	**Coste total (€)**	**Coste(€)**
Acer Aspire 771G	7%	600	42
Xiaomi MI3 W	9%	325	29,25
Internet	15%	55	8,25
Impresora	10%	190	19
TOTAL			**98,50 €**

Tabla 2.4 Estimación de costes de hardware

Software	Uso (%)	Coste total (€)	Coste(€)
Atom	10%	0	0,00
Windows 10	9%	0	0,00
Apache Cordova	75%	0	0,00
Google Chrome	5%	0	0,00
TOTAL			**0,00 €**

Tabla 2.5 Estimación de costes de software

También se debe tener en cuenta que para publicar la aplicación en los markets de Google hay que tener una licencia de desarrollador. Esta licencia solo se paga una vez y es de por vida, por lo que se incluye en el presupuesto.

Otros	Uso (%)	Coste total (€)	Coste(€)
Licencia Google	100%	25	25,00
TOTAL			**25,00 €**

Tabla 2.6 Otros gastos

Una vez calculadas todas las estimaciones, para obtener el presupuesto total del proyecto por estimación de costes, basta con sumar los valores obtenidos anteriormente

Estimación del coste del proyecto = Coste estimado RRHH + Coste estimado Hardware + Coste estimado Software + Otros gastos

Estimación del coste del proyecto= 8203,52 € + 98,50 € + 0,00 € + 25,00 €

Estimación del coste del proyecto = 8.327,02 €

2.3.2 Estimación mediante puntos de función

Este método se basa en una métrica que cuantifica la funcionalidad al construir la aplicación. Los parámetros que sirven para evaluar dicha funcionalidad son:

- *Número de entrada*s: Datos que el usuario aporta al sistema (nombres de ficheros, menús de selección).
- *Número de salidas*: Datos que el sistema aporta al usuario (informes, mensajes).
- *Número de ficheros lógicos internos*: Ficheros o bases de datos internos del sistema.
- *Número de ficheros externos*: Ficheros o bases de datos externos al sistema.
- *Número de consultas externas*: Entradas que requieren de una respuesta por parte del sistema.

Para hallar los PFNA[10] se debe contar el número de elementos de cada clase. Cada elemento de cada clase debe ser clasificado según su grado de complejidad (alta, media o baja). Los criterios para evaluar la complejidad de los elementos de cálculo se muestran en la tabla 2.6:

Ficheros lógicos externos e internos				**Salidas y consultas**				**Entradas**			
Registros Elementales	Datos Elementales			Tipos de Ficheros	Datos Elementales			Tipos de Ficheros	Datos Elementales		
	1-9	20-50	>51		1-9	6-19	>20		1-4	5-15	>16
1	Baja	Baja	Media	0-1	Baja	Baja	Media	0-1	Baja	Baja	Media
2-5	Baja	Media	Alta	2-3	Baja	Media	Alta	2-3	Baja	Media	Alta
>6	Media	Alta	Alta	>4	Media	Alta	Alta	>3	Media	Alta	Alta

Tabla 2.7 Grados de complejidad de PF

[10] Puntos de Función No Ajustados

	Complejidad Baja	Complejidad Media	Complejidad Alta	TOTAL
Entradas	1x3	0x4	0x6	3
Salidas	0x4	1x5	1x7	12
Consultas	27x3	1x4	0x6	85
Ficheros Internos	0x7	0x10	0x15	0
Ficheros Externos	1x5	0x7	0x10	5
TOTAL				**105**

Tabla 2.8 Valores de complejidad de PF

A través de las dos tablas anteriores, se obtienen los PFNA, que para este proyecto son **105**. Ahora, a través de la siguiente tabla se consigue el factor de complejidad para obtener los **PFA**[11].

[11] Puntos de Función Ajustados

Factor de Complejidad (FC)	0-5
Comunicación de datos	4
Rendimiento	5
Frecuencia de transacciones	5
Requisitos de manejo del usuario final	1
Procesos complejos	2
Facilidad de mantenimiento	1
Instalación en múltiples lugares	0
Funciones distribuidas	0
Gran carga de trabajo	1
Entrada on-line de datos	0
Actualizaciones on-line	3
Interacción con otros sistemas	3
Facilidad de operación	2
Facilidad de cambio	2

Tabla 2.9 Factores de complejidad

Para obtener el factor de ajuste, se aplica la siguiente fórmula:

$$\mathbf{FA}^{12} = (0{,}01 * \sum FC) + 0{,}65$$

$$\sum FC = 29$$

$$\mathbf{FA} = (0{,}01 * 29) + 0{,}65$$

$$\mathbf{FA = 0{,}94}$$

Una vez obtenido el factor de ajuste, se calculan los **PFA** :

$$\mathbf{PFA = PFNA * FA}$$

[12] Factor de Ajuste

PFA = 105 * 0,94

PFA = **98,7**

Tomando como referencia que se ha generado código, y que el punto de función equivale a 15 líneas de código, se obtiene un total de :

98,7 *15 = **1780 líneas de código (LDC)**

2.3.3 Estimación mediante COCOMO

Para realizar el modelado algorítmico de costes se va a llevar a cabo un COCOMO **Empotrado**, ya que a pesar de que el proyecto tiene requisitos restrictivos y existen presiones de tiempo, tenemos experiencia en el entorno a desarrollar y según las estimaciones anteriores sólo se necesitan 1780 LDC.

Proyecto Software	a	b	c	d
Orgánico	2,40	1,05	2,50	0,38
Semi-acoplado	3,0	1,12	2,5	0,35
Empotrado	3,20	1,20	2,50	0,32

Tabla 2.10 Tipos de COCOMO

Por tanto, primero se calculará el esfuerzo nominal para obtener el número de personas-mes que se necesitan para el proyecto, y para posteriormente calcular el tiempo de desarrollo.

Esfuerzo nominal (E) = $a*(KLDC)^b$

$$\mathbf{E} = 3{,}20*(1{,}78)^{1{,}20}$$

$$\mathbf{E} = 6{,}39$$

Previo a esto, se debe calcular el valor de los factores conductores de coste. Mediante tabla reflejada en la figura 2.1, se estiman los factores de coste necesarios para nuestro proyecto:

Factores Conductores del Coste		Valor de los factores					
		Muy bajo	Bajo	Medio	Alto	Muy alto	Extra
Software	Fiabilidad del software requerido	0,75	0,88	1,00	1,15	1,4	
	Tamaño de la base de datos		0,94	1,00	1,08	1,16	
	Complejidad del software	0,70	0,85	1,00	1,15	1,30	1,65
Hardware	Restricciones de rendimiento en tiempo de ejecución			1,00	1,11	1,30	1,66
	Restricciones de memoria			1,00	1,06	1,21	1,56
	Volatilidad del entorno de la máquina virtual		0,87	1,00	1,15	1,30	
	Tiempo de respuesta requerido		0,87	1,00	1,07	1,15	
Personal	Capacidad de los analistas	1,46	1,19	1,00	0,86	0,71	
	Experiencia con el tipo de aplicación	1,29	1,13	1,00	0,91	0,82	
	Experiencia con el hardware	1,21	1,10	1,00	0,90		
	Experiencia con el lenguaje de programación	1,14	1,07	1,00	0,95		
	Capacidad de los programadores	1,42	1,17	1,00	0,86	0,70	
Proyecto	Técnicas modernas de programación	1,24	1,10	1,00	0,91	0,82	
	Utilización de herramientas software	1,24	1,10	1,00	0,91	0,83	
	Restricciones en la planificación temporal del desarrollo	1,23	1,08	1,00	1,04	1,10	

Figura 2.1 Factores de coste de COCOMO

Conductores de Coste	Valoración
Fiabilidad requerida del software	1,15
Tamaño de la base de datos	0,94
Complejidad del software	1,00
Restricciones del tiempo de ejecución	1,00
Restricciones de memoria	1,00
Volatilidad de la máquina virtual	1,00
Tiempo de respuesta requerido	1,07
Capacidad de los analistas	1,00
Experiencia con el tipo de aplicación	1,00
Experiencia con el hardware	0,90
Experiencia con el lenguaje de programación	0,95
Capacidad de los programadores	0,86
Prácticas de programación modernas	0,91
Utilización de herramientas software	0,91
Restricciones en la planificación temporal del desarrollo	1,04

Tabla 2.11 Conductores de coste COCOMO

El valor de los factores conductores de coste es igual al producto de las estimaciones anteriores, por tanto :

FCC = 0,73

Con este valor calculado, ya se puede calcular el esfuerzo

E = En * FCC

E = 6,39 * 0,73

$$\mathbf{E} = 4{,}66 \text{ personas/mes}$$

Y el tiempo de desarrollo:

$$\mathbf{TD} = c*(E)^{d}$$

$$\mathbf{TD} = 2{,}50 * (4{,}66)^{0{,}32}$$

$$\mathbf{TD} = 4{,}09 \text{ meses}$$

El nº medio de personas es **E**/**TD** = 4,66/4,09 = 1,13 , o lo que es igual, una persona para realizar el trabajo en 4 meses.

Como se explicó en uno de los apartados anteriores, un programador *Fullstack* cobra 1857,14 € netos/ mes , por tanto, el presupuesto estimado para el proyecto mediante COCOMO es de **7.595,71 €.**

2.4 Costes finales

El tiempo real empleado en la realización del proyecto ha sido el siguiente:

Sprint 1		**4 días**
	Crear estructura de la aplicación	0.5 días
	Implementar estructura de la aplicación	0.5 días
	Introducir Api de Google Maps	1 días
	Definir área circular para un monumento	2 días
Sprint 2		**8 días**
	Detectar posición mientras el usuario se mueve	2 días
	Definir intervalo de tiempo de ejecución del script localizador	0.5 días
	Comprobar si la posición está dentro de un área	5.5 días
Sprint 3		**7 días**
	Enviar notificación cuando la posición esté dentro de un área	0.5 días
	Inserción de datos de todos los monumentos	1.5 días
	Generar script de localización para todos los monumentos	3 días
Sprint 4		**9 días**
	Testeo del correcto funcionamiento de la app	3 días
	Corrección de errores	4 días
	Optimización de código	2 días
Sprint 5		**5 días**
	Creación de una base de datos externa	0.5 días
	Creación de una API Rest	0.5 días
	Enlazar aplicación con API Rest externa	2 días
	Testeo del buen funcionamiento de la app	2 días

Tabla 2.12 Tiempo real de ejecución de tareas

Dado que lo único que ha variado son las horas empleadas en la aplicación, utilizado los estándares definidos en el apartado anterior:

- ➔ Un desarrollador Fullstack cobra **15,08 € / hora**.
- ➔ Las horas reales del proyecto son **264 horas.**

	Tiempo	**Coste**
Desarrollador Fullstack	264 horas	15,08 € / hora
TOTAL	264 horas	**3.981,12 €**

Tabla 2.13 Coste real de RRHH

Hardware	**Uso (%)**	**Coste total (€)**	**Coste(€)**
Acer Aspire 771G	7%	600	42
Xiaomi MI3 W	9%	325	29,25
Internet	15%	55	8,25
Impresora	10%	190	19
TOTAL			**98,50 €**

Tabla 2.14 Coste real de hardware

Software	**Uso (%)**	**Coste total (€)**	**Coste(€)**
Atom	10%	0	0,00
Windows 10	9%	0	0,00
Apache Cordova	75%	0	0,00
Google Chrome	5%	0	0,00
TOTAL			**0,00 €**

Tabla 2.15 Coste real de software

Otros	Uso (%)	Coste total (€)	Coste(€)
Licencia Google	100%	25	25,00
TOTAL			**25,00 €**

Tabla 2.16 Coste real de otros gastos

Una vez calculadas todos los costes reales, para obtener el coste real total del proyecto, basta con sumar los valores obtenidos anteriormente

Coste real del proyecto = Coste real RRHH + Coste real Hardware + Coste real Software + Coste real otros gastos

Coste real del proyecto= 3.981,12 € + 98,50 € + 0,00 € + 25,00 €

Coste real del proyecto = 4.104,62 €

CAPÍTULO 3

ANÁLISIS

3.1 Características principales

En este apartado se listan las características principales del software a desarrollar. Para dar con las características del sistema se debe pensar cómo los usuarios lo utilizarán y así, determinar tanto la falta de alguna característica, como la posibilidad de estar considerando características innecesarias.

- **C-01 Gestión de monumentos**
- **C-02 Gestión de mapa**
- **C-03 Gestión de notificaciones**

3.1.1 Árbol de características

El siguiente árbol de características pretende crear una imagen global del funcionamiento del sistema.

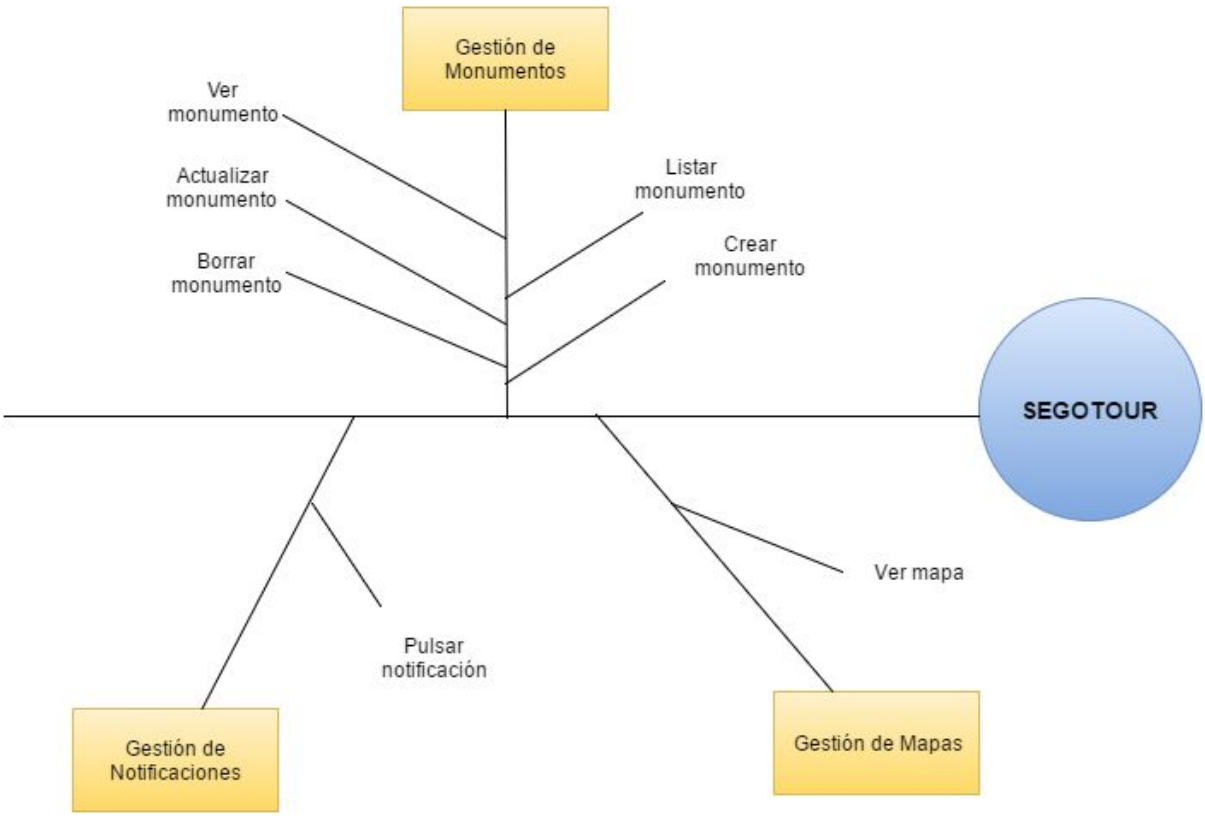


Figura 3.1 Árbol de características.

3.2 Identificación de los actores del sistema

Un actor es una persona u otra entidad externa al sistema software, el cual interactúa con el sistema y realiza casos de uso para efectuar tareas. Diferentes actores a menudo corresponden con diferentes clases de usuario, o roles, identificados a partir de una comunidad de clientes que utilizará el producto.
En este apartado se especificará el nombre del actor/es principal que inicia el caso de uso y cualquier otro actor secundario que participe en la realización de la ejecución del caso de uso.

ACT–01	***Usuario***
Versión	*1.0*
Autores	*Ángel Barroso Sanz*
Descripción	*Usuario que hace uso normal de la aplicación.*
Comentarios	

Tabla 3.1 Tabla de actor 1

ACT–02	***Cliente de Api Rest***
Versión	*1.0*
Autores	*Ángel Barroso Sanz*
Descripción	*Entidad que recibe peticiones y devuelve información.*
Comentarios	

Tabla 3.2 Tabla de actor 2

ACT–03	***Administrador***
Versión	*1.0*
Autores	*Ángel Barroso Sanz*
Descripción	*Persona que se encarga del mantenimiento de la base de datos de monumentos.*
Comentarios	

Tabla 3.3 Tabla de actor 3

3.3 Requisitos de usuario

En esta sección se van a exponer los requisitos de usuario modelandolos en forma de Casos de Uso, viéndose el sistema desde la perspectiva de los usuarios que interactúan con él.

3.3.1 Listado de casos de uso

Dividiremos los casos de uso por actores:

- ➔ **Actor 01 - Usuario**
  - **UC-01** Ver listado monumentos
  - **UC-02** Ver información de monumento
  - **UC-03** Ver mapa
  - **UC-04** Pulsar notificación
- ➔ **Actor 02 - Cliente API Rest**
  - **UC-05** Obtener JSON Listado Monumentos
  - **UC-06** Obtener JSON Monumento
- ➔ **Actor 03 - Admin**
  - **UC-07** Login
  - **UC-08** Crear monumento
  - **UC-09** Actualizar monumento
  - **UC-10** Borrar monumento

3.3.2 Diagrama de casos de uso

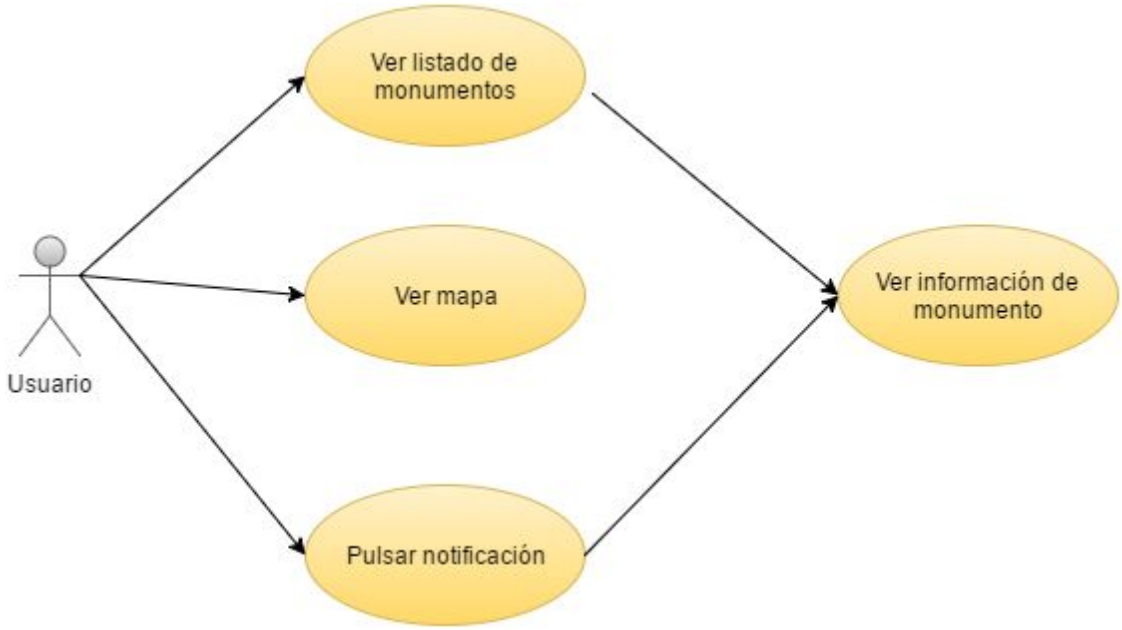


Figura 3.2 Caso de uso usuario

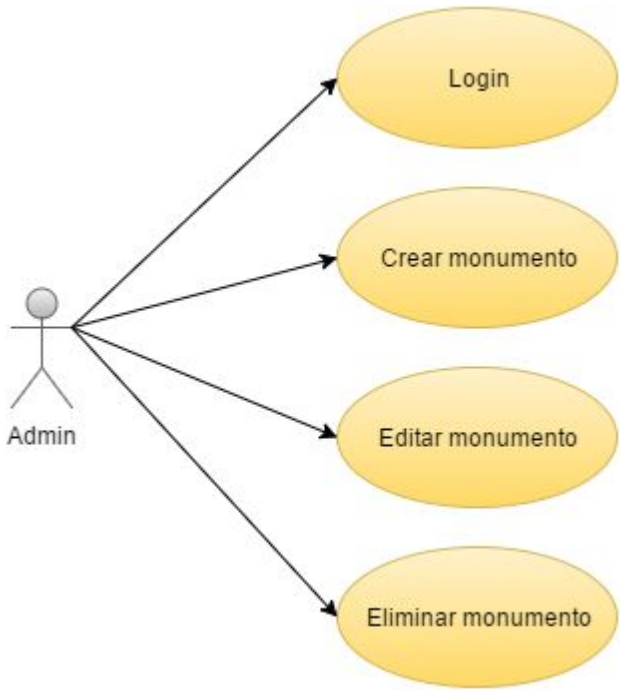


Figura 3.3 Caso de uso admin

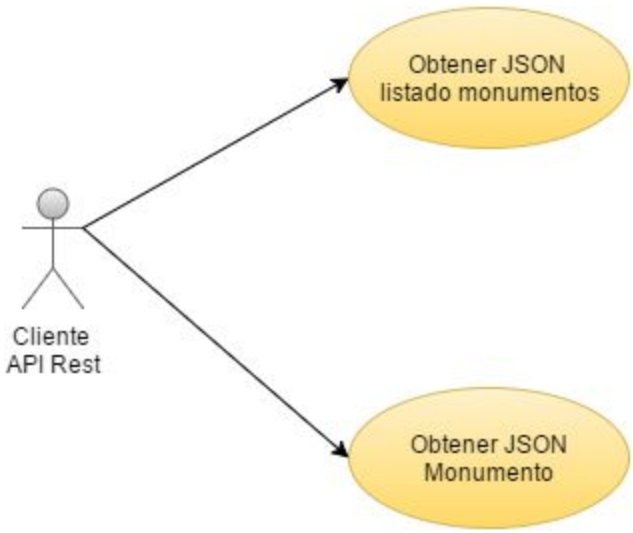


Figura 3.4 Caso de uso cliente API Rest

3.3.3 Especificación de casos de uso

UC-01	***Ver listado monumentos***	
Versión	1.0	
Actor principal	ACT-01	
Actor secundario	-	
Trigger	Pulsar botón “La guía”	
Descripción	El usuario visualiza un listado de los monumentos registrados en la aplicación	
Precondición	PRE-01 · API Rest operativa	
Secuencia normal	**Paso**	**Acción**
	p1	El usuario entra en la aplicación.
	p2	El usuario pulsa el botón “La Guía”.
	p3	La aplicación solicita la información a la API Rest.
	p4	El API Rest devuelve los datos solicitados.
	p5	La aplicación muestra el listado de monumentos.
Postcondiciones	El usuario visualiza la lista de monumentos	
Excepciones	**Paso**	**Acción**
	p4	Si el API Rest no está disponible la aplicación lanzará una alerta diciendo “El sistema no puede mostrar la información solicitada”
Frecuencia	*Muy alta*	
Importancia	*Muy Alta*	
Comentarios	-	

Tabla 3.4 Caso de uso “Ver listado monumentos”

UC-02	***Ver información monumentos***	
Versión	1.0	
Actor principal	ACT-01	
Actor secundario	*ACT-02*	
Trigger	• TR-01 - Pulsar monumento • TR-02 - Pulsar notificación	
Descripción	El usuario visualiza información sobre el monumento seleccionado	
Precondición	PRE-01 · API Rest operativa	
Secuencia normal	**Paso**	**Acción**
	p1	El usuario entra en la aplicación.
	p2	El usuario pulsa el botón “La Guía”.
	p3	La aplicación muestra el listado de monumentos.
	p4	El usuario selecciona un monumento
	p5	*La aplicación solicita información sobre ese monumento al API Rest*
	p6	*El API Rest envía la información solicitada*
	p7	*La aplicación muestra la información sobre el monumento*
Postcondiciones	El usuario visualiza información sobre un monumento	
Excepciones	**Paso**	**Acción**
	p5	Si el API Rest no está disponible la aplicación lanzará una alerta diciendo “El sistema no puede mostrar la información solicitada”
Frecuencia	*Muy alta*	
Importancia	*Muy Alta*	
Comentarios	-	

Tabla 3.5 Caso de uso “Ver información de monumento”

UC-03	***Ver mapa***	
Versión	1.0	
Actor principal	ACT-01	
Actor secundario	-	
Trigger	El usuario pincha el botón “El mapa”	
Descripción	El usuario visualiza un mapa indicando su posición y la de los monumentos cercanos	
Precondición	-	
Secuencia normal	**Paso**	**Acción**
	p1	El usuario entra en la aplicación.
	p2	El usuario pulsa el botón “El mapa”.
	p3	La aplicación abre la aplicación de mapas del dispositivo
Postcondiciones	El usuario visualiza el mapa con la información especificada anteriormente	
Excepciones	**Paso**	**Acción**
	-	-
Frecuencia	*Alta*	
Importancia	*Alta*	
Comentarios	-	

Tabla 3.6 Caso de uso “Ver mapa”

UC-04	***Pulsar notificación***	
Versión	1.0	
Actor principal	ACT-01	
Actor secundario	-	
Trigger	La aplicación ha detectado un monumento cercano y lanza una notificación	
Descripción	El usuario recibe una notificación de monumento cercano y la pulsa	
Precondición	PRE-01 · API Rest operativa	
Secuencia normal	**Paso**	**Acción**
	p1	El usuario recibe una notificación.
	p2	El usuario pulsa la notificación recibida.
	p3	El sistema se conecta con la API Rest y solicita información sobre el monumento localizado
	p4	El API Rest devuelve la información solicitada
	p5	La notificación dirige al usuario a la página de información del monumento
Postcondiciones	El usuario visualiza la información del monumento localizado	
Excepciones	**Paso**	**Acción**
	-	-
Frecuencia	*Muy Alta*	
Importancia	*Muy Alta*	
Comentarios	-	

Tabla 3.7 Caso de uso "Pulsar notificación"

UC-05	***Obtener JSON Listado***	
Versión	1.0	
Actor principal	Cliente API Rest	
Actor secundario	-	
Trigger	El usuario pulsa el botón guía	
Descripción	El Cliente API Rest recibe una petición de listado	
Precondición	PRE-01 · API Rest operativa	
Secuencia normal	**Paso**	**Acción**
	p1	El usuario pulsa el botón “La guía”
	p2	La aplicación envía al API Rest la petición de listado de monumentos
	p3	El API Rest obtiene los datos solicitados de la base de datos
	p4	El API Rest devuelve la información solicitada
Postcondiciones	El usuario visualiza la lista de monumentos registrados	
Excepciones	**Paso**	**Acción**
	-	-
Frecuencia	*Muy Alta*	
Importancia	*Muy Alta*	
Comentarios	-	

Tabla 3.8 Caso de uso “Obtener JSON Listado”

UC-06	***Obtener JSON Monumento***	
Versión	1.0	
Actor principal	Cliente API Rest	
Actor secundario	-	
Trigger	El usuario pulsa un monumento en el listado o pulsa la notificación	
Descripción	El Cliente API Rest recibe una petición de información de monumento	
Precondición	PRE-01 · API Rest operativa PRE-02 · Listado de monumentos visible PRE-03 · Notificación de monumento lanzada	
Secuencia normal	**Paso**	**Acción**
	p1	El usuario pulsa un monumento de los mostrados en el listado
	p2	La aplicación envía al API Rest la petición de información de monumento
	p3	El API Rest obtiene los datos solicitados de la base de datos
	p4	El API Rest devuelve la información solicitada
Postcondiciones	El usuario visualiza información sobre el monumento solicitado	
Excepciones	**Paso**	**Acción**
	-	-
Frecuencia	*Muy Alta*	
Importancia	*Muy Alta*	
Comentarios	-	

Tabla 3.9 Caso de uso "Obtener JSON Monumento"

UC-06	***Login***	
Versión	1.0	
Actor principal	Administrador	
Actor secundario	-	
Trigger	El administrador pincha el botón de “Iniciar sesión”	
Descripción	El administrador se identifica en la interfaz web para poder realizar modificaciones sobre la base de datos	
Precondición	PRE-01 · Base de datos disponible	
Secuencia normal	**Paso**	**Acción**
	p1	El administrador pulsa el botón “Iniciar sesión”
	p2	El administrador introduce su nombre de usuario y su contraseña
	p3	La interfaz web confirma los datos introducidos comparandolos con la base de datos
	p4	La interfaz web muestra un mensaje indicando el inicio de sesión correcto
Postcondiciones	El administrador inicia sesión	
Excepciones	**Paso**	**Acción**
	p3	Si los datos introducidos son erróneos, la interfaz web enviará un mensaje diciendo que los datos no son correctos
Frecuencia	*Alta*	
Importancia	*Alta*	
Comentarios	-	

Tabla 3.10 Caso de uso “Login”

UC-07	***Crear monumento***	
Versión	1.0	
Actor principal	Admin	
Actor secundario	-	
Trigger	El administrador pincha la opción “Crear monumento”	
Descripción	El administrador puede añadir nuevos monumentos a la base de datos	
Precondición	PRE-01 · Base de datos disponible PRE-02 · Administrador logueado	
Secuencia normal	**Paso**	**Acción**
	p1	El administrador pulsa el botón “Crear monumento”
	p2	La interfaz web muestra los campos a rellenar, a saber: nombre, localización, foto y descripción.
	p3	El administrador rellena los datos necesarios
	p4	La interfaz web valida los datos introducidos
	p5	La interfaz web introduce los nuevos datos en la base de datos
Postcondiciones	Monumento agregado a la base de datos	
Excepciones	**Paso**	**Acción**
	-	-
Frecuencia	*Baja*	
Importancia	*Baja*	
Comentarios	-	

Tabla 3.11 Caso de uso “Crear monumento”

UC-08	***Actualizar monumento***	
Versión	1.0	
Actor principal	Admin	
Actor secundario	-	
Trigger	El administrador pincha la opción “Actualizar monumento”	
Descripción	El administrador puede modificar la información de los monumentos registrados en el sistema	
Precondición	PRE-01 · Base de datos disponible PRE-02 · Administrador logueado PRE-03 · Monumento existente	
Secuencia normal	**Paso**	**Acción**
	p1	El administrador pulsa el botón “Actualizar monumento”
	p2	La interfaz web muestra la lista de monumentos registrados.
	p3	El administrador selecciona el monumento a modificar
	p4	La interfaz muestra la información de ese monumento
	p5	El administrador modifica los datos oportunos y pulsa el botón “Guardar cambios”
	p6	La interfaz web envía la nueva información a la base de datos
Postcondiciones	Monumento modificado	
Excepciones	**Paso**	**Acción**
	-	-
Frecuencia	*Baja*	
Importancia	*Baja*	
Comentarios	-	

Tabla 3.12 Caso de uso “Actualizar monumento”

UC-08	***Borrar monumento***	
Versión	1.0	
Actor principal	Admin	
Actor secundario	-	
Trigger	El administrador pincha la opción “Borrar monumento”	
Descripción	El administrador puede borrar un monumento registrado en la base de datos	
Precondición	PRE-01 · Base de datos disponible PRE-02 · Administrador logueado PRE-03 · Monumento existente	
Secuencia normal	**Paso**	**Acción**
	p1	El administrador pulsa el botón “Borrar monumento”
	p2	La interfaz web muestra la lista de monumentos registrados.
	p3	El administrador selecciona el monumento a borrar
	p4	La interfaz muestra un mensaje diciendo : “Realmente desea borrar el monumento?”
	p5	El administrador confirma el borrado del monumento
	p6	La interfaz web envía un mensaje al administrador indicando el éxito del borrado
Postcondiciones	Monumento borrado	
Excepciones	**Paso**	**Acción**
	-	-
Frecuencia	*Muy Baja*	
Importancia	*Muy Baja*	
Comentarios	-	

Tabla 3.13 Caso de uso “Borrar monumento”

3.4 Requisitos funcionales

- **RF-01** La aplicación accederá al sistema GPS del dispositivo.
- **RF-02** La aplicación accederá al sistema *WiFi* del dispositivo.
- **RF-03** La aplicación accederá al sistema de internet móvil del dispositivo.
- **RF-04** La aplicación permitirá ver una lista de los monumentos registrados.
- **RF-05** La aplicación se conectará a una API Rest externa para obtener información sobre los monumentos.
- **RF-06** La aplicación permitirá ver información sobre los monumentos almacenados en la base de datos.
- **RF-07** La aplicación permitirá ver un mapa con nuestra localización actual y los monumentos cercanos a nuestra posición.
- **RF-08** La aplicación avisará al usuario de la localización de un monumento cercano mediante una notificación push.
- **RF-10** La aplicación permitirá al usuario visualizar información sobre un monumento localizado pulsando la notificación push recibida.
- **RF-11** La aplicación permitirá al usuario confirmar su salida de la misma.
- **RF-12** El usuario administrador podrá crear nuevos monumentos.
- **RF-13** El usuario administrador podrá modificar monumentos existentes
- **RF-14** El usuario administrador podrá borrar monumentos.

3.5 Requisitos de información

Los requisitos de información describen la información que debe almacenar y gestionar el sistema para dar soporte a los procesos de negocio.

3.5.1 Modelo conceptual

Para ilustrar los diferentes tipos de entidades y relaciones que se pueden dar, se crea el siguiente modelo utilizado como una descripción conceptual de la base de datos.

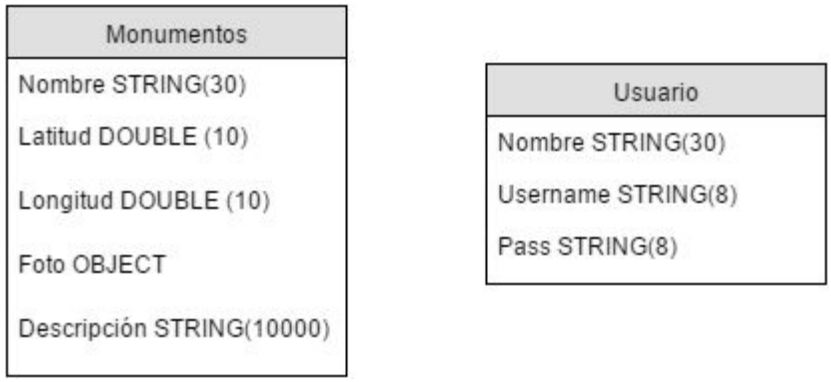


Figura 3.5 Modelo conceptual de base de datos

3.5.2 Diccionario de datos

ENTIDAD	ATRIBUTO	VALOR	DESCRIPCIÓN
Monumento	Nombre	Cadena de caracteres	Nombre que identifica el monumento
	Latitud	Entero decimal	Coordenada del monumento
	Longitud	Entero decimal	Coordenada del monumento
	Foto	Foto	Imagen del monumento
	Descripción	Cadena de caracteres	descripción del monumento

Tabla 3.14 Diccionario de datos

3.6 Requisitos de interfaces externas

- **RIE-01** El sistema obtendrá la información de los monumentos conectándose a una API Rest externa que tendrá almacenados todos los datos disponibles de un monumento, a saber: nombre, latitud, longitud, foto y descripción del mismo.

3.7 Requisitos no funcionales

- **RNF-01** El sistema debe funcionar en la mayoría de dispositivos con sistema operativo Android.
- **RNF-02** El sistema deberá resultar sencillo de usar y se podrá acceder cada función del sistema en un tiempo estimado menor a 1 segundo.
- **RNF-03** El sistema deberá ser capaz de capturar todos los posibles fallos y tratarlos adecuadamente.
- **RNF-04** El software debe ser lo más modular posible pensando en que sea reutilizable.
- **RNF-05** El sistema debe ser eficiente a la hora de realizar consultas y hacer uso de sus recursos con el objetivo de tener un alto rendimiento, que el usuario no espere más de 1 segundo en ver resultados.
- **RNF-06** El sistema debe estar disponible las 24 horas del día, 7 días de la semana.
- **RNF-07** Los requisitos hardware necesarios para el sistema deben ser fácilmente alcanzables, como un dispositivo móvil inteligente que tenga un sistema operativo del que disponga la mayor parte de usuarios de estos dispositivos, y una memoria interna suficiente para instalar la aplicación.
- **RNF-08** Debe ser fácil de mantener para poder añadir nuevas funcionalidades sin que exista detrimento de la calidad o del funcionamiento ya alcanzado.

3.8 Requisitos de internacionalización y localización

- **RIL-01** La primera versión (1.0) sólo soportará el idioma español.

CAPÍTULO 4

DISEÑO

La etapa de diseño permite refinar el modelo del análisis hasta obtener un diseño del sistema adecuado, considerando los requisitos no funcionales y restricciones del entorno, para después pasar a la implementación. Además esta etapa, permite tener una idea más clara del sistema, ya que elimina parte de la abstracción de la etapa anterior.

4.1 Tecnologías

Para la implementación de esta aplicación se han utilizado varias tecnologías:

- Apache Cordova como marco de desarrollo. Para crear una aplicación Android requiere además:
    - El SDK de Android.
    - JDK de Java .
- Atom como entorno de desarrollo, es un editor de código integrado con github y de uso gratuito.
- HTML, CSS, Angular y JavaScript para programar el lado cliente.
- Node para programar el lado servidor.

Además, debido a que la aplicación es para dispositivos móviles, se deben tener en cuenta sus limitaciones, y en base a ello se han tomado las siguientes decisiones:

- Dado que muchos dispositivos no disponen de mucha memoria disponible, externalizamos el almacenamiento de información, es decir, la aplicación sólo será un esqueleto, y cogerá la información de un servicio Rest externo.
- La decisión anterior conlleva que, para poder disponer de toda la información, el dispositivo tendrá que estar conectado a la red de datos continuamente. Para evitar un consumo de datos masivo, hemos definido un rango de tiempo, en el que cada 5 minutos, la aplicación comprueba los monumentos cercanos, y en caso de haberlos pide información.
- Como tener la aplicación abierta constantemente puede llevar a un uso excesivo de batería, la aplicación podrá ejecutarse en segundo plano.

4.2 Arquitectura lógica

Una aplicación Cordova está formada por varios componentes. El siguiente diagrama muestra una vista de alto nivel de la arquitectura de la aplicación Cordova:

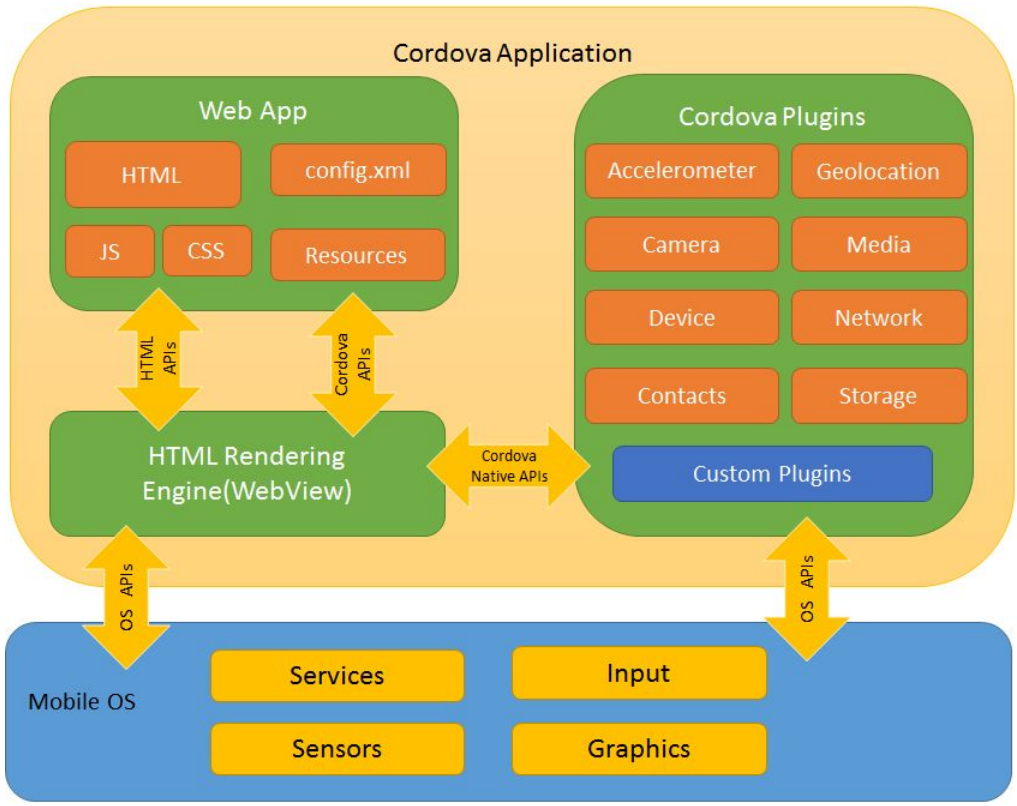


Figura 4.1 Arquitectura de una aplicación Cordova

➔ **Web App**

Esta es la parte en la que reside el código de aplicación. La propia aplicación se implementa como una página web, en un archivo local denominado index.html, conformada por archivos CSS, JavaScript, imágenes, archivos multimedia u otros recursos son necesarios para que se ejecute. La aplicación se ejecuta en un *WebView* dentro de la envoltura de aplicación nativa.

➔ **Web View**

El *WebView* ofrece la aplicación con la totalidad de su interfaz de usuario. En algunas plataformas también puede ser un componente dentro de una aplicación

híbrida más grande, que mezcla el *WebView* con los componentes de aplicaciones nativas.

➔ **Plugins**

Los *plugins* son una parte integral del ecosistema Cordova. Proporcionan una interfaz para Cordova y componentes nativos para comunicarse entre sí y los enlaces con las diferentes APIs del dispositivo. Esto permite invocar código nativo de JavaScript.

Cordova mantiene un conjunto de plugins llamado core-plugins. Estos complementos del núcleo permiten acceder a diferentes componentes del dispositivo, tales como la batería, cámara, contactos, etc.

Por otro lado, hay varios plugins de terceros que proporcionan funciones adicionales para características no necesariamente disponibles en todas las plataformas.

Además, Cordova no proporciona ningún widget de interfaz de usuario o ningún *framework* MV*. Cordova ofrece sólo el tiempo de ejecución en que se pueden ejecutar.

4.3 Arquitectura física

La arquitectura física se limita al dispositivo móvil y al servicio Rest. Cuando el usuario se encuentre dentro de un rango determinado, la aplicación pedirá al servicio Rest los datos del monumento localizado.

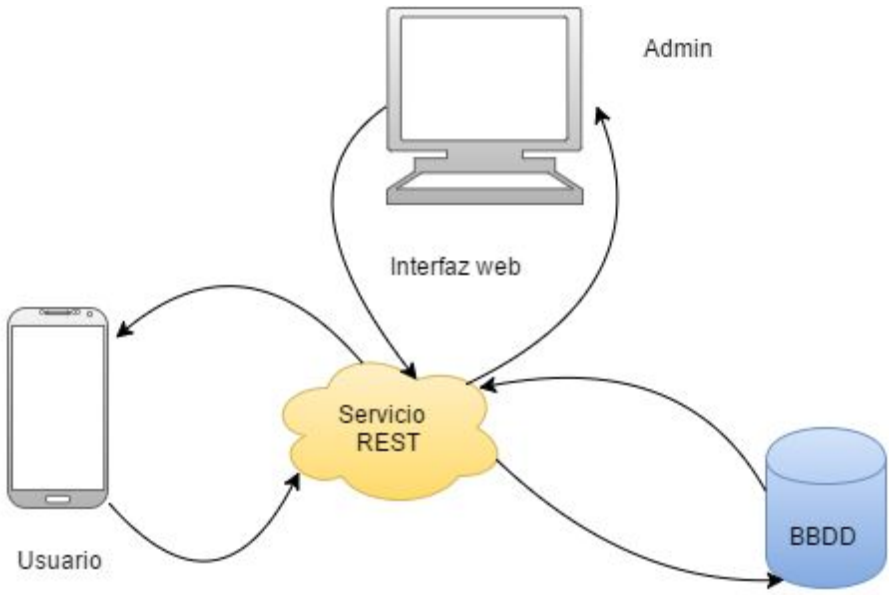


Figura 4.2 Arquitectura física añadir etiquetas admin y usuario

4.4 Clases de diseño

Este tipo de diagrama describe la estructura del sistema, mostrando las clases, con sus atributos y métodos correspondientes, y las relaciones entre ellas. El diagrama de clases de diseño es elaborado para tener en cuenta los detalles concretos de la implementación del sistema a desarrollar.

Dado que es una aplicación web, tenemos 6 clases que son fundamentales:

- Index: carga la pantalla principal
- Home: muestra las opciones que puede realizar el usuario
- Guide: muestra la guía de monumentos
- Monument: muestra información sobre monumentos
- Map: muestra un mapa que sitúa al usuario
- Script: contiene todos los métodos necesarios para posicionar al usuario y obtener la información del API Rest.

4.5 Diagramas de secuencia

Los diagramas de secuencia muestran cómo se comunica la aplicación entre sí y la secuencia de mensajes que se realizan entre ella durante un escenario concreto. A continuación se mostrarán los diagramas de secuencia de los casos de uso que se especificaron en el apartado de análisis:

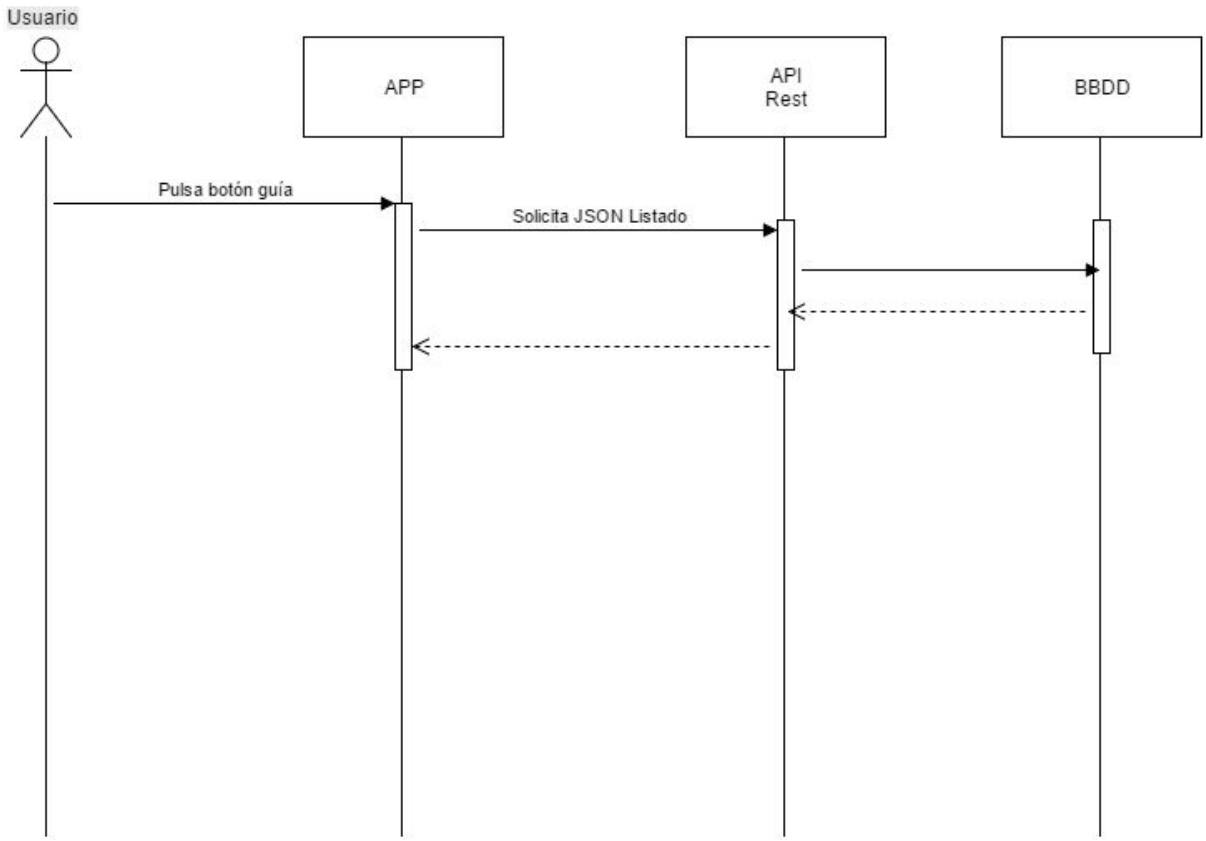


Figura 4.3 Diagrama de secuencia correspondiente a UC-01 Añadir operaciones

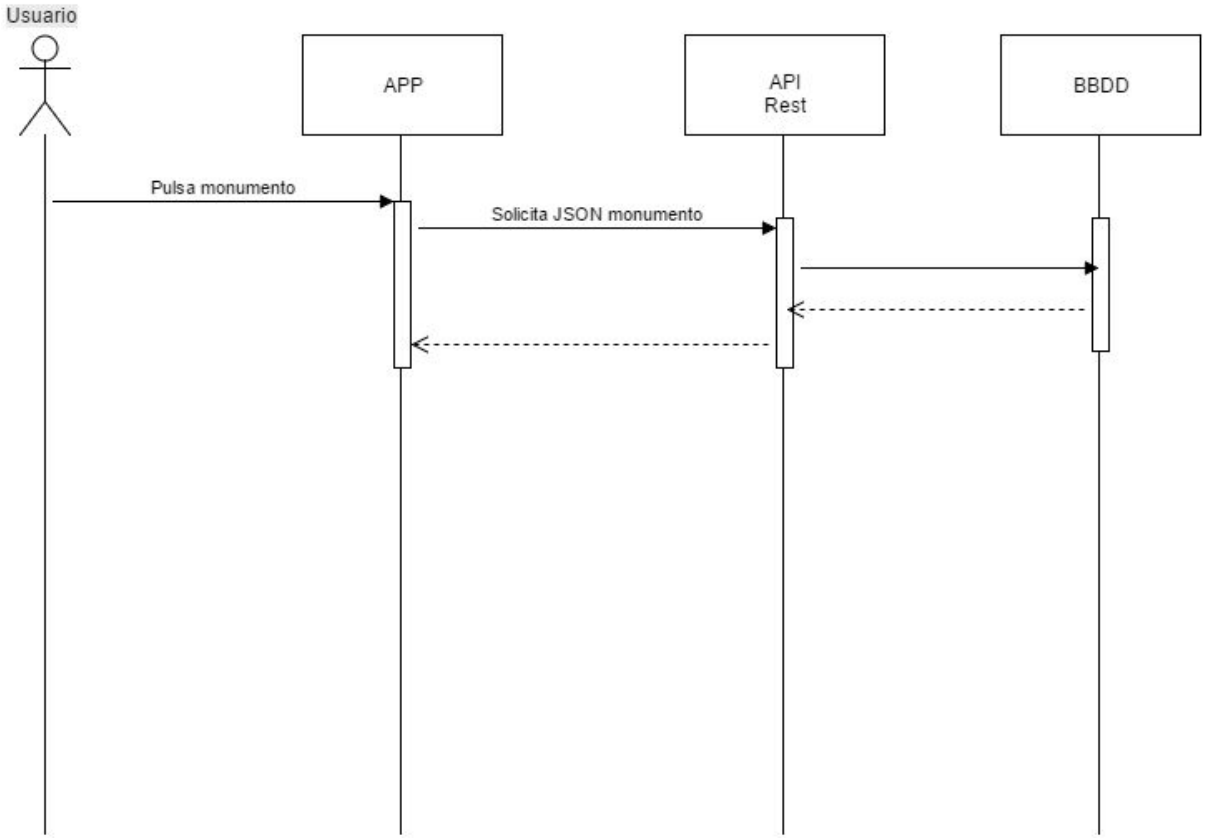


Figura 4.4 Diagrama de secuencia correspondiente a UC-02

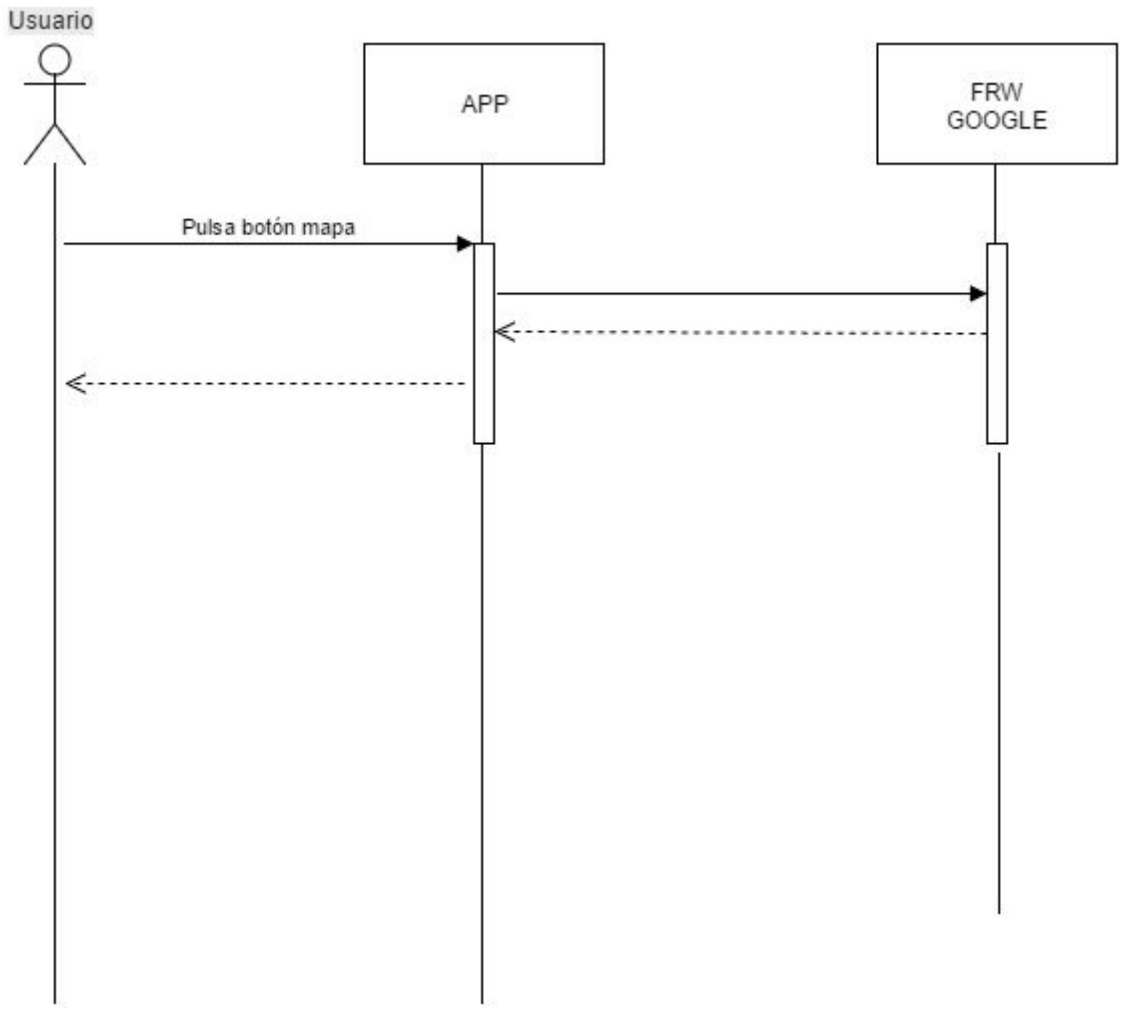


Figura 4.5 Diagrama de secuencia correspondiente a UC-03

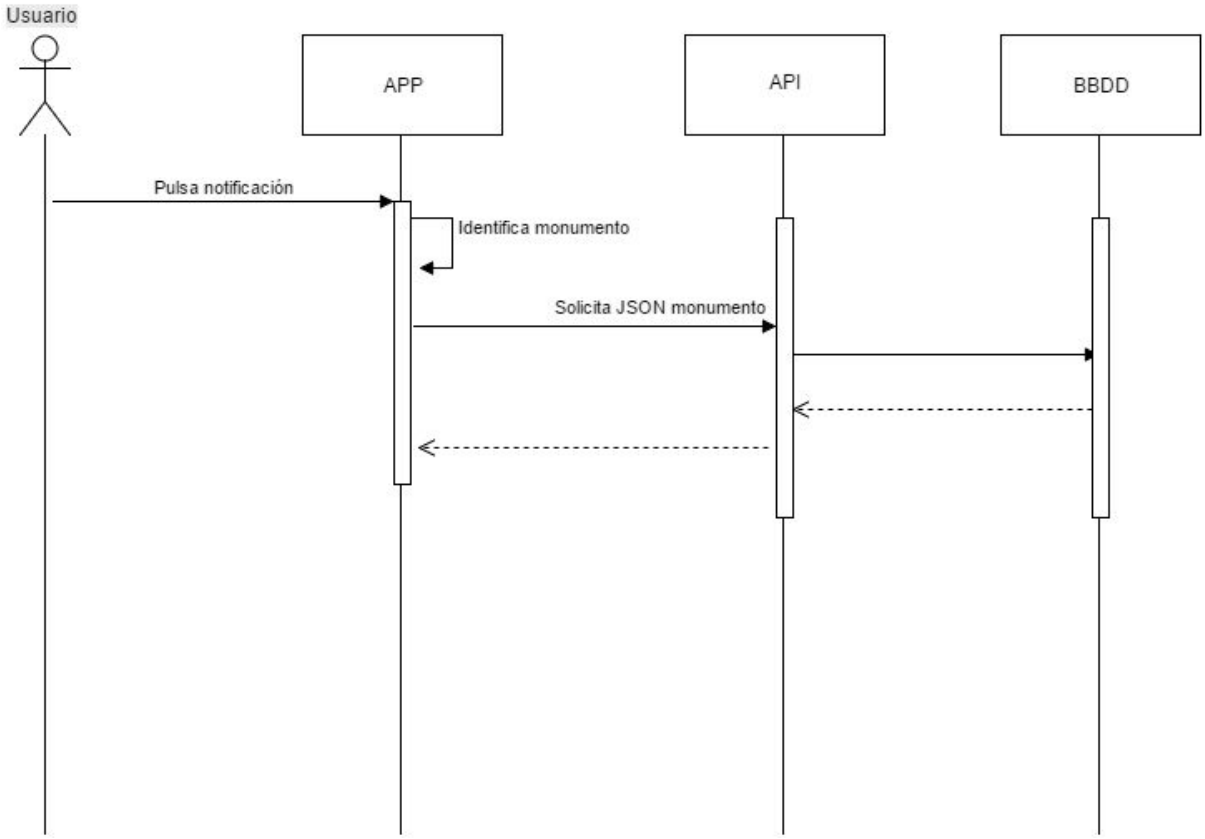


Figura 4.6 Diagrama de secuencia correspondiente a UC-04

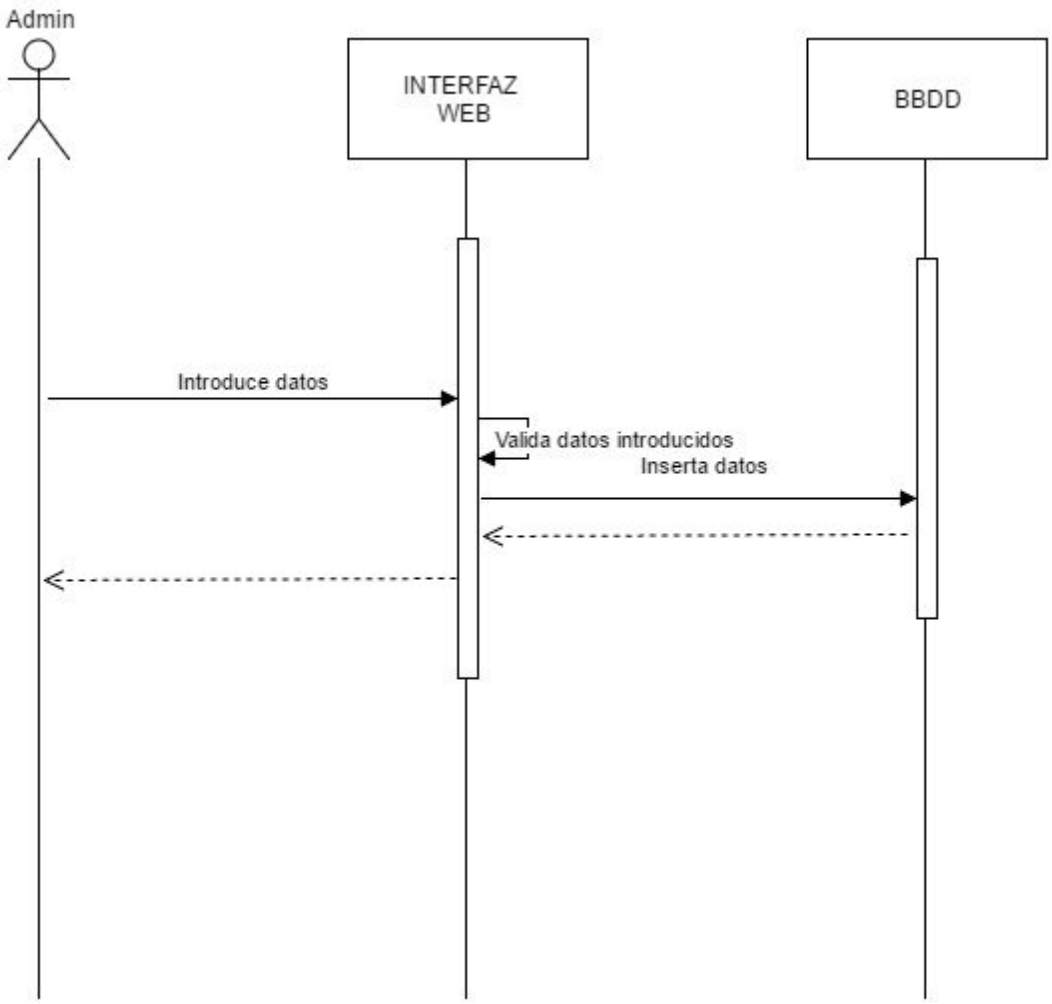


Figura 4.7 Diagrama de secuencia correspondiente a UC-07

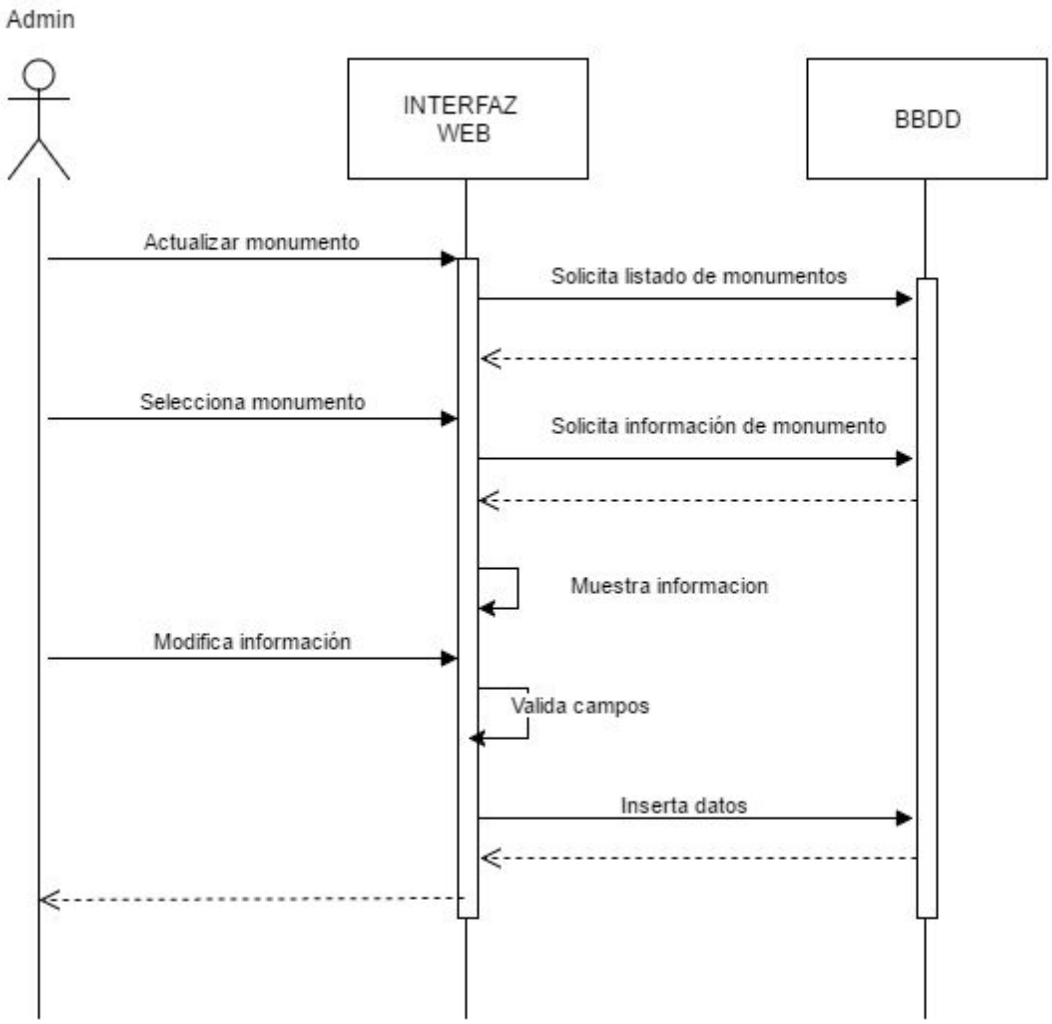


Figura 4.8 Diagrama de secuencia correspondiente a UC-08

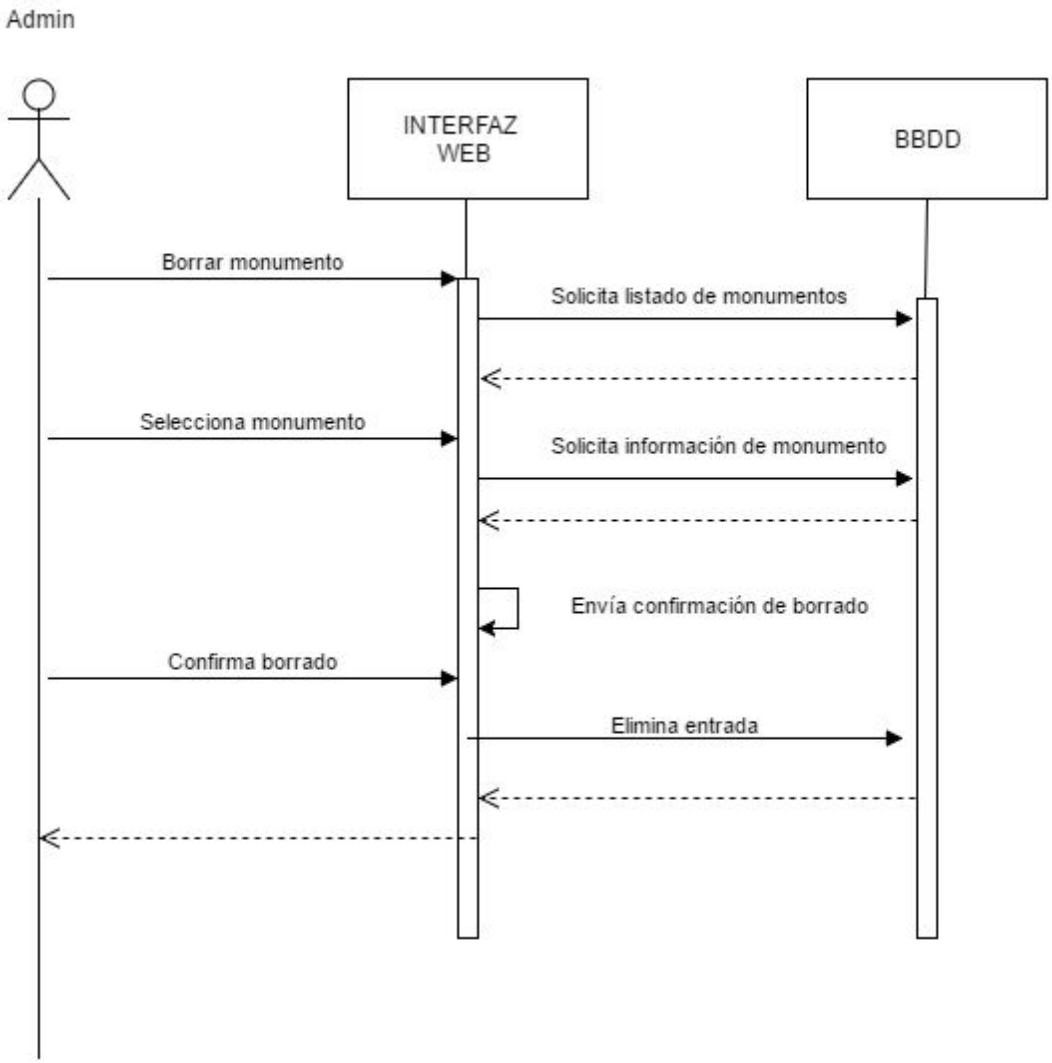


Figura 4.9 Diagrama de secuencia correspondiente a UC-09

4.6 Factores que determinan la posición del usuario

A la hora de situar al usuario en una posición determinada se deben tener en cuenta dos factores principales: la posición del usuario y la localización del monumento. Con esos dos valores, la aplicación es capaz de avisar al usuario cuando tiene un monumento a una distancia cercana.

Ayudándonos de la tecnología de Google y del sistema de geolocalización que proporciona HTML5, situar al usuario es una tarea fácil. Se define un área circular alrededor del centro de coordenadas del monumento deseado. También se define un radio para ese área

circular, que será de 300 metros. Entonces, para que salte la notificación, la posición del usuario deberá estar dentro de ese área.

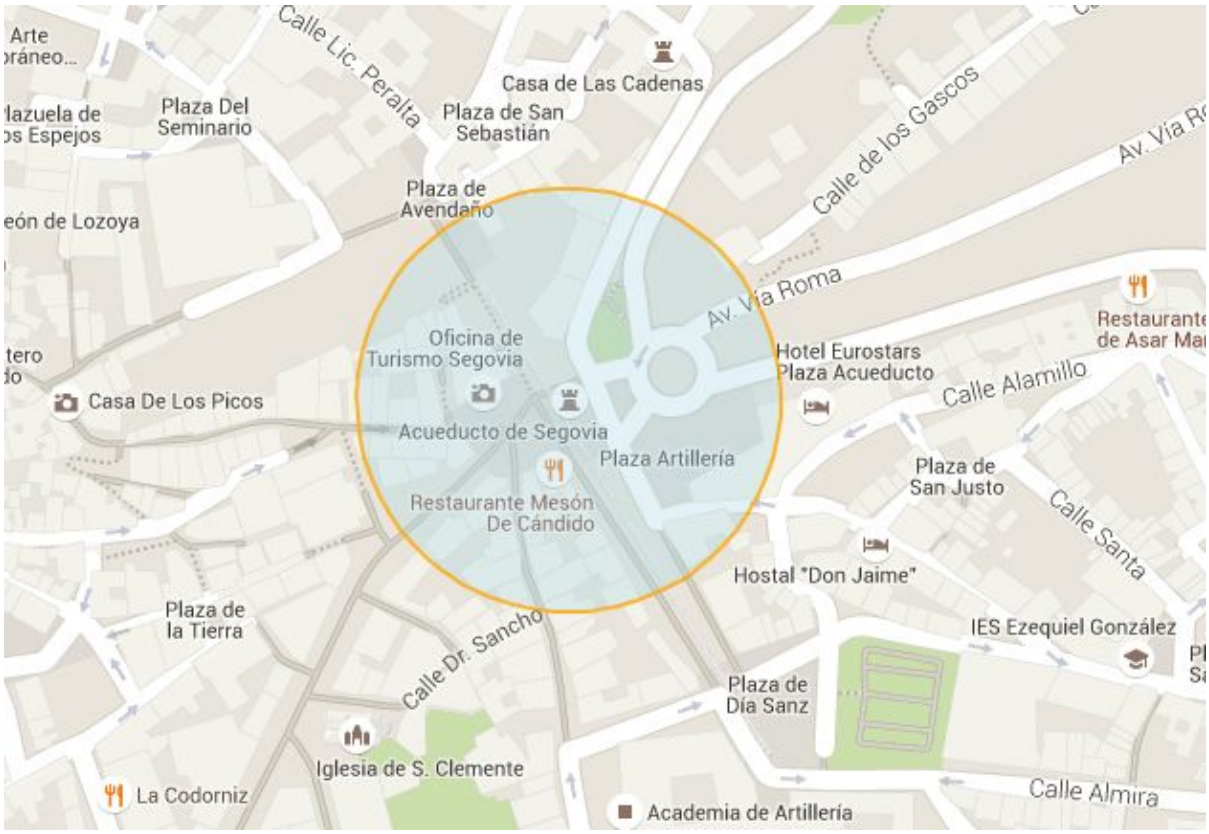


Figura 4.10 Ejemplo de área circular definida.

Mediante el uso de una librería del API de Google (concretamente *"geometry spherical"*) y de una función de la librería mencionada anteriormente (*ComputeDistanceBetwen)*, la aplicación calcula la posición de la siguiente manera: si la distancia del centro del área definida a la posición del usuario es menor que la del centro del área a su límite (es decir, el radio), se considera que el usuario está dentro del área. Por ello, a la función que calcula la distancia se le ha de pasar el centro de coordenadas del área definida del monumento, la posición del usuario y el radio de acción.

4.7 Diseño de la interfaz

El aspecto elegido para la aplicación tiene que ser un diseño sencillo (debido el reducido tamaño de los dispositivos móviles) y que resulte intuitivo y fácil de usar para cualquier persona.

En las siguientes tablas se presenta la estructura y distribución de los elementos de las pantallas principales que forman la aplicación.

Nombre	**index.html**
Descripción	Pantalla principal que muestra dos botones con los que se puede acceder a los distintos módulos.
Activación	Al acceder a la aplicación.
Boceto	
Opciones	Ver guía , Ver mapa

Tabla 4.1 Diseño de index

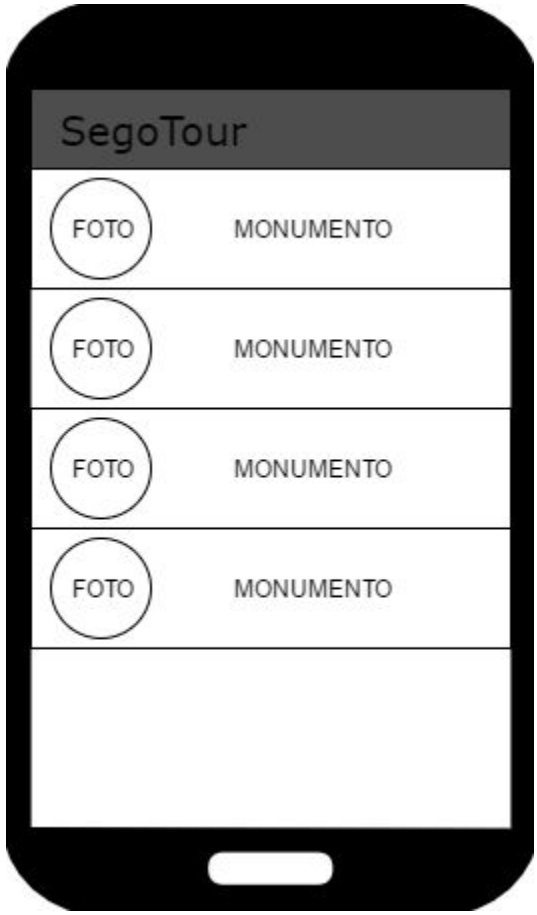
Nombre	**guia.html**
Descripción	Pantalla que muestra todos los monumentos registrados en la aplicación.
Activación	Pulsando en index.html el botón de “La guía”
Boceto	
Opciones	Ver monumento

Tabla 4.2 Diseño de la interfaz guide

Nombre	**monumento.html**
Descripción	Pantalla que muestra una foto del monumento y una breve descripción
Activación	Pulsando en guia.html cualquiera de los monumentos.
Boceto	
Opciones	-

Tabla 4.3 Diseño de interfaz monument

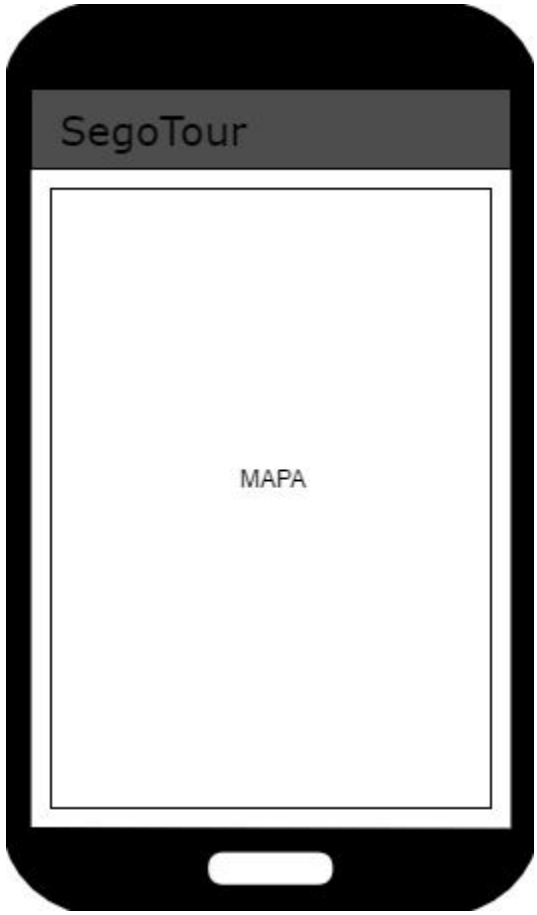
Nombre	**mapa.html**
Descripción	Pantalla que muestra un mapa con la localización del usuario y los monumentos cercanos
Activación	En index.html, pinchando el botón “Mapa”
Boceto	
Opciones	-

Tabla 4.4 Diseño de la interfaz mapa

CAPÍTULO 5

IMPLEMENTACIÓN Y PRUEBAS

Una vez se tienen claros los aspectos principales sobre el análisis y el diseño, se puede dar paso a la implementación del sistema, pues se debe tener el conocimiento necesario para llevarlo a cabo.

5.1. Introducción a Cordova

Como ya se ha mencionado antes, Apache Cordova es un marco para desarrollo móvil de código abierto, el cual nos permite utilizar lenguajes web como HTML5, CSS3 y JavaScript en un desarrollo multiplataforma, evitando de este modo el desarrollo en lenguajes nativos como Java en Android.

Las aplicaciones se ejecutan dentro de un encapsulado dependiente de la plataforma indicada y dependen de enlaces estándares a las APIs para poder tener acceso a los sensores de los dispositivos, los datos y el estado de la red.

5.1.2. Componentes de un proyecto Cordova

Las aplicaciones Cordova se basan en un archivo común llamado *config.xml*,que proporciona información acerca de la aplicación y especifica los parámetros que afectan a cómo funciona, cómo responde a la orientación o qué hace cuando se mueve el dispositivo. Este archivo se adhiere a la especificación de Empaquetado de la aplicación Web de la W3C.

La misma aplicación se implementa como una página web. Un archivo local llamado index.html hace referencia a cualquier CSS, JavaScript, imágenes, archivos multimedia u otros recursos que son necesarios para que se ejecute de forma predeterminada. La aplicación se ejecuta como un WebView dentro de la envoltura de la aplicación nativa, que distribuye a tiendas de aplicaciones.

El WebView Cordova puede proporcionar la aplicación con su interfaz de usuario completa. En algunas plataformas, también puede ser un componente dentro de una aplicación híbrida más grande, que mezcla la vista Web con componentes de la aplicación nativa. Cordova también dispone de numerosos plugins para comunicar la aplicación con los demás componentes nativos del dispositivo.

5.1.2. Estructura de un proyecto Cordova

- Carpeta *Hooks*: contiene archivos que representan secuencias de comandos especiales que podrían ser añadidos por la aplicación y desarrolladores de plugins o incluso por su propio sistema de construcción para personalizar los comandos de Cordova.
- Carpeta *Platforms*: contiene todos los sistemas operativos en los que dispondremos de la aplicación. Cada sistema operativo crea una carpeta con los archivos necesarios para generar el paquete instalador en el dispositivo móvil.
- Carpeta *Plugins*: contiene los plugins adicionales que queramos añadir a nuestra aplicación.
- Carpeta *Www*: contiene toda la parte de programación de la aplicación y su funcionalidad.
- Archivo *config.xml*: proporciona información acerca de la aplicación.

5.2. Estructura del proyecto

El código implementado se localiza en la carpeta *Www* mencionada anteriormente.

1. Carpeta *CSS*: contiene todos los archivos .css que dan estilo a la aplicación.
2. Carpeta *Fonts*: contiene las diferentes fuentes que utiliza la aplicación.
3. Carpeta *Images*: contiene todas las imágenes utilizadas en la aplicación.
4. Carpeta *JS*: contiene todos los archivos JavaScript necesarios para dotar a la aplicación de funcionalidad.
5. Archivo *index.html*, es la pantalla de inicio de la aplicación.
6. Archivo guide.ejs, es el archivo que muestra la guía de monumentos de la aplicación.
7. Archivo *monument.ejs*, es el archivo que muestra un monumento en concreto.
8. Archivo *map.html*, muestra el mapa que localiza al usuario.

5.3. Aspectos relevantes de implementación

El aspecto más destacado de todo el proyecto es la localización de monumentos a través de la posición del usuario. Dado que localizar al usuario continuamente supone un

gasto de batería y datos considerable, se ha optado por establecer un intervalo de tiempo de localización. Esto quiere decir que cada x minutos se localiza al usuario y se comprueba el número de monumentos a su alrededor.

El intervalo de tiempo establecido ha sido de cinco minutos, tiempo suficiente para que el usuario visualice el monumento y tenga tiempo de desplazarse hasta una nueva localización. Para poder implementar esto se han utilizado dos tecnologías principales: API de Google, que proporciona los métodos para determinar si un usuario está cerca de un monumento, y el sistema de geolocalización de HTML5, que nos permite obtener las coordenadas del usuario en todo momento.

5.4. Pruebas

Como se indicó al principio del trabajo, la metodología utilizada en el desarrollo de la aplicación ha sido el modelo ágil *Scrum*, donde en cada *sprint* se utilizaba un método iterativo. Así que con cada iteración se han realizado una serie de test unitarios antes de continuar con la siguiente.

Se entiende que los casos en los que únicamente se visualizan datos sin que sea necesaria la interacción con el usuario, no son casos problemáticos, por lo que no es necesario la descripción de algunos de esos casos de prueba.

CASO DE PRUEBA	Visualizar guía
Propósito	Ver la lista de monumentos disponible
Prerrequisitos	-
Datos de entrada	-
Pasos	El usuario pulsa el botón “La Guía” en la pantalla de inicio
Resultado esperado	Visualizar lista de monumentos
Resultado obtenido	Se visualiza la lista de monumentos

Tabla 5.1 Caso de prueba visualizar guía

CASO DE PRUEBA	Visualizar monumento
Propósito	Ver información sobre un monumento
Prerrequisitos	El monumento debe estar en la lista de monumentos registrados
Datos de entrada	-
Pasos	El usuario elige un monumento de la lista de monumento
Resultado esperado	Visualizar información del monumento
Resultado obtenido	Se visualiza información sobre el monumento

Tabla 5.2 Caso de prueba visualizar monumento

CASO DE PRUEBA	Visualizar mapa
Propósito	Ver posición del usuario en el mapa y monumentos cercanos
Prerrequisitos	-
Datos de entrada	-
Pasos	El usuario pulsa el botón mapa de la pantalla de inicio
Resultado esperado	Visualizar el mapa con la posición del usuario y los monumentos cercanos
Resultado obtenido	Se visualiza el mapa con la posición del usuario y los monumentos cercanos

Tabla 5.3 Caso de prueba visualizar mapa

CASO DE PRUEBA	Obtener notificación
Propósito	Obtener una notificación push cuando estemos cerca de un monumento
Prerrequisitos	Datos móviles activos
Datos de entrada	Coordenadas del usuario
Pasos	El usuario se sitúa en las cercanías de un monumento
Resultado esperado	Obtener una notificación push con información del monumento
Resultado obtenido	Se obtiene una notificación push con información del monumento

Tabla 5.4 Caso de prueba obtener notificación

CAPÍTULO 6

MANUALES

6.1. Manual de instalación

Al tratarse de una aplicación con base de datos externa, será necesario instalar la aplicación en un dispositivo físico con sistema operativo android y la creación e instalación de la base de datos en un entorno externo.

1. **Instalación de la aplicación en dispositivo físico**. Se descarga el archivo .apk y se instala.

Figura 6.1 . Explicación de instalación en un dispositivo móvil.

6.2 Manual de usuario

6.2.1 Gestión de monumentos

❖ Ver listado de monumentos

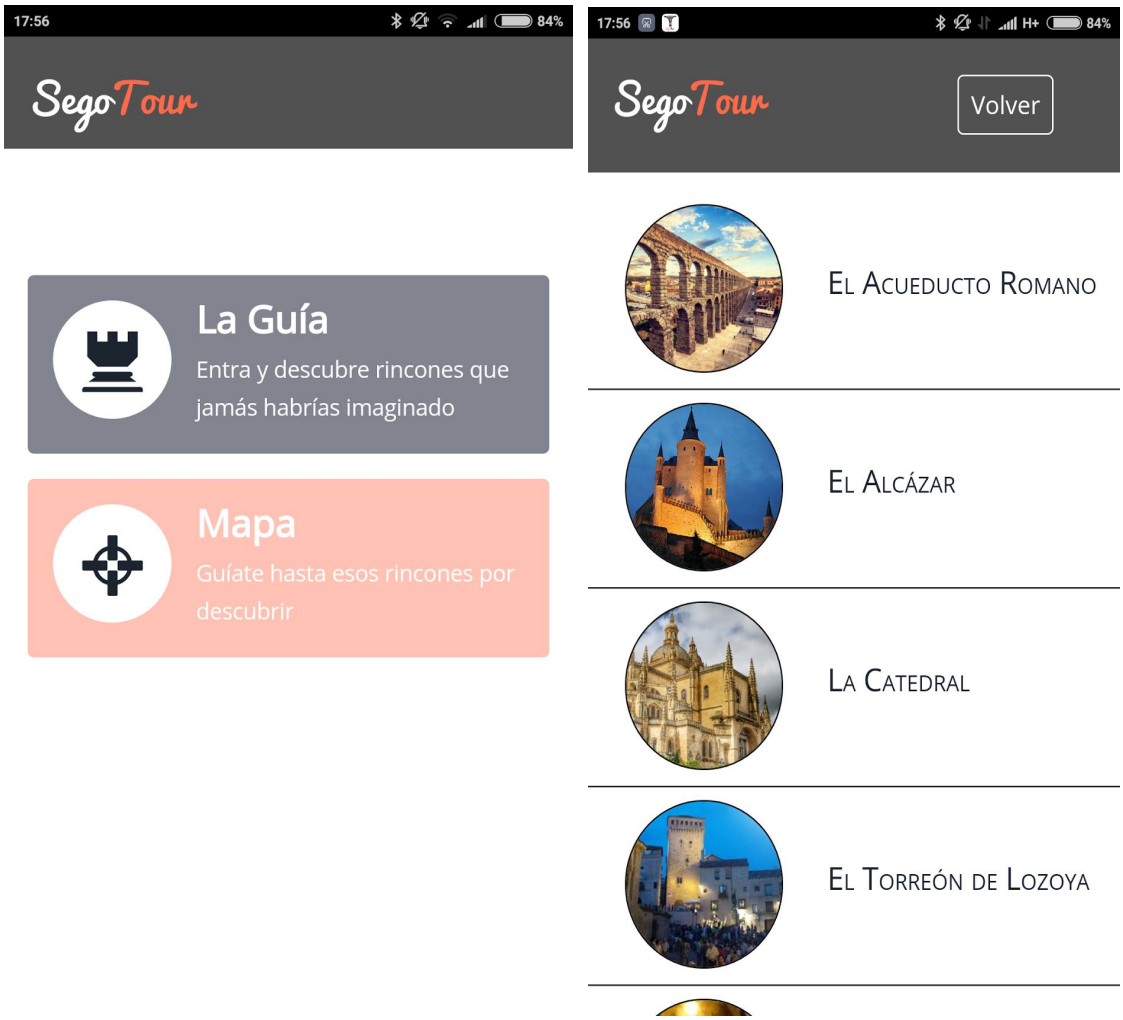


Figura 6.2 Manual de usuario : Ver listado de monumentos

Se podrá acceder al listado de monumentos desde la pantalla principal en el botón “La Guía”. Entonces se visualizará la siguiente pantalla, que indica que se puede seleccionar una foto desde la cámara o desde la galería:

❖ Ver información de monumento

Figura 6.3 Manual de usuario: pantalla monumento

Para llegar a esta sección basta con pulsar cualquier monumento en la lista de monumentos visualizada anteriormente.

6.2.2 Gestión de mapas

- Ver mapa

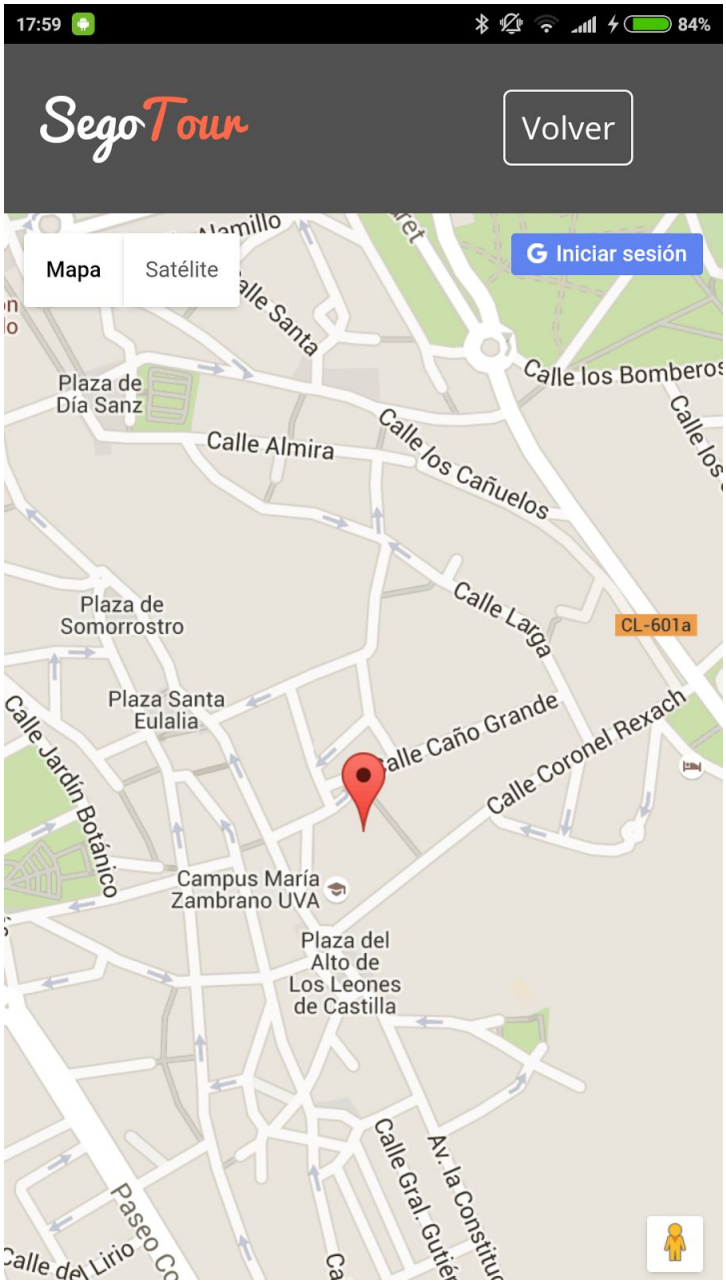


Figura 6.4 Manual de usuario: pantalla mapa

Pulsando en la pantalla inicial el botón "Mapa", accedemos a un mapa que localiza nuestra posición.

6.2.3 Gestión de notificaciones

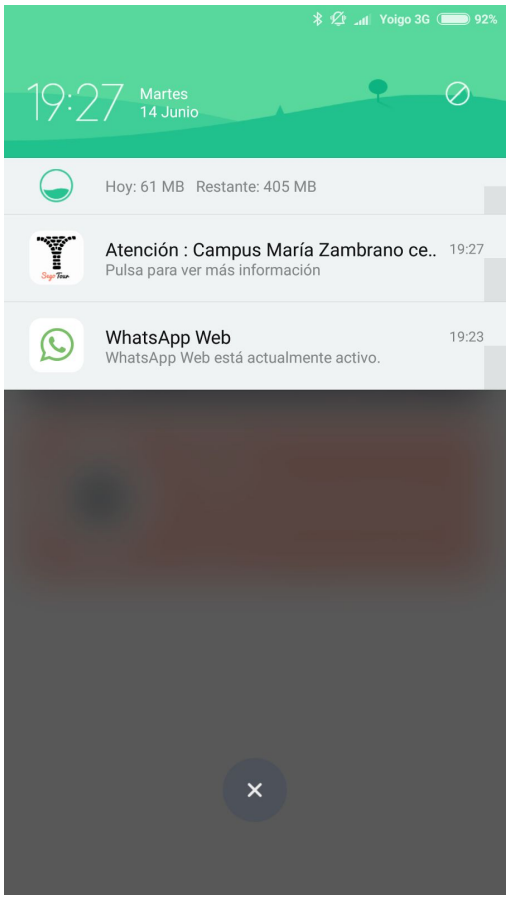


Figura 6.5 Manual de usuario: pantalla notificación

Al recibir una notificación, en la zona de notificación del dispositivo recibiremos una alerta como la de la figura mostrada. Una vez se pulse, nos llevará a la pantalla de monumento en cuestión mostrada anteriormente.

CAPÍTULO 7

FUTURAS MEJORAS Y CONCLUSIONES

7.1 Futuras mejoras

Una mejora podría consistir en ampliar el producto que ahora mismo ofrece SegoTour. Además de ser una simple guía de monumentos, añadir también restaurantes, con ofertas actualizadas a diario por los propios comerciantes.

También sería interesante crear una sección con una guía del programa cultural que ofrece la ciudad de Segovia. Es decir, incluir información actualizada sobre festivales, como Titirimundi, incluir museos e incluso fiestas tradicionales como San Frutos.

Así mismo, sería interesante crear la aplicación en el sistema operativo iOS, ya que en dicha plataforma existen bastantes usuarios potenciales.

Sería de vital importancia incluir otros idiomas como el inglés, francés o chino, ya que la gran mayoría de turistas en Segovia son principalmente extranjeros. Se propone además la inclusión de vídeos descriptivos sobre los monumentos y también una guía auditiva para ciegos.

En definitiva es una aplicación que tiene posibilidades para expandirse a otras opciones.

7.2 Conclusiones personales

Desde que comencé a cursar la asignatura de Plataformas Software Móviles me llamó mucho la atención enfocar mi vida a la tecnología móvil. Fue con las prácticas en empresa y con el posterior trabajo cuando me di cuenta de que usando tecnología híbrida se podía sacar mucho con muy poco esfuerzo.

Cuando empecé la carrera tuve claro que quería crear algo que fuera "mío", y dado que una de las ofertas de la universidad era la creación de una aplicación móvil para Segovia supe que quería ese proyecto para mí. Aparte de ampliar mi conocimiento sobre la ciudad, también me ha ayudado a decidir hacia donde quería enfocar mi carrera profesional.

Hubo malos ratos en los que no salía nada, momentos en los que el agobio y la impotencia hicieron mella en el avance, pero con esfuerzo y sobre todo con mucha ayuda por parte de foros en internet, dí con la solución.

También debo destacar la ayuda de las personas que han hecho que el proyecto salga adelante, empezando por los tutores. Tanto Jesús con sus "posibles problemas" como con Diego aportando soluciones a esos problemas. Ambos me han ayudado a tener una imagen mucho más amplia de la que tenía y me han ayudado a mejorar tanto técnicas de programación como manera de hacer las cosas. Agradecer también a dos antiguos compañeros de trabajo las horas dedicadas a sacarme de dudas y a aportar soluciones.

Han sido varios meses y al final todo llega, por fin he terminado mi trabajo fin de grado completamente satisfecho, tanto por haberlo acabado como por lo que he aprendido a lo largo del proceso.

No sé si esta aplicación tendrá una gran aceptación entre las personas en general, pero yo me siento muy feliz por haberla creado.

7.3 Conclusiones profesionales

Este trabajo me ha ayudado sobre todo a formarme más como profesional y a poder profundizar más en conocimientos que no se imparten en la carrera.

Partiendo de la base de la carrera con desarrollo web (HTML, CSS y JavaScript) y de los conocimientos de aplicaciones móviles (Android), he podido juntar ambas vías de desarrollo y desarrollar una aplicación en un mercado que está creciendo exponencialmente. Utilizando tecnologías como Apache Cordova o NodeJS siento que estoy más preparado para lo que me encuentre el día de mañana. Ha sido prácticamente empezar de cero con cada nuevo lenguaje, y me siento muy orgulloso de haber podido sacar esto adelante.

También, modelar todo el sistema desde el análisis a la implementación ha hecho que tenga una visión mucho más amplia y detallada del sistema, ayudándome a construir una buena base en todo el ciclo de ingeniería de requisitos y a saber enfrentrarme a problemas similares en el futuro.

En cuanto a la metodología de trabajo, haber elegido una metodología ágil me ha ayudado a aumentar currículum, ya que hoy día todas las empresas (o la gran mayoría) desarrollan software ágil.

Y lo que más me ha enseñado este proyecto es que si trabajas duro puedes conseguir cualquier cosa que te propongas.

CAPÍTULO 8

REFERENCIAS

8.1 Bibliografía

K. Wiegers, J.Beatty. “Software Requirements”. Microsoft Press, 3rd edition, 2013.

8.2 Referencias web

- Diseño de interfaz
    - http://getbootstrap.com/ [Última consulta 22/05/2016]
- Funcionalidad
    - https://cordova.apache.org/ [Última consulta 06/06/2016]
    - https://developers.google.com/maps/get-started/?hl=es [Última consulta 01/05/2016]
    - http://www.w3schools.com/html/html5_geolocation.asp [Última consulta 03/05/2016]
    - http://stackoverflow.com/questions/16222330/geolocation-moving-only-google-maps-marker-without-reload-the-map [Última consulta 01/05/2016]
    - http://stackoverflow.com/questions/16192186/google-maps-api-watchposition [Última consulta 01/05/2016]
    - http://stackoverflow.com/questions/21645329/googlemaps-api-follow-and-center-my-watchpostion-without-reloading-the-map [Última consulta 01/05/2016]
    - http://www.coordenadas-gps.com/ [Última consulta 04/06/2016]
    - https://github.com/phonegap/phonegap-plugin-push/blob/master/docs/API.md [Última consulta 23/05/2016]
    - https://github.com/katzer/cordova-plugin-local-notifications [Última consulta 23/05/2016]
    - http://www.dbsnippets.com/2013/03/12/android-permisos-sobre-la-localizacion/ [Última consulta 27/04/2016]
    - https://github.com/apache/cordova-plugin-dialogs/blob/master/doc/es/index.md [Última consulta 15/05/2016]

- **Contenido y descripciones**
    - http://casamonedasegovia.es/historia/casa-de-moneda/ [Última consulta 26/05/2016]
    - https://es.wikipedia.org/ [Última consulta 26/05/2016]

- http://www.turismoespanagps.com/puntosinteres.php?id=816 [Última consulta 26/05/2016]
- http://triplenlace.com/2012/09/16/la-casa-de-la-quimica-de-segovia-donde-enseno-proust/ [Última consulta 26/05/2016]

- **Memoria**
  - https://es.wikipedia.org/wiki/Android [Última consulta 29/05/2016]
  - https://blog.uchceu.es/informatica/ranking-de-sistemas-operativos-mas-usados-para-2015/ [Última consulta 29/05/2016]
  - http://www.androidcurso.com/index.php/tutoriales-android/31-unidad-1-vision-general-y-entorno-de-desarrollo/146-las-versiones-de-android-y-niveles-de-api [Última consulta 29/05/2016]
  - http://www.elandroidelibre.com/2016/01/informe-android-enero-2016.html [Última consulta 29/05/2016]
  - http://www.accensit.com/index.php/en/accensit-blog-en/150-mobile-platforms.html [Última consulta 29/05/2016]
  - https://proyectosagiles.org/beneficios-de-scrum/ [Última consulta 29/05/2016]
  - https://proyectosagiles.org/que-es-scrum/ [Última consulta 29/05/2016]
  - http://www.aplicandoscrum.com/rol-perfil-product-owner/ [Última consulta 03/06/2016]
  - https://es.wikipedia.org/wiki/Retorno_de_la_inversi%C3%B3n [Última consulta 03/06/2016]
  - http://www.cristalab.com/blog/que-significa-backend-y-frontend-en-el-diseno-web-c106224l/ [Última consulta 03/06/2016]
  - http://finanzas-personales.practicopedia.lainformacion.com/sueldo/como-es-el-coste-real-de-un-trabajador-para-una-empresa-20064 [Última consulta 03/06/2016]

Anexo I

CONTENIDO DEL CD-ROM

El CD-ROM tiene la siguiente estructura:

- **Software**: Contiene el .apk para instalar la aplicación en un dispositivo móvil. También contiene el código de la aplicación.

- **Documentación**: Contiene la versión digital de esta memoria en .pdf.