Universidad de Valladolid

ESCUELA DE INGENIERÍA INFORMÁTICA DE SEGOVIA

Grado en Ingeniería Informática de Servicios y Aplicaciones

BlueButtonApp: Aplicación de ayuda a la dependencia

Alumno: Víctor Rujas Calvo
Tutores: Aníbal Bregón Bregón
Miguel Ángel Martínez Prieto

Agradecimientos

Gracias a mis profesores a todos y cada uno de ellos, ya que de una manera u otra han influido en mí, muy en especial a los que más me hayan sufrido.A Aníbal y a Miguel Ángel como tutores de este proyecto, que han sufrido mis agobios tanto o más que yo y con los que siempre tenía nuestras bromas en clase, ya sabéis con que...

Agradecer a mi familia y a mis amigos y a todos los que me habéis apoyado, que estos últimos días me han visto poco, y que lo poco que me han visto ha sido con mi cara de 'borde' como dirían ellos.

Definitivamente, gracias a todos los que hayan influido en mí de alguna manera, directa o indirecta, ya que si no fuera por eso yo no estaría aquí, y mucho menos habría llegado a conseguir esto.

Resumen

Es un hecho que actualmente un gran porcentaje de personas en la sociedad requieren de una atención continuada por múltiples causas: enfermedad, edad, etc. Esas causas crean una dependencia a dichas personas que les impide llevar una vida normal.

Enfocado en esa dependencia, se desarrolla una aplicación móvil multiplataforma que permita a esa persona estar localizable y a su vez sentirse autónoma ya que le permite gestionar su medicación (en el caso de que la necesitase) con avisos y alarmas. También da acceso rápido a un aviso de emergencia, en caso en que fuera necesario, a la persona que este a cargo de la misma. La aplicación por su cuenta monitoriza el teléfono continuamente para detectar posibles accidentes o caídas del usuario con el dispositivo encima, y avisar de las mismas. Una funcionalidad extra que proporciona la aplicación es enviar automáticamente la posición del usuario cada quince minutos. La persona a cargo podrá acceder a una aplicación web y así visualizar la posición del usuario de la aplicación móvil.

La aplicación ha sido desarrollada mediante Phonegap lo que permite que la aplicación móvil fuese híbrida multiplataforma. En Phonegap se ha programado con la ayuda de ReactJS, que permite crear componentes interactivos y reutilizables de interfaces de usuario, y para la parte visual se ha basado en TouchstoneJS que es un framework que genera la parte gráfica de la interfaz. A la hora de acceder a los sensores del teléfono, así como el GPS o el acelerómetro, o implementar el resto de funcionalidades de la aplicación se han utilizados diferentes librerías o plugins de Phonegap que permiten el acceso a la parte nativa del dispositivo mediante JavaScript.

El proyecto ha sido pensado y creado con la principal y única finalidad de conseguir mejorar la vida de aquellas personas que tengan algún tipo de dependencia y a su vez dotarles de autonomía, lo que se traduce en tranquilidad y calidad de vida para esas personas y sus familiares.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Motivación

En la actualidad existe un gran número de personas que requieren de una atención continuada a causa de enfermedades, de la edad o de ambas. Dichas personas tienen una dependencia, que les impide realizar una vida normal de forma autónoma.

La Real Academia Española de la lengua define *dependencia*[1] como " situación de una persona que no puede valerse por sí misma ". Esa dependencia es un gran problema para dichas personas y para sus familias, por lo que aprovechando la implantación, cada vez mayor, de numerosas y novedosas tecnologías en la vida cotidiana, se afronta de distinta manera.

La Cruz Roja asiste a numerosas personas en dicha situación mediante un servicio de Teleasistencia, que les permite avisar de cualquier situación de emergencia en su hogar inmediatamente ya que se encuentra conectado directamente con la línea telefónica. Es un servicio preventivo de ayuda a personas que por motivos de discapacidad, aislamiento social, edad avanzada, enfermedad, o situación de riesgo psicosocial o físico, se encuentran en una situación de dependencia. Ofrece disponibilidad total, 24 horas al día los 365 días del año y tiene la capacidad de movilizar los recursos necesarios para dar solución a cualquier tipo de situación de emergencia que hubiese sucedido.

Buscando complementar e intentar mejorar la ayuda que se pueda ofrecer a dichas personas, se ha afrontado la realización de una aplicación móvil que permita realizar unas funciones similares a las de la Teleasistencia ofrecida por la Cruz Roja y a su vez añadir otras que mejoren su calidad de vida y la de sus familiares. En la aplicación que se desea crear un ejemplo de función añadida podría ser la detección automática de caídas por parte del dispositivo.

1.2. Objetivos del proyecto

El objetivo principal del TFG es desarrollar una aplicación móvil para dotar de mayor independencia a las personas que puedan sufrir algún tipo de dependencia, discapacidad o enfermedad, permitiéndole a su vez estar más controladas en todo momento. Para ello

[1](http://dle.rae.es/?id=CEjjsLO)

se toma como referencia la aplicación de la Teleasistencia de la Cruz Roja[2] mejorándola y adaptándola para mejorar la calidad de vida de los futuros usuarios. El cumplimiento de dichos objetivos viene ligado con los siguientes subobjetivos:

- La gestión de la posición GPS del usuario ofreciendo una monitorización continua del usuario.

- La detección de posibles caídas del usuario, lo que pondrá en alerta la aplicación y podrá realizar algún aviso a la persona de contacto del usuario o a urgencias si así se requiriese.

- Por último, para dotar de mayor independencia se incorporará un "pastillero"que gestionará mediante recordatorios las horas en las que el usuario debiera tomarse su medicación en el caso de que lo necesitase.

1.3. Características principales del sistema

La aplicación pretende controlar y dotar de independencia a los usuarios y familiares de los mismos mejorando su calidad de vida. Con esta aplicación el usuario configurando adecuadamente su dispositivo puede:

- Gestionar contactos: se podrá agregar, eliminar y/o visualizar contactos a los que retransmitir en todo momento lo relativo al usuario dependiente.

- Gestión de posición GPS: nos da la posibilidad de obtener la localización GPS del usuario en el momento, asignar una zona segura para el usuario, y modificar dicha zona de control para que nos permita saber cuando la abandona por ejemplo.

- Detectar caídas: permite activar o desactivar el servicio, que nos permitirá avisar automáticamente a su/s contacto/s si ha sufrido alguna caída en caso de que esta esté activada.

- Gestionar medicación: el usuario podrá gestionar su medicación en cada momento añadiendo recordatorios o alertas, modificándolos en el caso de que el mismo haya cambiado, y por último eliminándolos en el caso de el tratamiento haya terminado.

A continuación, se describen las características de estos componentes a través de un árbol de características:

[2]http://www.cruzroja.es/principal/web/teleasistencia

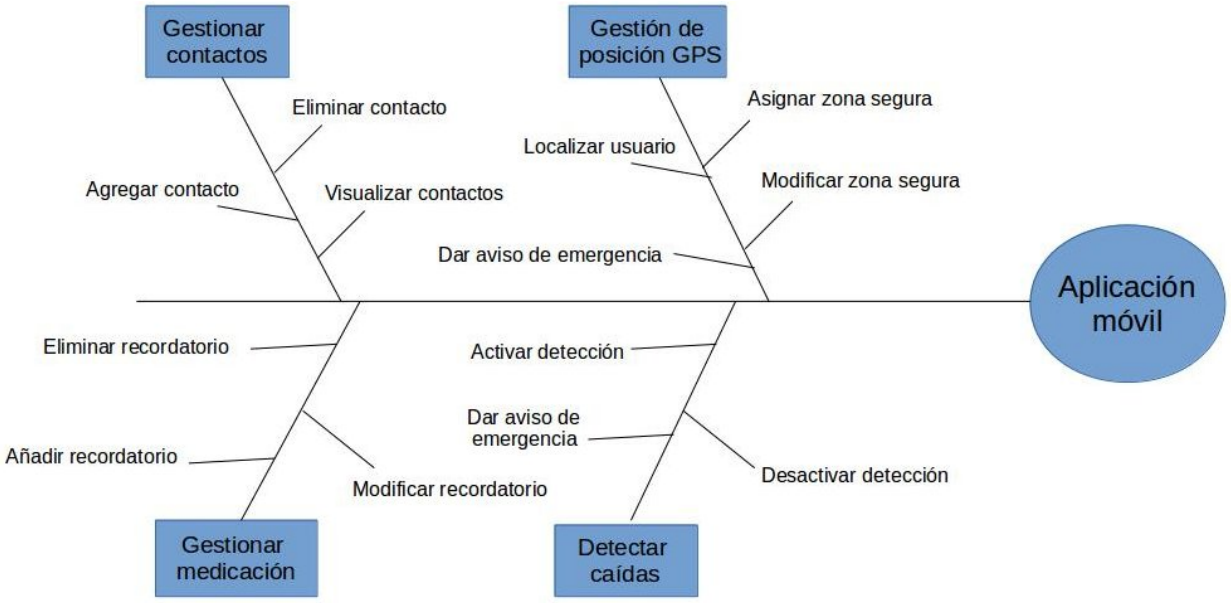


Figura 1.1: Árbol de Características.

1.4. Estructura del documento y del CD-ROM

A continuación se realiza una enumeración de los distintos capítulos del documento y una breve explicación de los contenidos incluidos dentro de ellos. El documento en sí esta formado por nueve capítulos, los cuales a su vez están divididos en varias secciones organizando el contenido para facilitar su entendimiento.

Capítulo 1, Introducción . Es un capítulo introductorio que plantea el contexto relacionado con el proyecto. Sus capítulos afrontan la motivación, los objetivos, las características del proyecto en sí,, incluyendo la sección actual sobre la estructura del documento.

Capítulo 2, Estado del Arte . Se ejecuta en estudio de los múltiples sistemas del mercado que se engloban dentro del nicho de mercado del sistema a desarrollar. Lo que permite definir y valorar las funcionalidades a desarrollar.

Capítulo 3, Plan de proyecto . Se plantean estimaciones de tipo económico, temporal, y se muestra la conclusión sobre la metodología a seguir durante el proyecto. Se incorpora un presupuesto inicial y el coste real para evaluar los resultados.

Capítulo 4, Análisis . Se realiza un análisis desde el punto de vista de usuario, de sistema y desde el punto de los datos manejados.Es uno de los capítulos más importante del documento general.

Capítulo 5, Diseño . En este capítulo se especifica como desarrollar el sistema, y a su vez como debería estructurarse y comportarse. Todo lo aquí definido se tendrá muy en cuenta en el siguiente capítulo.

Capítulo 6, Implementación . A la hora de implementar el sistema se usan un conjunto de tecnologías que se describen en este apartado y las diferentes herramientas usadas para el desarrollo del sistema.

Capítulo 7, Pruebas . Se incluyen pruebas realizadas durante la implementación del sistema, y también las realizadas a posteriori, se corresponden con pruebas de caja blanca y caja negra respectivamente.

Capítulo 8, Manuales . En este capítulo se deja constancia de la instalación de ambos sistemas en un manual de instalación y configuración y a su vez también un manual de usuario de las aplicaciones resultantes del proyecto.

Capítulo 9, Conclusiones y líneas futuras . Una vez realizado el nuevo sistema se llega a unas conclusiones y se crean expectativas sobre mejoras posibles en una posterior versión de la aplicación que se exponen finalmente en este capítulo.
El documento termina con los anexos, uno trata sobre la creación de aplicaciones móviles híbridas mediante el uso de una de las herramientas utilizadas en el proyecto, para mayor entendimiento del usuario si así lo requiriese.

CD-ROM : el CD-ROM contiene dos carpetas con el código fuente de los sistemas desarrollados: sistema web(**BlueButtonWeb**) y sistema móvil(**BlueButtonApp**, el archivo .apk para ejecutar la aplicación móvil(blueButtonApp.apk), y el pdf de la memoria del dicho TFG.

Capítulo 2

Estado del Arte

Existe un alto número de aplicaciones que ofrezcan las mismas funcionalidades que las que ofrece la Cruz Roja en su Teleasistencia y cada una a su manera pero ninguna afronta facilitar aún mas la vida de los usuarios, persona con dependencia y/o familiares. A partir de ese punto se ha realizado la aplicación. Se busca ser complementario con la Teleasistencia de la Cruz Roja[1] y seguir su testigo en situaciones fuera de su propio hogar dónde esta no puede brindarnos su supervisión.

Con una intención similar a la de este TFG existen varias aplicaciones desarrolladas para smartphones orientadas a distintos sistemas operativos móviles y con diferencias notables. Las aplicaciones que se muestran a continuación son similares a la desarrollada en este trabajo.

[1] http://www.cruzroja.es/principal/web/teleasistencia

ElbotónrojoApp .Es una aplicación de teleasistencia móvil, que desde 8,95 €/mes permite con solo pulsar un botón asistencia inmediata de tipo sanitario, policía, bomberos, etc.. Además permite a un familiar vinculado con el usuario saber en todo momento la ubicación del usuario a través de la vinculación con el GPS. Es un aplicación limitada a la localización GPS y la realización de avisos, únicamente disponible para Android e iOs.

Figura 2.1: El Botón Rojo App.

TeleAsistenci@TIC+ . Es una aplicación gratuita y sin publicidad, que ofrece una serie de funcionalidades complementarias a los servicios de Teleasistencia. Permite alertar rápidamente a los contactos mediante un SMS pulsando un botón, permite la geolocalización mediante GPS, y un sistema de detección de caídas experimental que envía un mensaje en caso de detección. La única plataforma disponible es Android y en fase Beta.

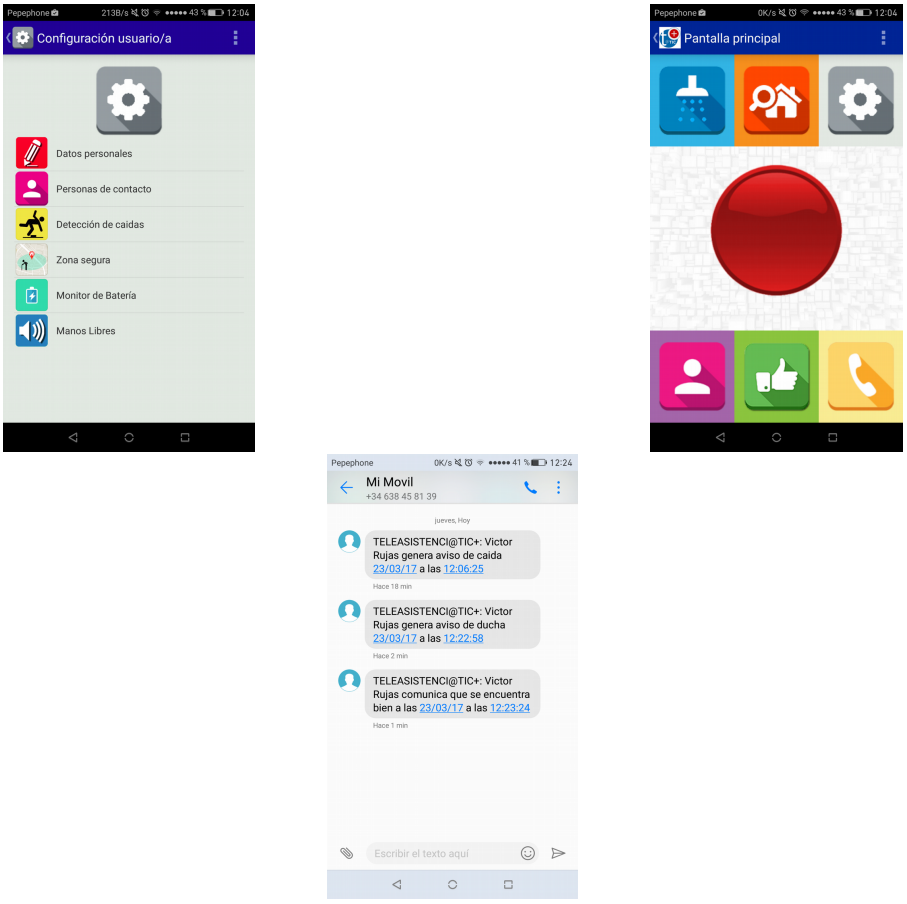


Figura 2.2: TeleAsistenci@TIC+

Tweri . En una solución de movilidad orientada para las personas con Alzheimer en las primeras etapas de la enfermedad. Permite geolocalizar a las personas en todo momento e informar a un familiar de contacto, y facilitando la navegación hacia su propio domicilio en caso de que el usuario se sienta desorientado. Es una aplicación gratuita, disponible para iOS y Android.

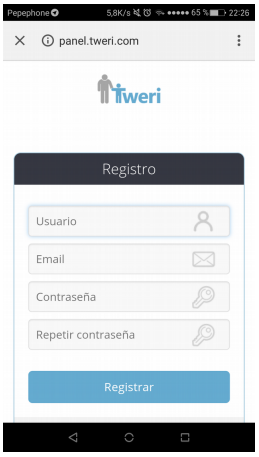


Figura 2.3: Tweri

MIMOV . Es una tecnología que incluye dispositivo móvil MIMOV que usa cualquier tarjeta SIM para el usuario monitorizándolo, y un servicio que se accede y configura desde una página web o app para un smartphone(Android e iPhone).

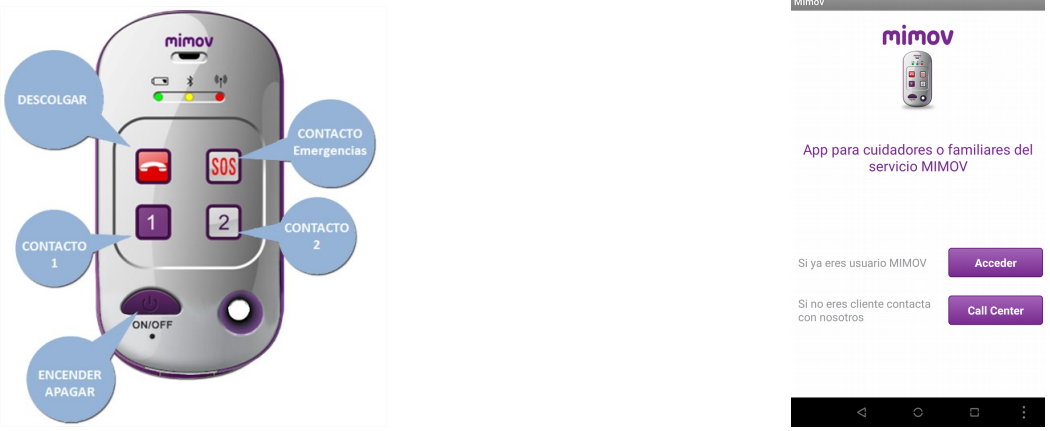


Figura 2.4: MIMOV

Anteriormente se enumeran y caracterizan brevemente las alternativas más similares y que mayor competencia podrían tener en el mercado frente al sistema a desarrollar, **BlueButton**. Para realizar una comparativa más fácil e intuitiva se añade la siguiente tabla que muestra los factores principales del tipo de aplicaciones que se tiene en cuenta:

Aplicaciones	Conexión a Internet	Interfaz e intuición	Ubicación por GPS	Detección de caídas	Gestionar medicamentos
ElBotonRojoApp	Si	Buena	Si	No	No
TeleAsistenci@TIC+	No	Mala	Si	Si	No
Tweri	Si	Buena	Si	No	No
MIMOV	No	Mala	Si	No	No
BlueButton	Si	Buena	Si	Si	Si

Tabla 2.1: Comparativa de aplicaciones.

Como se puede observar, todas las aplicaciones tienen varios puntos fuertes, aunque ninguna consigue ser completa, siempre suele tener alguna desventaja en ese sentido frente a **BlueButton**, tanto como en interfaz simple y agradable a la vista, así como en tener unas funcionalidades u otras. **BlueButton** es una alternativa completa que engloba las características más importantes de las otras aplicaciones del mercado.

Capítulo 3

Plan de proyecto

El capítulo que viene a continuación numera y expone los diferentes puntos relativos a la planificación del proyecto, así como la metodología a seguir, la planificación temporal y la estimación presupuestaria sobre el desarrollo.

3.1. Metodología

A la hora de elegir consecuentemente la metodología a utilizar en el proyecto se ha optado por un modelo de tipo incremental, ya que el proyecto comienza con unas funcionalidades a desarrollar que irán aumentando y evolucionando a medida que el proyecto vaya progresando. Por lo que se realizarán varias iteraciones, compuestas cada una por distintas fases. Serán cinco iteraciones cada una compuesta por cinco fases distintas. Las fases son las siguientes:

- Análisis.
- Diseño.
- Implementación.
- Pruebas.
- Documentación.

Las iteraciones tendrán fases comunes, a excepción de la primera, que no tendrá la fase de documentación en cuenta, pero sí una fase dedicada a la planificación.

Este tipo de modelo suele ser el más utilizado en los proyectos de software ya que es muy completo, permite tener un prototipo funcional al terminar la primera iteración y seguir mejorándolo añadiendo y puliendo funcionalidades en cada iteración. Lo que permite obtener un prototipo bastante conseguido al final del plan del proyecto.

3.2. Fases de trabajo y estimación temporal

La planificación inicial del trabajo estaba pensada en un plazo aproximado de cinco meses dada la previa adquisición de conocimientos relativos al uso de tecnologías como *Phonegap* o

ReactJS, que serían indispensables para la implementación del proyecto, y al reducido tiempo disponible a causa la situación laboral durante todo el año.

Basado en el modelo incremental se llevarán a cabo cinco iteraciones, o incrementos, en los que se tendrá en cuenta que durante los primeros incrementos se invertirá más tiempo en las fases relativas al análisis y al diseño, mientras que a partir del tercer incremento se invertirá para las fases de implementación y pruebas, y de la documentación del proyecto, por supuesto sin descuidar el análisis o el diseño.

En la siguiente figura se muestra detalladamente la planificación inicial de tiempos del proyecto en cuestión, teniendo en cuenta todo lo citado anteriormente, de manera estimada.

	Nombre	Duración	Inicio	Fin	Predecesoras
1	**Iteracción 1**	**23d**	**09/02/2017**	**13/03/2017**	
2	Planificación	5d	09/02/2017	15/02/2017	
3	Análisis	6d	16/02/2017	23/02/2017	2
4	Diseño	5d	24/02/2017	02/03/2017	3
5	Implementación	5d	03/03/2017	09/03/2017	4
6	Pruebas	2d	10/03/2017	13/03/2017	5
7	**Iteración 2**	**19d**	**14/03/2017**	**07/04/2017**	**1**
8	Análisis	6d	14/03/2017	21/03/2017	
9	Diseño	5d	22/03/2017	28/03/2017	8
10	Implementación	4d	29/03/2017	03/04/2017	9
11	Pruebas	2d	04/04/2017	05/04/2017	10
12	Documentación	2d	06/04/2017	07/04/2017	11
13	**Iteración 3**	**18d**	**10/04/2017**	**03/05/2017**	**7**
14	Análisis	3d	10/04/2017	12/04/2017	
15	Diseño	2d	13/04/2017	14/04/2017	14
16	Implementación	7d	17/04/2017	25/04/2017	15
17	Pruebas	3d	26/04/2017	28/04/2017	16
18	Documentación	3d	01/05/2017	03/05/2017	17
19	**Iteración 4**	**21d**	**04/05/2017**	**01/06/2017**	**13**
20	Análisis	2d	04/05/2017	05/05/2017	
21	Diseño	2d	08/05/2017	09/05/2017	20
22	Implementación	7d	10/05/2017	18/05/2017	21
23	Pruebas	5d	19/05/2017	25/05/2017	22
24	Documentación	5d	26/05/2017	01/06/2017	23
25	**Iteracción 5**	**23d**	**02/06/2017**	**04/07/2017**	**19**
26	Análsis	3d	02/06/2017	06/06/2017	
27	Diseño	2d	07/06/2017	08/06/2017	26
28	Implementación	4d	09/06/2017	14/06/2017	27
29	Pruebas	7d	15/06/2017	23/06/2017	28
30	Documentación	7d	26/06/2017	04/07/2017	29

Figura 3.1: Planificación temporal de las iteraciones.

Ligado a la planificación de las iteraciones se realiza un diagrama de Gantt, que representa de manera gráfica la distribución de dichos intervalos a lo largo del tiempo.

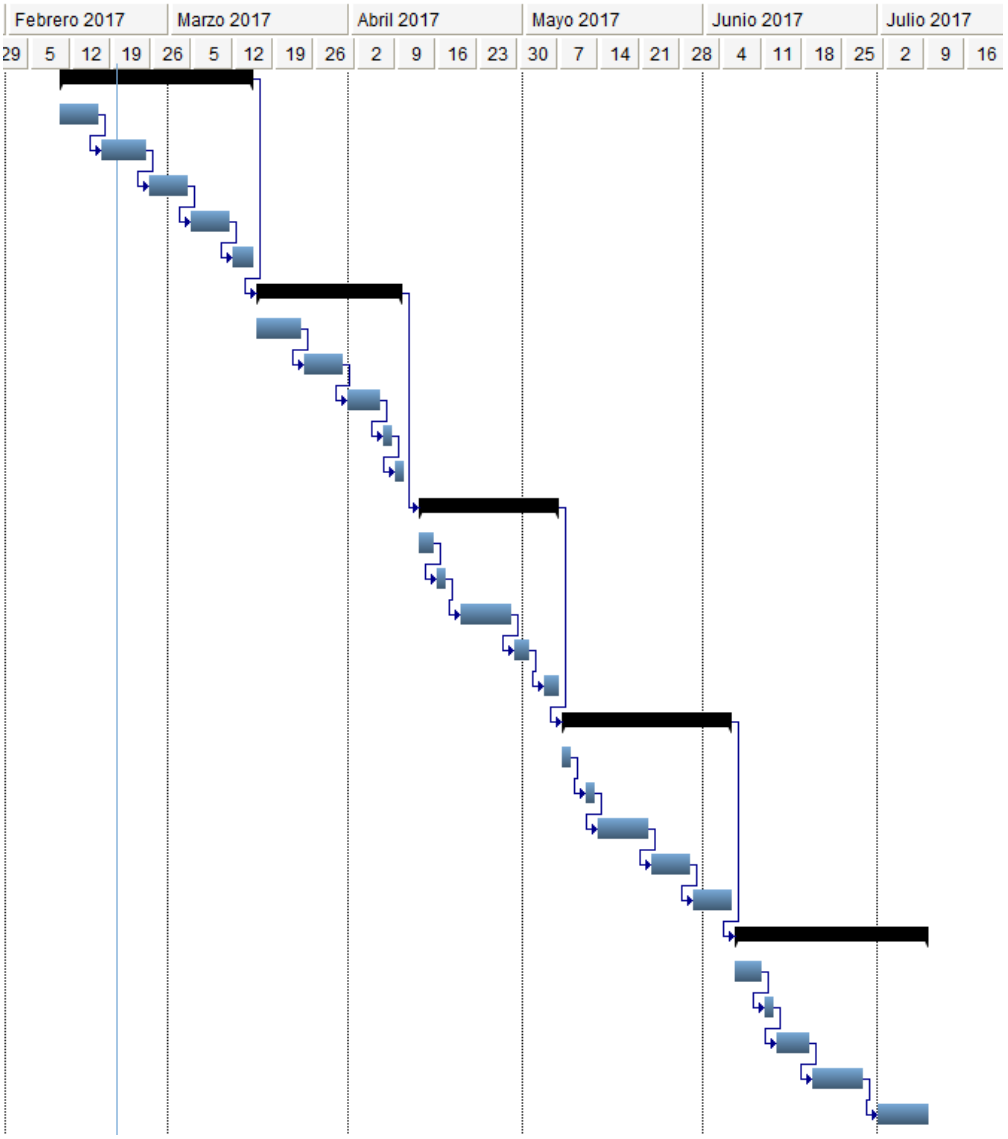


Figura 3.2: Diagrama de Gantt del proyecto.

3.3. Estimaciones

Para realizar un cálculo correcto del presupuesto es necesario realizar las estimaciones oportunas. En este caso utilizaré dos tipos de modelos: el modelo de Estimación por Puntos de Función (**PF**) y el Modelo Constructivo de Costos (**COCOMO**).

Estimación por Puntos de Función

Este método se basa en una métrica que cuantifica la funcionalidad que se debe entregar al usuario al construir la aplicación. Como el proyecto se compone de dos aplicaciones, la

móvil y la aplicación web, por lo que el procedimiento debería realizarse dos veces.

El proceso deberá seguir los siguientes pasos:

- Los parámetros que se utilizan para la evaluación de esa funcionalidad son:
  - Número de entradas: aquellos datos que el usuario aporta al sistema..
  - Número de salidas: los datos que el sistema aporta al usuario,
  - Número de ficheros lógicos internos: son ficheros o bases de datos internos al sistema.
  - Numero 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.
- Se contabiliza el número de elementos de cada clase y su complejidad en relación a tres niveles.
- Por último se obtiene los Puntos de Función No Ajustados (**PFNA)**) mediante una suma ponderada de esas cantidades con los pesos que aparecen en la siguiente tabla:

Parámetro	**Complejidad**		
	Baja	**Media**	**Alta**
Entradas	x3	x4	x6
Salidas	x4	x5	x7
Ficheros internos	x7	x10	x15
Ficheros externos	x5	x7	x10
Consultas externas	x3	x7	x6

Tabla 3.1: Pesos de los dominios de información según su complejidad

- En el momento en el que se obtienen PFNA, se deben ajustar mediante un factor de ajuste (FA). Dicho factor se obtiene media la suma de 14 factores de complejidad diferentes.

$$FA = (0,01 \times \sum FC) + 0,65$$

- A partir de aquí se obtienen los puntos de función ajustados (PF).

$$PF = FA \times PFNA$$

- Por último hay que obtener una aproximación de las lineas de código que hay en cada lenguaje, para estimar los puntos de función equivalentes. Aproximadamente esta es la relación entre las líneas de código que equivale un punto de función según los lenguajes usados en proyecto.

Lenguaje de Programación	LDC/PF
HTML5	34
JavaScript	47
J2EE	46
Java	53

Tabla 3.2: Equivalencia entre líneas de código(LDC) y puntos de función (PF).

Este sería el proceso es cuestión, a continuación se harán dos estimaciones teniendo en cuenta una relativa a la aplicación móvil híbrida y otra a la aplicación web.

Aplicación móvil híbrida

- Número de entradas:
  - Formulario de login: complejidad baja.
  - Formulario de creación de medicamento: complejidad media.
  - Datos de usuario de contacto: complejidad baja.
  - Menú de navegación: complejidad media.
- Número de salidas:
  - Mostrar siguiente alarma: complejidad baja.
  - Listado de prescripciones: complejidad media.
  - Mensajes de error: complejidad baja.
  - Notificaciones push: complejidad media.
  - Despertar aplicación por caída de emergencia: complejidad media.
- Número de ficheros lógicos internos:
  - Base de datos: complejidad media.
- Numero de ficheros externos:
  - Base de datos: complejidad media.
- Número de consultas externas:
  - Consulta de usuarios: complejidad media.
  - Envío de posición: complejidad media.

Parámetro significativo	Complejidad	Total x Complejidad	Total
Entradas	Baja Media Alta	3 x 3 1 x 4 0 x 6	13
Salidas	Baja Media Alta	2 x 4 3 x 5 0 x 7	23
Ficheros internos	Baja Media Alta	0 x 7 1 x 10 0 x 15	10
Fichero externos	Baja Media Alta	0 x 5 1 x 7 0 x 10	7
Consultas externas	Baja Media Alta	0 x 3 2 x 7 0 x 10	14
Total de Puntos de Función (PFNA)			**67**

Tabla 3.3: PFNA para la aplicación móvil

Factor de complejidad	Complejidad/Influencia
Comunicación de datos	3
Rendimiento	2
Funciones distribuidas	0
Gran carga de trabajo	3
Frecuencia de transacciones	2
Entrada on-line de datos	1
Requisito de manejo del usuario final	1
Actualizaciones on-line	2
Procesos complejos	1
Utilización de otros sistemas	2
Facilidad de mantenimiento	1
Facilidad de operación	1
Instalación en múltiples lugares	2
Facilidad de cambio	1

Tabla 3.4: Factores de complejidad aplicación móvil híbrida.

$$FA = (0,01 \times 22) + 0,65 = 0,87$$

$$PF = PFNA \times FA = 67 \times 0,87 = 58,29$$

Para calcular las líneas de código de la aplicación, es necesario una media de las equivalencias de los lenguajes de programación JavaScript y HTML5 ya que todos serán necesarios para la implementación y no se sabe la proporción exacta de cada uno en el proyecto. Lo que resulta en una media de 1 PF por cada 40,5 LDC.

$$LDC = 40,5 \times 58,29 = 2360,745 \approx 2,4KLDC$$

Aplicación web

- Número de entradas:
  - Formulario de login: complejidad baja.
  - Formulario de registro: complejidad baja.
  - Formulario de actualización de datos:complejidad baja.
  - Datos de usuario de contacto: complejidad baja.
  - Menú de navegación: complejidad baja.
- Número de salidas:
  - Listado de recursos: complejidad baja.
  - Listado de datos: complejidad baja.
  - Mensajes de error: complejidad baja.
  - Mostrar recursos: complejidad media.
- Número de ficheros lógicos internos:
  - Base de datos: complejidad media.
- Numero de ficheros externos:
  - Base de datos: complejidad media.
- Número de consultas externas:
  - Consulta de usuarios: complejidad baja.
  - Consulta de posición: complejidad media.

Parámetro significativo	Complejidad	Total x Complejidad	Total
Entradas	Baja Media Alta	5 x 3 0 x 4 0 x 6	15
Salidas	Baja Media Alta	3 x 4 1 x 5 0 x 7	17
Ficheros internos	Baja Media Alta	0 x 7 1 x 10 0 x 15	10
Fichero externos	Baja Media Alta	0 x 5 1 x 7 0 x 10	7
Consultas externas	Baja Media Alta	1 x 3 1 x 7 0 x 10	10
Total de Puntos de Función (PFNA)			**59**

Tabla 3.5: PFNA para la aplicación web

Factor de complejidad	Complejidad/Influencia
Comunicación de datos	2
Rendimiento	3
Funciones distribuidas	0
Gran carga de trabajo	2
Frecuencia de transacciones	3
Entrada on-line de datos	2
Requisito de manejo del usuario final	1
Actualizaciones on-line	1
Procesos complejos	2
Utilización de otros sistemas	2
Facilidad de mantenimiento	3
Facilidad de operación	2
Instalación en múltiples lugares	2
Facilidad de cambio	2

Tabla 3.6: Factores de complejidad aplicación web.

$$FA = (0,01 \times 27) + 0,65 = 0,92$$

$$PF = PFNA \times FA = 59 \times 0,92 = 54,28$$

En este caso ocurre lo mismo que para la aplicación móvil, resulta que se han utilizado varios lenguajes de programación, y para calcular las líneas de código (LDC) se hace la media entre HTML, JavaScript, J2EE y Java. Se obtiene una media de .

$$LDC = 45 \times 54,28 = 2442,6 \approx 2,4KLDC$$

Estimación por COCOMO II

COCOMO se define como Constructive Cost Model. Es un tipo de estimación que se basa en la medición del esfuerzo y el tiempo que ocupará la realización del proyecto. Similar al método usado en el apartado anterior, este busca estimar las líneas de código del sistema software. Cada sistema tendrá una estimación diferente. El procedimiento a seguir es el siguiente.

En primer lugar hay que tener en cuenta que existen tres modos de aplicar COCOMO en función del tipo de sistema que se vaya a desarrollar.

Proyecto software	a	b	c	d
Orgánico	2,4	1,05	2,5	0,38
Semi-acoplado	3,0	1,12	2,5	0,35
Empotrado	3,2	1,2	2,5	0,32

Tabla 3.7: Tipos de modelos de COCOMO

En caso de tener la necesidad de ajustar los factores de coste, existe la siguiente referencia:

Factores conductores del coste	Valor de los factores					
	Muy bajo	Bajo	Medio	Alto	Muy Alto	Extra
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
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	
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	
Técnicas modernas de programación	1,24	1,10	1,00	0,91	0,82	
Utilización de herramientas de 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	

Tabla 3.8: Factores conductores del coste COCOMO

Las fórmulas a utilizar en los cálculos referentes en COCOMO:

$$Esfuerzonominal = A \times (KLDC)^B$$

$$Esfuerzo = Esfuerzonominal \times FactoresConductoresCoste$$

$$Tiempodedesarrollo = 2,5 \times Esfuerzo^C$$

$$N^{o}mediodepersonas = Esfuerzo/TiempoDesarrollo$$

Como en nuestro caso se quiere realizar una estimación del esfuerzo requerido, se va a utilizar el modelo básico del COCOMO. Ya que se trata de un proyecto de un tamaño relativamente pequeño y realizado por un grupo muy reducido de programadores el modo orgánico será el más adecuado.

Aplicación móvil híbrida La estimación por puntos de función establece las líneas de código en 2,360 KLDC

- Esfuerzo nominal:

$$Esfuerzonominal = 2,4 \times (2,360)^{1,05} = 5,91 personas/mes$$

- Esfuerzo:

$$Esfuerzo = 5,91 \times (0,91 \times 0,95 \times 0,94 \times 1,19 \times 0,90) = 5,14 personas/mes$$

- Tiempo de desarrollo:

$$Tiempodedesarrollo = 2,5 \times (5,14)^{0,38} = 4{,}66 \approx 5 meses$$

- Nº medio de personas:

$$N^{o}mediodepersonas = 5,14/4,66 = 1,10 \approx 1 persona$$

Aplicación web La estimación por puntos de función en este caso también es similar con 2,442 KLDC

- Esfuerzo nominal:

$$Esfuerzonominal = 2,4 \times (2,442)^{1,05} = 6,12 personas/mes$$

- Esfuerzo:

$$Esfuerzo = 6,12 \times (0,91 \times 0,95 \times 0,94 \times 1,19 \times 0,90) = 5,33 personas/mes$$

- Tiempo de desarrollo:

$$Tiempodedesarrollo = 2,5 \times (5,33)^{0,38} = 4,72 \approx 5 meses$$

- Nº medio de personas:

$$N^{o}mediodepersonas = 5,33/4,72 = 1,12 \approx 1 persona$$

3.4. Presupuestos

A grandes rasgos en este ámbito de ingeniería del software, se entiende por presupuesto como el cómputo anticipado del coste de los gastos ocasionados en un proyecto de ingeniería de desarrollo. El presupuesto del proyecto en cuestión se obtiene teniendo en cuenta tres partes fundamentales: la parte de presupuesto hardware, la parte de presupuesto software y por último la parte referente al presupuesto de personal.

3.4.1. Presupuesto inicial

Para realizar un presupuesto inicial se toma una duración de 5 meses de trabajo a realizar por una única persona, como se estimo inicialmente en el anteproyecto. Esos 5 meses equivalen a 832 horas si tenemos en cuenta 8 horas de trabajo diarias.

Prespuesto hardware

Para el desarrollo de este proyecto hay que tener en cuenta los siguientes elementos hardware:

- **Ordenador personal**: Toshiba Satellite L750/L755, procesador x64 Intel® Core™ i5-2430M CPU @ 2.40GHz de cuatro núcleos, disco duro de SandDisk SDSSDA-240G-G26 SATA de 240GB, 8GB de memoria RAM DDR3 y sistema operativo Ubuntu 16.04 LTS. Teniendo en cuenta que la vida útil suele equivaler a 4 años (48 meses aproximados) y que su uso corresponde con 5 meses, el porcentaje usado sería de un 10,41 %.

- **Teléfono de pruebas**: Huawei Honor 8 FRD-L09, procesador Hisilicon Kirin 950, disco duro de 32 GB, 4GB memoria RAM DDR4, versión Android 7.0 Nougat. Aproximadamente se ha estimado un uso de un 15 %.

- **Internet**: se ha estimado un uso aproximado de un 25 % al mes durante 5 meses.

- Todo el hardware utilizado para la parte del servidor web de aplicaciones no se tendrá en cuenta ya que ha sido proporcionado por la universidad para la realización de dicho proyecto.

Componente hardware	Uso(%)	Coste total(€)	Coste(€)
Ordenador personal	10,41	699	72,76
Teléfono de pruebas	15	335	50,25
Internet	25	50 x 5	62,5
		TOTAL	185,51 €

Tabla 3.9: Presupuesto inicial componentes hardware

Presupuesto software

Los siguientes elementos software son requeridos en el sistema a crear:

- Ubuntu 16.04 LTS, XAMPP, TexMaker, NetBeans IDE, Adobe Phonegap, Atom, SqliteMan y AndroidStudio son todas herramientas gratuitas por lo que no se tendrá en cuenta en el presupuesto.

El presupuesto software gracias a la utilización de herramientas gratuitas es 0.

Presupuesto de desarrollo

Hay que tener en cuenta que el trabajo ha sido desarrollado por un Ingeniero Informático de Servicios y Aplicaciones, que dicha persona asume los tres roles desempeñados en el proyecto: analista, programador y documentalista.

Teniendo en cuenta el diagrama de planificación inicial se obtiene un cálculo estimado de los días empleados por cada rol del proyecto y a su vez se asume que cada día de trabajo equivale a 8 horas, por lo que se obtiene:

	Horas de trabajo	**Coste/hora(€)**	**Total(€)**
Analista	41 x 8	13,80	4526,4
Programador	46 x 8	11,90	4379,2
Documentalista	17 x 8	9,20	1251,2
		TOTAL	10156,8

Tabla 3.10: Presupuesto inicial de desarrollo

El salario bruto por hora se ha estimado en función del salario medio de un Ingeniero Informático sin experiencia laboral[1]

Presupuesto total inicial

Para obtener el presupuesto total inicial es necesario sumar los importes de todos los presupuestos calculados anteriormente:

Componente	**Coste (€)**
Hardware	185,51
Software	0
Desarrollo	10156,8
TOTAL	10342,31

Tabla 3.11: Coste total

3.4.2. Coste real

Para realizar un calculo del presupuesto final es necesario tener en cuenta la duración real del trabajo, que fue de 5 meses y medio.

[1] http://www.tusalario.es/main/salario/comparatusalario?job-id=2512010000000/

Coste hardware

En el caso del coste real, el hardware a utilizar será el mismo que el tomado en el apartado 3.4.1.Presupuesto hardware, por lo que el valor del costo será el mismo ya que la derivación influenciada por el incremento de tiempo real es mínima y no se tiene en cuenta. El coste hardware sería equivalente a la tabla 3.9[2].

Coste software

En este caso ocurre algo similar al coste hardware anteriormente citado, la variación de tiempo no se tiene en cuenta por su ínfimo valor a la hora de presupuestar el software, ya que el calculo del valor del presupuesto software(tabla 3.10[3]) es 0.

Coste de desarrollo

Como hemos dicho anteriormente la duración real del trabajo ha sido de 5 meses y medio. Por lo que se tiene en cuenta una aproximación de 11 días más que en el caso del presupuesto inicial, lo que equivale a 88 horas repartidas en su mayoría en las etapas de implementación, documentación y pruebas. Se obtiene el siguiente cálculo:

	Horas de trabajo	**Coste/hora(€)**	**Total(€)**
Analista	41 x 8	13,80	4526,4
Programador	51 x 8	11,90	4855,2
Documentalista	23 x 8	9,20	1692,8
		TOTAL	11074,4

Tabla 3.12: Coste de desarrollo

Coste total

Para obtener el presupuesto total inicial es necesario sumar los importes de todos los presupuestos calculados anteriormente:

Componente	**Coste (€)**
Hardware	185,51
Software	0
Desarrollo	11074,4
TOTAL	11259,91

Tabla 3.13: Presupuesto total inicial

3.5. Conclusiones

El cálculo previsto en la estimación inicial de tiempo no se ha cumplido, ya que el plazo final de entrega se ha retrasado bastantes días. Es un error muy grave teniendo en cuenta

[2]Presupuesto inicial componentes hardware
[3]Presupuesto inicial componentes software

que la diferencia presupuesto estimado inicialmente y el coste real es casi de 1000€una cantidad bastante grande. Dicho retraso ha surgido por diferentes problemas en las dos últimas iteraciones a la hora de la implementación y de las pruebas del sistema en conjunto, lo que también ha retrasado algunas partes de la documentación del proyecto.

Capítulo 4

Análisis

Este capítulo esta enfocado a realizar el análisis del sistema que se va a desarrollar. Para poder satisfacer todos los objetivos establecidos en un principio para el proyecto se debe detallar lo que debe hacer el sistema, como se especifica en los siguientes apartados.

4.1. Requisitos de usuario

Dentro del apartado de requisitos de usuario se muestra la interacción entre los actores y el sistema, representándolo como casos de uso.

4.1.1. Actores del sistema

ACT-01	**Usuario no registrado**
Descripción	Es el rol que desempeña el usuario de la aplicación antes de identificarse y actuar como un usuario registrado.

Tabla 4.1: Actor 01, Usuario no registrado.

ACT-02	**Usuario registrado.**
Descripción	Usuario favorecido de la aplicación, con este usuario se tendrá acceso a la aplicación móvil y a la plataforma web para monitorizar al usuario móvil.

Tabla 4.2: Actor 02, Usuario registrado.

ACT-03	**Reloj.**
Descripción	Es un rol que describe la necesidad de programación de un evento periódico en el sistema

Tabla 4.3: Actor 03, Reloj.

ACT-04	**Acelerómetro.**
Descripción	Es un rol que describe la forma en la que el acelerómetro produce una nueva información en el sistema de caídas.

Tabla 4.4: Actor 04, Acelerómetro.

Se tendrá en cuenta que el usuario registrado tendrá acceso a la parte privada de la aplicación, tanto móvil como web, con las mismas credenciales, es decir, el usuario que inicia sesión en la aplicación móvil se corresponde con el mismo usuario que podría acceder al mismo tiempo a la aplicación web para visualizar la posición del usuario de aplicación móvil.

4.1.2. Listado de casos de uso

Actor Primario	**Caso de uso**
Usuario no registrado	CU-01: Alta usuario
	CU-02: Login usuario
Reloj	CU-03: Tracking posición GPS
Usuario registrado	CU-04: Agregar contacto
	CU-05: Modificar contacto
	CU-06: Elegir nivel de detección
	CU-07: Dar aviso de emergencia
	CU-08: Añadir recordatorio
	CU-09: Visualizar prescripciones
	CU-10: Cerrar sesión
	CU-11: Visualizar posición GPS del usuario móvil
Acelerómetro	CU-12: Detectar caída

Tabla 4.5: Listado de casos de uso.

4.1.3. Descripción global para los casos de uso

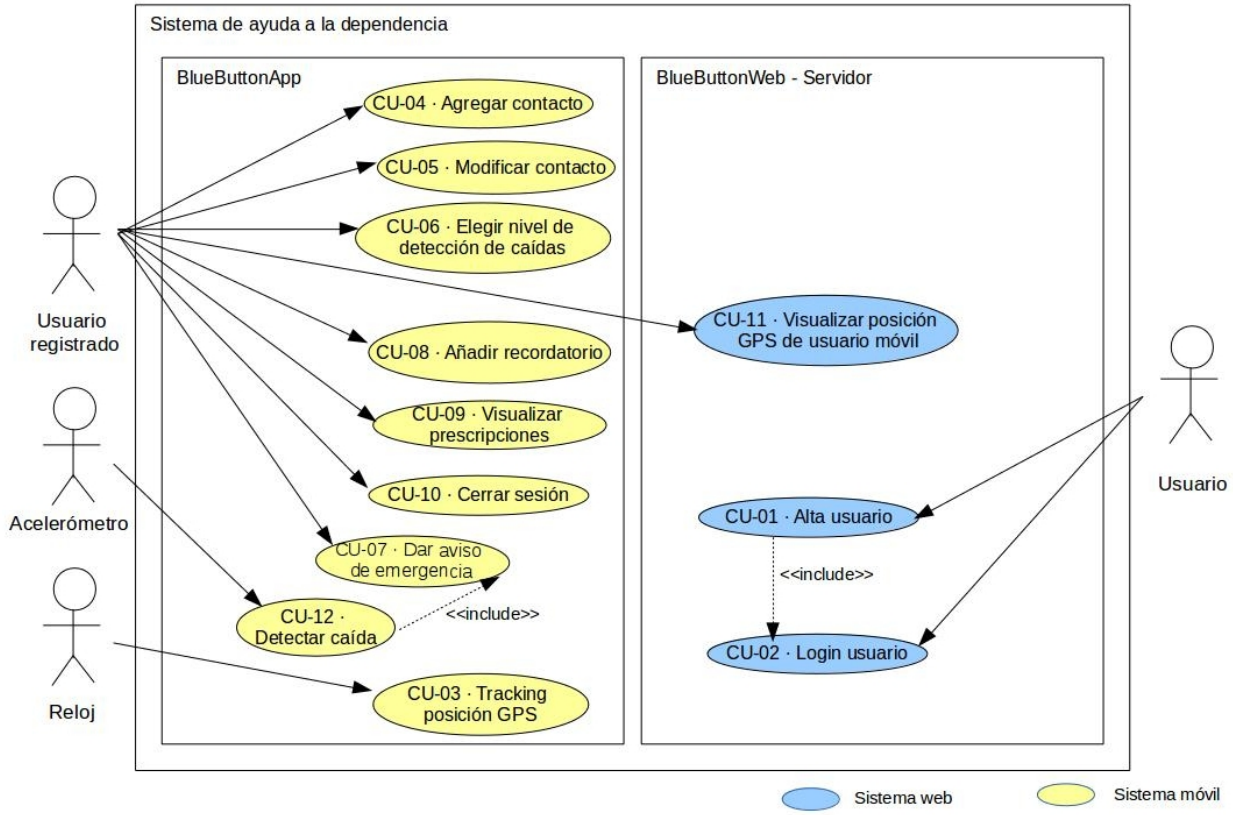


Figura 4.1: Diagrama de casos de uso

4.1.4. Especificación de casos de uso

CU-01	**Alta usuario**	
Actor principal	Usuario no registrado	
Actor secundario	-	
Trigger	-	
Descripción	El usuario se registra en la aplicación	
Precondición		
Secuencia normal	Paso	Acción
	p1	El usuario solicita el registro de un nuevo usuario
	p2	El sistema presenta los campos necesarios para el registro.
	p3	El usuario rellena los datos ofrecidos por el sistema.
	p4	El sistema valida la información introducida y la envía al servidor.
	p5	El caso de uso finaliza correctamente.
Secuencia alternativa	Paso	Acción
	p4	El usuario cancela la operación.
	p5	El caso de uso finaliza correctamente.
Postcondiciones	POST-01 · Usuario registrado correctamente.	
Excepciones	Paso	Acción
	p4	El sistema detecta un error en los datos y devuelve un mensaje de error.
	p4	El sistema detecta que el usuario ya esta registrado y devuelve un mensaje de error.
Frecuencia	1/semana	
Priodidad	Media	
Reglas de negocio		
Supuestos	Se supone que todos los usuarios disponen de un número de teléfono.	
Comentarios	Cada contacto será único e identificado por su número de teléfono. Los datos de registro se rigen por el requisito de información DD-04.	

Tabla 4.6: CU-01 · Alta usuario

CU-02	**Login usuario**	
Actor principal	Usuario registrado	
Actor secundario	-	
Trigger	-	
Descripción	El sistema deberá permitir la entrada del usuario al sistema identificándose con número de teléfono y contraseña.	
Precondición	PRE-01 · Usuario sin iniciar sesión. PRE-02 · Usuario registrado en la aplicación.	
Secuencia normal	**Paso**	**Acción**
	p1	El usuario solicita el login de sesión.
	p2	El sistema presenta los campos de inicio de sesión.
	p3	El usuario rellena los datos ofrecidos por el sistema.
	p4	El sistema valida la información introducida con el servidor y redirige al usuario a la parte privada.
	p5	El caso de uso finaliza correctamente.
Secuencia alternativa	**Paso**	**Acción**
	p3	El usuario cancela la operación.
	p4	El caso de uso finaliza correctamente.
Postcondiciones	POST-01 · Usuario logueado en el sistema.	
Excepciones	**Paso**	**Acción**
	p4	El sistema valida la información con el servidor, y al no ser correcta devuelve un mensaje de error
Frecuencia	10/día	
Priodidad	Alta	
Reglas de negocio		
Supuestos		
Comentarios		

Tabla 4.7: CU-02 · Login de usuario

CU-03	**Tracking posición GPS.**	
Actor principal	Reloj	
Actor secundario	-	
Trigger	Se ejecutará cada 15 minutos.	
Descripción	El reloj accede a la ubicación del dispositivo y la envía al servidor cada 15 minutos.	
Precondición	PRE-01 · Usuario registrado en la aplicación. PRE-02 · El acceso a la ubicación en el dispositivo debe estar activado.	
Secuencia normal	**Paso**	**Acción**
	p1	El actor reloj obtiene los datos de longitud y latitud, y los envía al sistema servidor.
	p2	El sistema servidor almacena los datos de posición y el caso de uso finaliza.
Secuencia alternativa	**Paso**	**Acción**
	p3	El servidor devuelve mensaje de error
Postcondiciones	-	
Excepciones	**Paso**	**Acción**
	-	-
Frecuencia	4/hora	
Priodidad	Muy Alta	
Reglas de negocio		
Supuestos		
Comentarios		

Tabla 4.8: CU-03 · Tracking posición GPS

CU-04	**Agregar contacto**	
Actor principal	Usuario de la aplicación	
Actor secundario	-	
Trigger	-	
Descripcion	El usuario de la aplicación añade un contacto de emergencia	
Precondicion	PRE-01 · Usuario registrado en la aplicación	
Secuencia normal	**Paso**	**Acción**
	p2	El usuario solicita agregar un nuevo contacto.
	p3	El sistema muestra los campos necesarios para la operación.
	p4	El usuario rellena los campos ofrecidos por el sistema.
	p5	El sistema actualiza las preferencias compartidas con dicha información.
	p6	El caso de uso finaliza correctamente.
Secuencia alternativa	**Paso**	**Acción**
	p5	El usuario no facilita la información
	p6	El caso de uso finaliza correctamente.
Postcondiciones	POST-01 · Contacto actualizado correctamente	
Excepciones	**Paso**	**Acción**
	-	-
Frecuencia	1/mes	
Prioridad	Media	
Reglas de negocio		
Supuestos	Se supone que todos los posibles usuarios de la aplicación poseen un número de contacto	
Comentarios	El contacto será aquel al que se le realicen las llamadas de emergencia.	

Tabla 4.9: CU-04 · Agregar contacto

CU-05	**Modificar contacto**	
Actor principal	Usuario registrado	
Actor secundario	-	
Trigger	-	
Descripción	El usuario modifica los datos de su contacto de emergencia.	
Precondición	PRE-01 · Usuario registrado en la aplicación.	
Secuencia normal	**Paso**	**Acción**
	p1	El usuario solicita modificar contacto.
	p2	El sistema presenta campos de contacto almacenados.
	p3	El usuario modifica los datos de contacto.
	p4	El sistema valida los datos y actualiza las preferencias compartidas.
	p5	El caso de uso finaliza correctamente.
Secuencia alternativa	**Paso**	**Acción**
	p3	El usuario rechaza la petición
	p4	El caso de uso finaliza correctamente.
Postcondiciones	POST-01 · Contacto actualizado	
Excepciones	**Paso**	**Acción**
	-	-
Frecuencia	2/semana	
Priodidad	Media	
Reglas de negocio		
Supuestos	Si no hay datos almacenados de contacto, el sistema devolverá un conjunto de datos vacíos	
Comentarios		

Tabla 4.10: CU-05 · Modificar contacto

CU-06	**Elegir nivel de detección**	
Actor principal	Usuario registrado	
Actor secundario	-	
Trigger	-	
Descripción	El usuario elige uno de los tres niveles de detección de caídas.	
Precondición	PRE-01 · Usuario registrado en la aplicación.	
Secuencia normal	**Paso**	**Acción**
	p1	El usuario solicita elegir un nivel de detección al sistema.
	p2	El sistema devuelve los tres valores correspondientes a los niveles.
	p3	El usuario envía el nivel que elige al sistema.
	p4	El sistema calibra los valores del sensor en función al nivel elegido por el usuario.
	p5	El caso de uso finaliza correctamente.
Secuencia alternativa	**Paso**	**Acción**
	-	-
Postcondiciones	POST-01 · Nivel de detección de caídas actualizado.	
Excepciones	**Paso**	**Acción**
	-	-
Frecuencia	3/día	
Priodidad	Media	
Reglas de negocio		
Supuestos		
Comentarios	Los valores correspondientes con los niveles de detección son tres: Poco sensible, Normal y Muy sensible.	

Tabla 4.11: CU-06 · Elegir nivel de detección

CU-07	**Dar aviso de emergencia**	
Actor principal	Usuario registrado	
Actor secundario	-	
Trigger	-	
Descripción	El usuario emite a través la aplicación una llamada telefónica a su contacto de emergencia.	
Precondición	PRE-01 · Usuario registrado en la aplicación. PRE-02 · El contacto de emergencia debe tener un número de teléfono guardado.	
Secuencia normal	**Paso**	**Acción**
	p1	El usuario solicita una llamada de emergencia al sistema.
	p2	El sistema realiza la llama al número de contacto almacenado para el usuario.
	p3	El caso de uso finaliza correctamente.
Secuencia alternativa	**Paso**	**Acción**
	-	-
Postcondiciones	-	
Excepciones	**Paso**	**Acción**
	p2	Cuando el número de contacto sea incorrecto el sistema mostrará un mensaje de error.
Frecuencia	2/día	
Priodidad	Alta	
Reglas de negocio		
Supuestos	El usuario tiene cobertura móvil para realizar la llamada.	
Comentarios	La llamada tiene los costes añadidos en función de la tarifa telefónica que tenga el usuario en su línea.	

Tabla 4.12: CU-07 · Dar aviso de emergencia

CU-08	**Añadir recordatorio**	
Actor principal	Usuario registrado	
Actor secundario	-	
Trigger	-	
Descripción	El usuario añade una prescripción a la aplicación que incluye recordatorios en función del medicamento y los parámetros que el usuario haya introducido.	
Precondición	PRE-01 · Usuario registrado en la aplicación.	
Secuencia normal	**Paso**	**Acción**
	p1	El usuario solicita agregar un recordatorio al sistema.
	p2	El sistema presenta los campos a rellenar para la operación.
	p3	El usuario rellena los campos presentados por el sistema.
	p4	El sistema valida la información, la almacena y lanza una alerta de creación de medicamento nuevo.
	p5	El caso de uso finaliza correctamente.
Secuencia alternativa	**Paso**	**Acción**
	p3	El usuario cancela la creación del nuevo medicamento
	p4	El caso de uso finaliza correctamente.
Postcondiciones	POST-01 · Medicamento añadido y recordatorio de toma creado.	
Excepciones	**Paso**	**Acción**
	p4	El sistema valida la información, si surge algún problema mostrará una alerta de error al usuario.
Frecuencia	50/día	
Priodidad	Alta	
Reglas de negocio		
Supuestos		
Comentarios	Los datos relativos a la creación de un nuevo recordatorio tendrán en cuenta los requisitos de información DD-01, DD-02, DD-03	

Tabla 4.13: CU-08 · Añadir recordatorio.

CU-09	**Visualizar prescripciones**	
Actor principal	Usuario registrado	
Actor secundario	-	
Trigger	-	
Descripción	El sistema muestra un listado con las prescripciones que tiene actualmente el usuario	
Precondición	PRE-01 · Usuario registrado en la aplicación. PRE-02 · Tener alguna prescripción dada de alta en la aplicación móvil.	
Secuencia normal	**Paso**	**Acción**
	p1	El usuario solicita un listado de prescripciones.
	p2	El sistema devuelve un listado de las prescripciones almacenadas.
	p3	El caso de uso finaliza correctamente.
Secuencia alternativa	**Paso**	**Acción**
	-	-
Postcondiciones	-	
Excepciones	**Paso**	**Acción**
	p3	Si el usuario no tiene ninguna prescripción dada de alta, obtendrá listado vacio .
Frecuencia	50/día	
Priodidad	Alta	
Reglas de negocio		
Supuestos	-	
Comentarios	En este caso los datos a mostrar serán solo los campos de nombre del requisito de información DD-02 y fecha-toma del DD-03.	

Tabla 4.14: CU-09 · Visualizar prescripciones.

CU-10	Cerrar sesión.	
Actor principal	Usuario registrado	
Actor secundario	-	
Trigger	-	
Descripción	El usuario cierra la sesión en la aplicación, web o móvil.	
Precondición	PRE-01 · Usuario registrado en la aplicación.	
Secuencia normal	**Paso**	**Acción**
	p1	El usuario solicita el cierre de sesión al sistema.
	p2	El sistema borra la sesión del usuario almacenada.
	p3	El caso de uso finaliza correctamente.
Postcondiciones	El usuario ya no aparece como iniciado en el sistema.	
Excepciones	**Paso**	**Acción**
	-	-
Frecuencia	10/día	
Priodidad	Media	
Reglas de negocio		
Supuestos		
Comentarios		

Tabla 4.15: CU-10 · Cerrar sesión.

CU-11	Visualizar posición GPS del usuario móvil	
Actor principal	Usuario registrado	
Actor secundario	-	
Trigger	-	
Descripción	La aplicación web muestra al usuario la última posición registrada del usuario móvil	
Precondición	PRE-01 · Usuario registrado en la aplicación. PRE-02 · El usuario debe tener registrada alguna posición en la base de datos.	
Secuencia normal	**Paso**	**Acción**
	p1	El usuario solicita visualizar la posición GPS del usuario móvil.
	p2	El sistema devuelve la última posición registrada.
	p3	El caso de uso finaliza correctamente.
Secuencia alternativa	**Paso**	**Acción**
	-	-
Postcondiciones		
Excepciones	**Paso**	**Acción**
	-	-
Frecuencia	10/día	
Priodidad	Alta	
Reglas de negocio		
Supuestos	Se supone que el usuario tiene mínimo una posición registrada en el servidor	
Comentarios		

Tabla 4.16: CU-11 · Visualizar posición GPS del usuario móvil.

CU-12	Detectar caída	
Actor principal	Acelerómetro	
Actor secundario	-	
Trigger	Sensor del acelerómetro preparado.	
Descripción	El Acelerómetro rastrea el valor de la aceleración del dispositivo móvil en los eje x, y, z, y los compara con las variables asignadas al nivel de detección de caídas.	
Precondición	PRE-01 · Usuario registrado en la aplicación.	
Secuencia normal	**Paso**	**Acción**
	p1	El Acelerómetro detecta una caída y envía una petición al sistema.
	p2	El sistema mostrará una alerta preguntando al usuario su estado
	p3	El usuario devolverá al sistema el valor correcto.
	p4	El sistema mostrará una alerta al usuario mostrando "Perfecto!".
	p5	El caso de uso finaliza correctamente.
Secuencia alternativa	**Paso**	**Acción**
	p3	El usuario devuelve al sistema un valor incorrecto.
	p4	El sistema llamará al caso de uso CU-07.
	p5	El caso de uso finaliza correctamente.
Postcondiciones		
Excepciones	**Paso**	**Acción**
	-	-
Frecuencia	5/día	
Priodidad	Alta	
Reglas de negocio		
Supuestos		
Comentarios	Hay que tener en cuenta que si el móvil solo se cae el sistema lo detecta	

Tabla 4.17: CU-12 · Detectar caída.

4.2. Reglas de negocio

- RN-01 · Los usuarios de la aplicación deberán registrarse con un número de teléfono que permita realizar llamadas salientes para permitir el correcto funcionamiento de la aplicación.
- RN-02 · El uso que proporciona la plataforma será gratuito para todos los usuarios.
- RN-03 · El tratamiento y almacenamiento de datos, se hará respetando la LOPD(Ley Orgánica de Protección de Datos).
- RN-04 · Se cumplirá el Real Decreto-Ley 13/2012, y en consecuencia LSSOI(Ley de Servicios de la Sociedad de la Información y del Comercio Electrónico)

4.3. Requisitos no funcionales

Los requisitos no funcionales indican una propiedad del sistema, especificando como de bien la realiza o la debería realizar. Son los siguientes:

- RNF-01 · Se deberá poder acceder a la aplicación móvil desde dispositivos móviles Android.
- RNF-02 · La aplicación móvil deberá poder ejecutarse desde cualquier dispositivo Android con una versión superior a la 4.0.
- RNF-03 · La aplicación web será accesible desde los navegadores más utilizados como Google Chrome, Mozilla Firefox, Internet Explorer, Edge, y Safari.
- RNF-04 · Tanto la autenticación de la aplicación móvil se hará a través del servidor y la contraseña se mantendrá cifrada con MD5.
- RNF-05 · La aplicación móvil debe ejecutar todos los procesos y los cambios entre vistas y ventanas en un máximo de tres segundos y medio.
- RNF-06 · Tanto la aplicación móvil como la aplicación web deberá estar disponible las 24 horas del día los 7 días de la semana.
- RNF-07 · El tamaño de los componentes y del texto de la aplicación móvil están diseñados para que sean simples y de un tamaño visible para el posible uso de personas mayores.
- RNF-08 · La reducida funcionalidad de la aplicación móvil esta pensada para facilitar el uso de la misma a las personas con dependencia a las que va orientada.
- RNF-09 · La aplicación web esta pensada para que cualquier persona que supervise al usuario de la aplicación móvil, y tenga accesos a sus datos, pueda obtener rápidamente la posición GPS con solo loguearse.
- RNF-10 · Los datos relativos a la aplicación móvil y a las prescripciones del usuario se almacenarán como SQLite para disminuir el tamaño en el dispositivo.

Por último, si un requisito no funcional describe conexiones entre el sistema y otro sistema o dispositivo se denominan Requisitos de Interfaz Externa, en este caso son los siguientes:

- RIE-01 · La aplicación móvil se comunicará con el servidor web mediante peticiones GET y POST.
- RIE-02 · El sistema se comunicará con el servicio de Google Maps a través de la API del mismo[1], para visualizar un mapa en la aplicación web y situar una localización mediante la latitud y la longitud.

4.4. Requisitos de información

El sistema debe almacenar un conjunto de información relativa a los usuarios y a las prescripciones médicas de los mismos para permitir su correcto funcionamiento. Estos datos se recogen mediante los requisitos de información, que son aquellos que describen que información almacena el sistema para poder ofrecer sus servicios.

[1]API Google Maps - https://developers.google.com/maps/documentation/?hl=es-419

4.4.1. Modelo de datos conceptual

Para facilitar el modelado conceptual de los datos del problema se ha optado por la representación mediante un diagrama de Entidad - Relación (E-R).

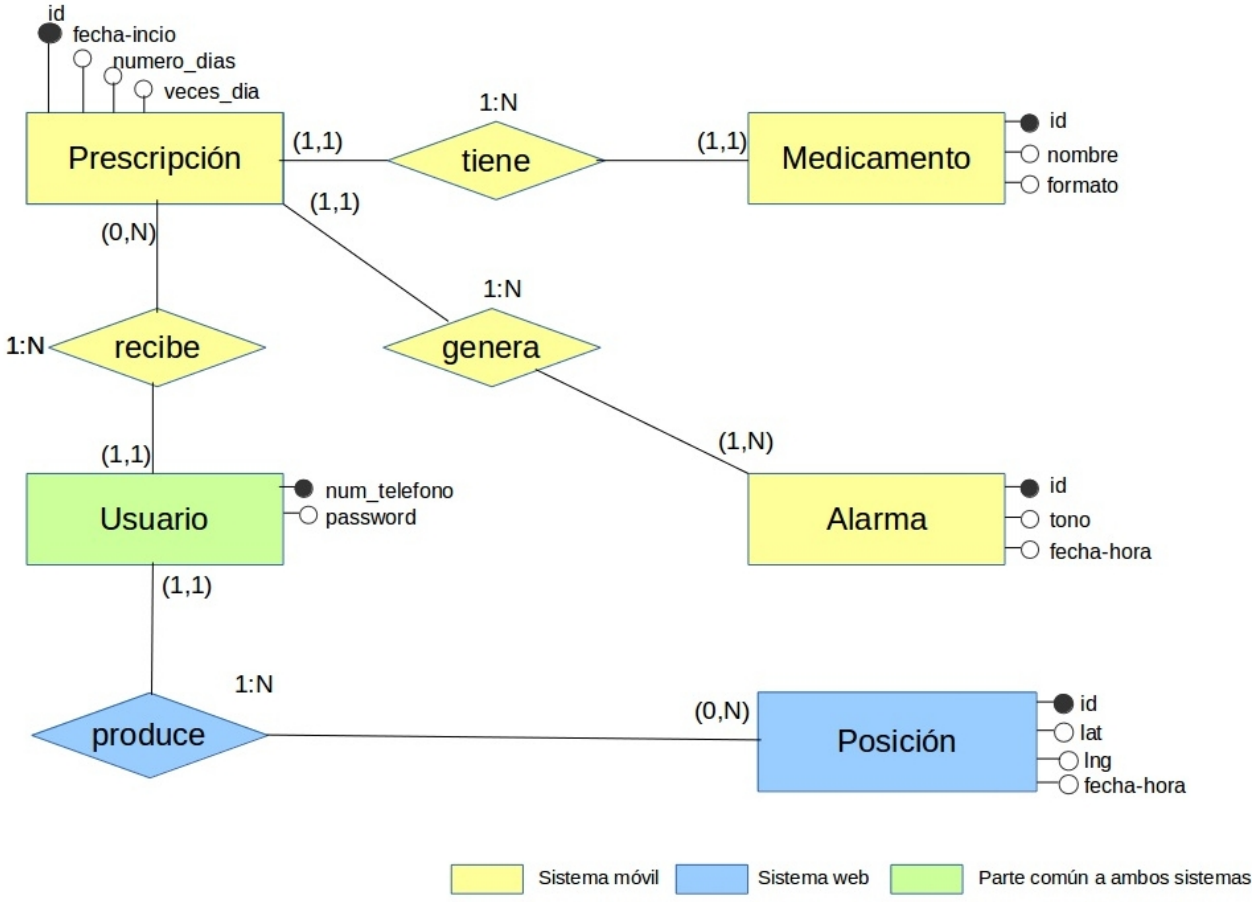


Figura 4.2: Diagrama Entidad - Relación (E-R)

4.4.2. Diccionario de datos

Mediante el uso del diccionario del datos definimos la composición de estructuras de datos, su significado, formato y los valores permitidos para los elementos de datos que componen esas estructuras. A continuación se detallará los distintos datos pertenecientes a las entidades que intervienen en el modelo conceptual.

- **Prescripción**: representa a las distintas prescripciones médicas que pudiera tener el paciente. Se incluye un *id* único e identificativo, una *fecha-inicio* que indica el momento de creación de la prescripción en el dispositivo, un campo *numero_dias* que indica la duración en días de la misma, y *veces_dia* donde se expresa el número de tomas de la medicación.

Atributos	Descripción	Tipo de dato	Longitud	Otro
_id	Identificador único de la prescripción	INTEGER	11	Auto Incremental
fecha-inicio	Fecha inicial de la prescripción	DATETIME	11	Obligatorio
num_dias	Duración del tratamiento	INTEGER	11	Obligatorio
veces_dia	Número de tomas al día,	INTEGER	11	Obligatorio

Tabla 4.18: DD-01 · Entidad Prescripción

- **Medicamento**: se encarga de almacenar correctamente los distintos tipo de medicamentos. Un *id* único que permite su localización, un nombre descriptivo y único, y por último un número entero que valora si el medicamento es de tipo sobre o pastilla, según su valor, 1 sobre, 2 pastilla.

Atributo	Descripción	Tipo de dato	Longitud	Otro
id	Identificador único del medicamento	INTEGER	11	Auto Incremental
nombre	Almacena el nombre del medicamento de cada prescripción	VARCHAR	40	Obligatorio
formato	Almacena un valor en función del tipo de formato:	INTEGER	11	Obligatorio

Tabla 4.19: DD-02 · Entidad Medicamento

- **Alarma**: es una entidad que contiene todo lo necesario para hacer saltar la alarma de toma de medicamento por parte del sistema para el usuario.

Atributo	Descripción	Tipo de dato	Longitud	Otro
id	Identificador único de la alarma	INTEGER	11	Auto Incremental
fecha_toma	Fecha en la que debe realizarse la toma de medicamento	DATETIME	40	Obligatorio
tono	almacena la ubicación del tono de alarma	VARCHAR	40	Obligatorio
numero_toma	Controla el número de tomas que lleva actualmente el usuario sobre esa prescripción de medicamento	INTEGER	11	Obligatorio

Tabla 4.20: DD-03 · Entidad Alarma

- **Usuario**: sirve como credenciales de usuario al inciar sesión en cualquiera de las dos plataformas. Un id único que corresponde con el número de telefono con el que se usará la apliación móvil y la contraseña cifrada en el lado servidor.

Atributo	Descripción	Tipo de dato	Longitud	Otro
num_telefono	Identificador único de la alarma, que se corresponde con su número de teléfono	INTEGER	11	Único
password	Almacena la contraseña de inicio de sesión cifrada con el algoritmo SHA-256 para favorecer su seguridad	VARCHAR	255	Obligatorio

Tabla 4.21: DD-04 · Entidad Usuario

- **Posición**: modela la información relativa sobre los lugares que han sido registrados por el GPS del teléfono móvil: latitud, longitud, fecha y hora y el id del usuario al que esta relacionado.

Atributo	Descripción	Tipo de dato	Longitud	Otro
id	Identificador único de posición	INTEGER	11	Auto Incremental
Lat	Almacena la latitud registrada por el dispositivo	VARCHAR	40	Obligatorio
Lng	Almacena la longitud registrada por el dispositivo	VARCHAR	40	Obligatorio
fecha-hora	Fecha y hora del momento en el que se registro la posición.	DATETIME	11	Obligatorio

Tabla 4.22: DD-05 · Entidad Posición

Capítulo 5

Diseño

En este capítulo se tendrá en cuenta que diseño tendrá el sistema a desarrollar. Se detallará con la mayor exactitud posible cómo llegar al objetivo de un sistema completo, funcional y eficiente. Se tendrá en cuenta la arquitectura física y lógica, el diagrama de clases del proyecto, los distintos diagramas de secuencia, los modelos lógicos de datos y el diseño de la interfaz mediante prototipos.

5.1. Arquitectura lógica

La arquitectura lógica tiene la función de describir los diferentes componentes lógicos del sistema y la relación entre los mismos. El diagrama siguiente la refleja y engloba a las dos aplicaciones en una sola.

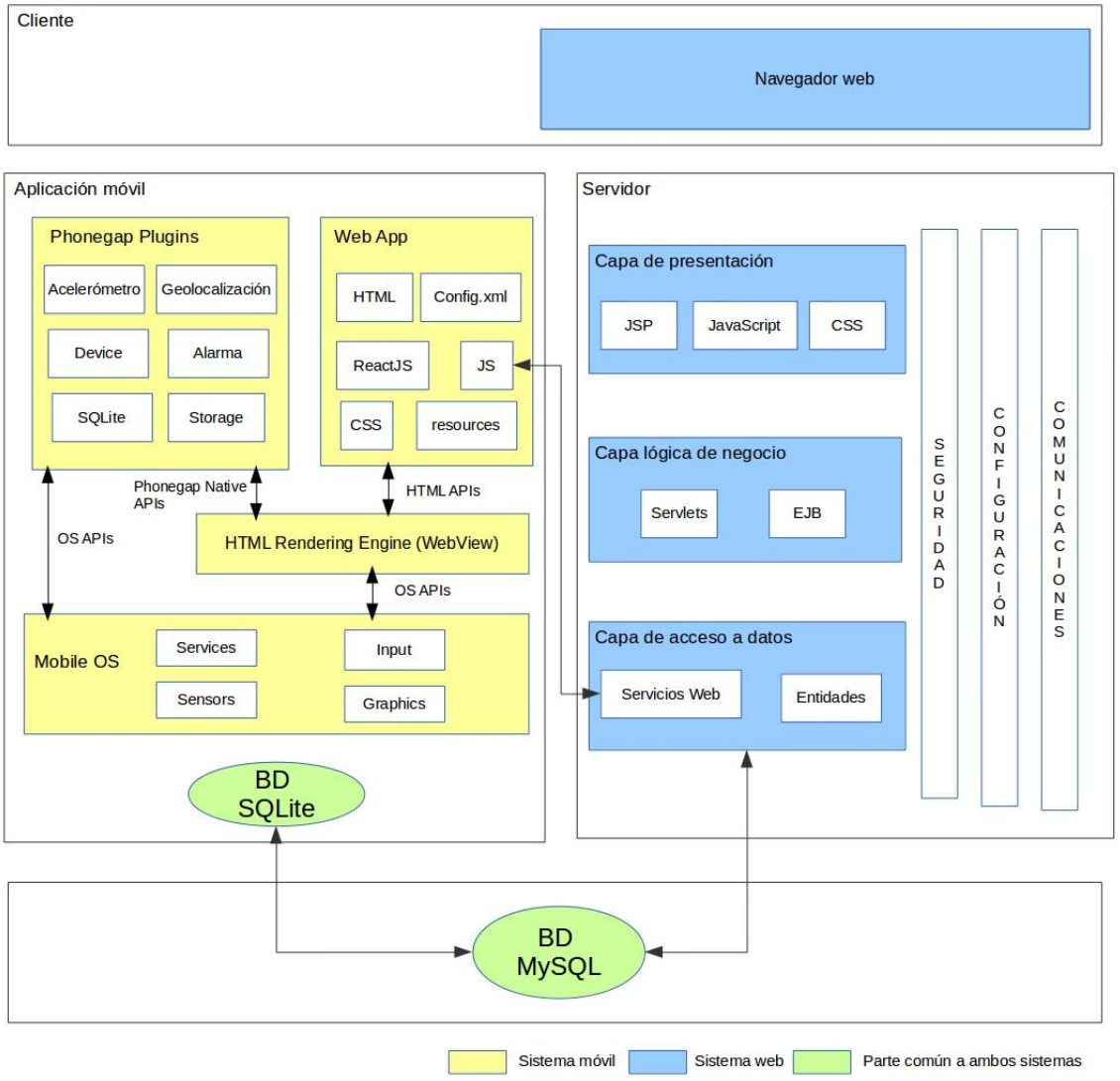


Figura 5.1: Arquitectura lógica.

A simple vista se observa que los componentes situados a la izquierda de la figura anterior se refieren a la parte de la aplicación móvil Phonegap, y en cambio los situados a la derecha lo hacen para la parte del servidor. El servidor maneja una estructura de 3 capas: presentación, lógica de negocio, y capa de acceso a datos. Ambas aplicaciones comparten servidor de base de datos, aunque la aplicación móvil tiene su propia base de datos interna, en SQLite, ajena al servidor.

Mientras que la aplicación móvil basada en Phonegap esta compuesta por cuatro elementos:

- **WebApp**: que contiene el código referente a las funcionalidades de la aplicación y a la parte visual ejecutada con ReactJS mediante el framework TouchStoneJS[1].

[1]Se explicará en el Capítulo 6.2 del mismo.

- **WebView**: se encarga de renderizar la parte contenida en la capa anterior, proporciona un navegador incrustado en la aplicación ejecutada que lo simula como si de una página web se tratara.

- **Plugins**: se utilizan para comunicar la aplicación móvil con los elementos que forman el dispositivo o sus componentes hardware.

- **OS APIs**: se trata de aquellos mecanismos proporcionados por el SDK o el xCode que hace funcionar la aplicación en los distintos sistemas operativos.

5.2. Arquitectura física

La arquitectura física representa los distintos componentes físicos que forman parte del sistema y el tipo de relaciones entre ellos. La topología muestra los componentes hardware utilizados para la creación del nuevo sistema.

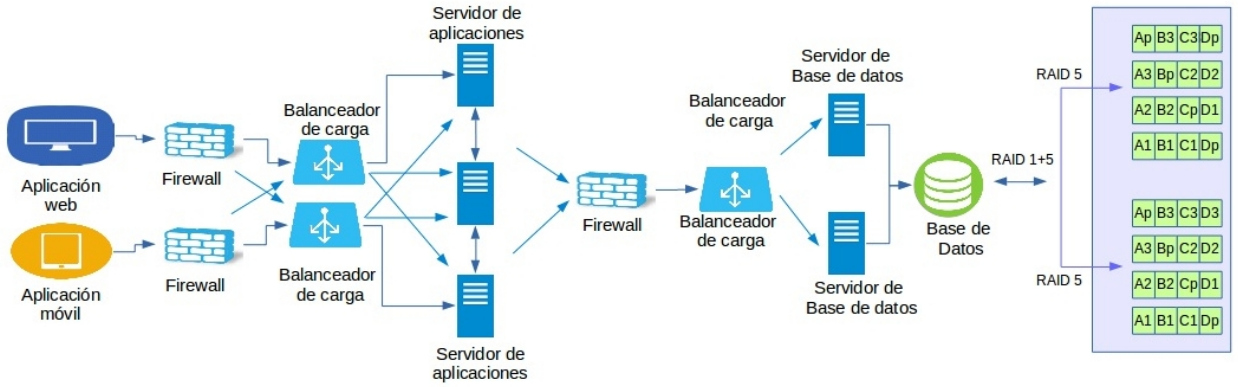


Figura 5.2: Arquitectura Física: topología.

Se puede ver como los firewalls se encargarán de brindar seguridad al sistema filtrando peticiones antes del procesamiento en los servidores. Los balanceadores de carga manejarán las peticiones para repartirlas en función de la carga de trabajo de los servidores. Las base de datos es donde se almacenará la mayor parte de la información usada. Se incorpora un RAID 1+5 a nivel de datos para evitar problemas de integridad y facilitar la recuperación en el sistema.

El diagrama de despliegue físico del sistema describirá la configuración de todos los nodos del sistema y la disposición de componentes de los mismos.

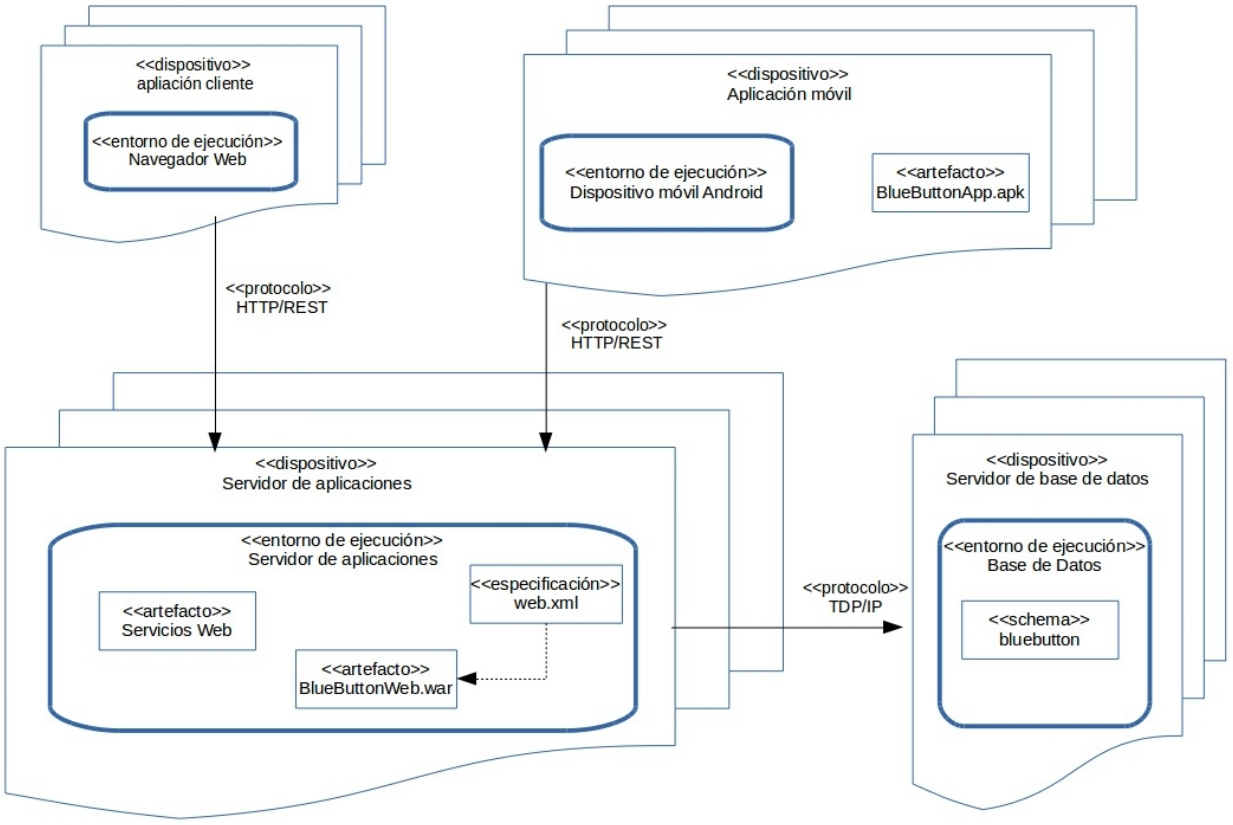


Figura 5.3: Arquitectura Física: diagrama de despliegue.

El servidor de aplicaciones interacciona con todos los componentes del sistema, tanto con la parte de base de datos a través de peticiones TCP/IP, como en el caso del acceso al sistema a través de la aplicación móvil o de un navegador web mediante servicios REST.

5.3. Diagramas de clases

Los diagrama de clases de Diseño añaden al análisis del proyecto los aspectos más específicos sobre el diseño que tendrá el sistema. Ambas aplicaciones utilizan MVC para su diseño, es decir, que mantiene separada tanto la lógica de negocio y datos, como la interfaz de usuario y la parte que se encarga de las comunicaciones entre eventos. En este caso sólo se incluirá el diagrama de clases del servidor de aplicaciones y base de datos, ya que la aplicación móvil Phonegap viene desarrollada en ReactJS, por lo que gestiona de manera distinta la lógica de negocio y las interfaces, a través de prototipos, todo ejecutado dentro del mismo archivo (app.js).

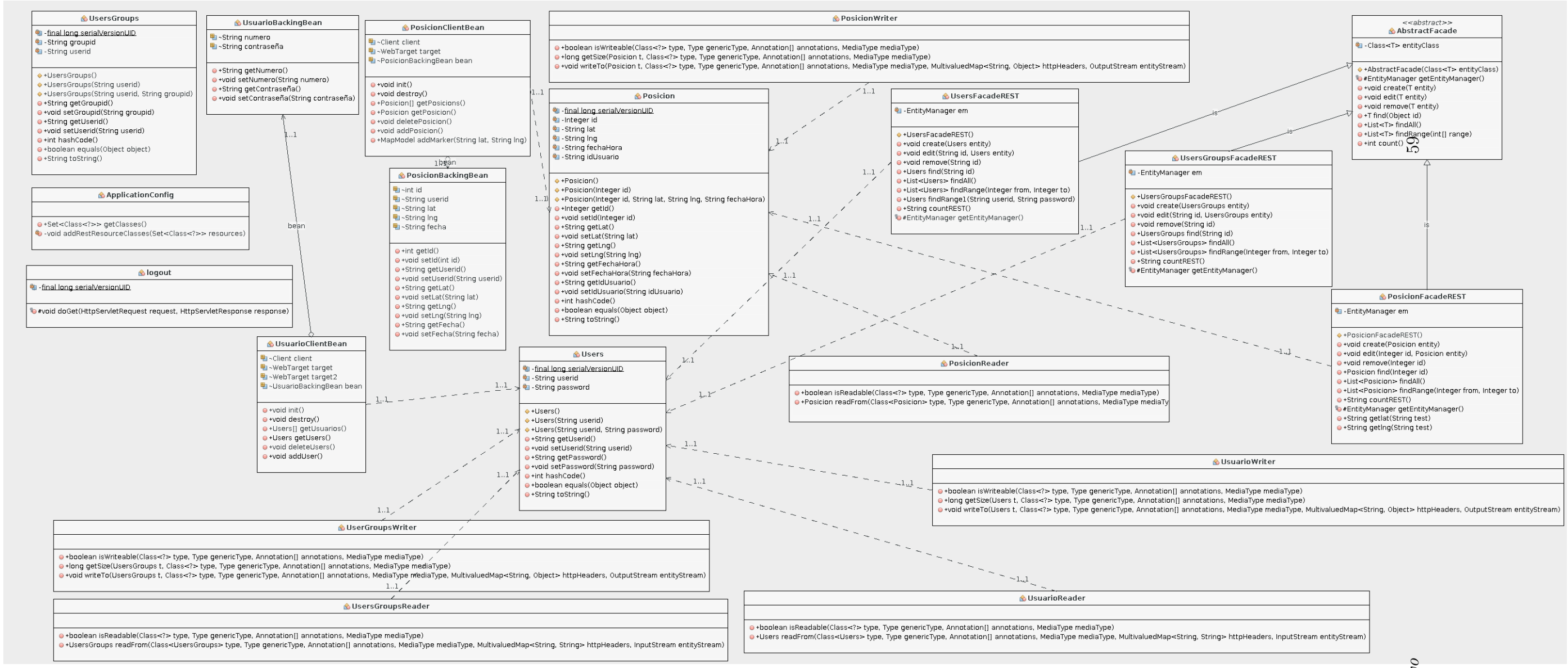


Figura 5.4: Diagrama de clases · Servidor de aplicaciones.

5.4. Diagramas de secuencia

Lo diagramas de secuencia describen las interacciones existentes entre los distintos componentes del sistema durante su vida útil. En algunos casos los diagramas son semejantes en la aplicación móvil tanto como en la aplicación web. A continuación se ejemplificarán los diagramas que suponen mayor dificultad a la hora de explicar e implementar dicho proceso ya que el resto son bastante comunes.

El primero de los diagramas que entraña mayor complicación, es el correspondiente al caso de uso *CU-03 - Tracking posición GPS*. Comprende el proceso para enviar la posición controlada por un reloj que hará que cada 15 minutos se ejecute una solicitud de la posición al sensor del dispositivo, si este esta disponible, devolverá la información requerida que a su vez será tratada y enviada por mediante una petición POST a los servicios REST del servidor web. Cuando el servidor recibe la petición la trata y devuelve un valor "trueçuando todo este correcto. En el caso de que la petición no tenga éxito se devuelve un valor false y se le notificará al usuario.

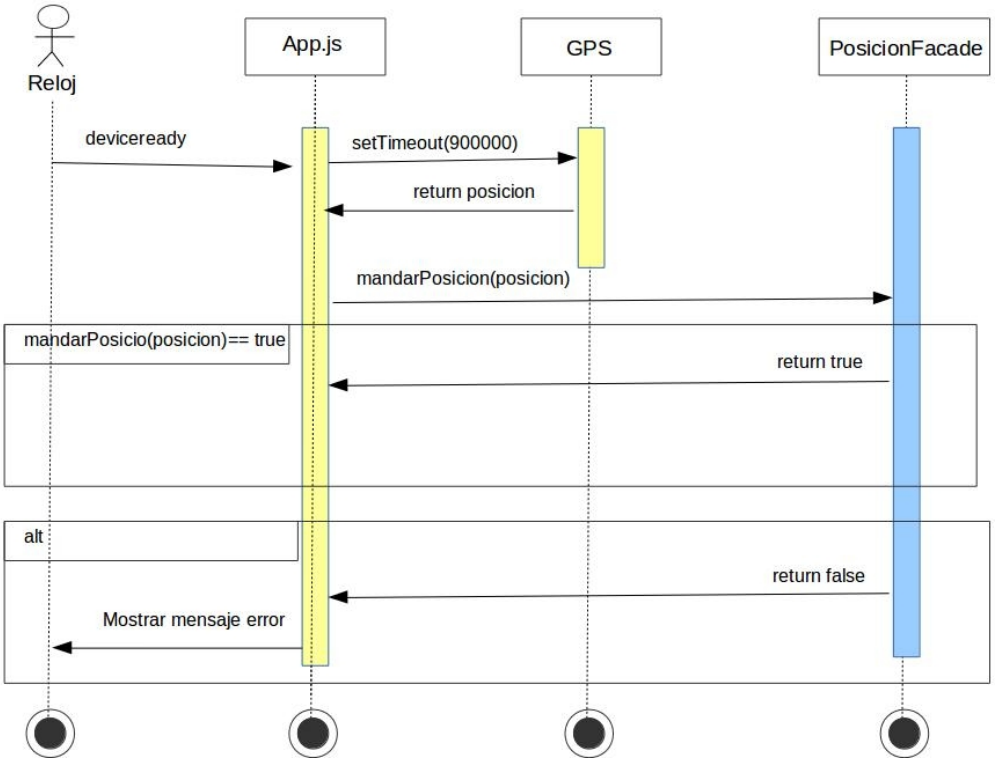


Figura 5.5: Diagrama de secuencia · CU-03 · Tracking posición GPS

El otro caso, se trata del comportamiento del sistema en el caso de uso *CU-12 - Detectar caída*. Comprende el flujo generado cuando el actor Acelerómetro detecta una caída en el dispositivo móvil. El Acelerómetro accede al sensor y obtiene a tiempo real los valores de la aceleración en los ejes x, y ,z, y en el momento que superen unos límites definidos llamará al caso de uso *CU-08*, que lo tratará como una emergencia.

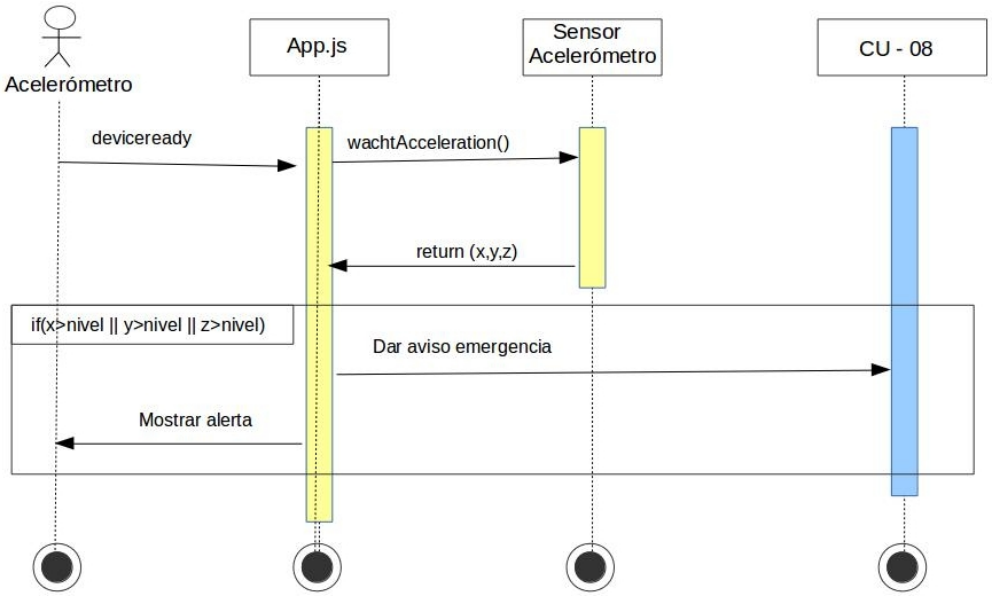


Figura 5.6: Diagrama de secuencia · CU-12 · Detectar caída

5.5. Modelo lógico de datos

El modelo de lógico de datos de la aplicación móvil híbrida y de la aplicación web se han juntado para obtener una visión mas global del sistema.

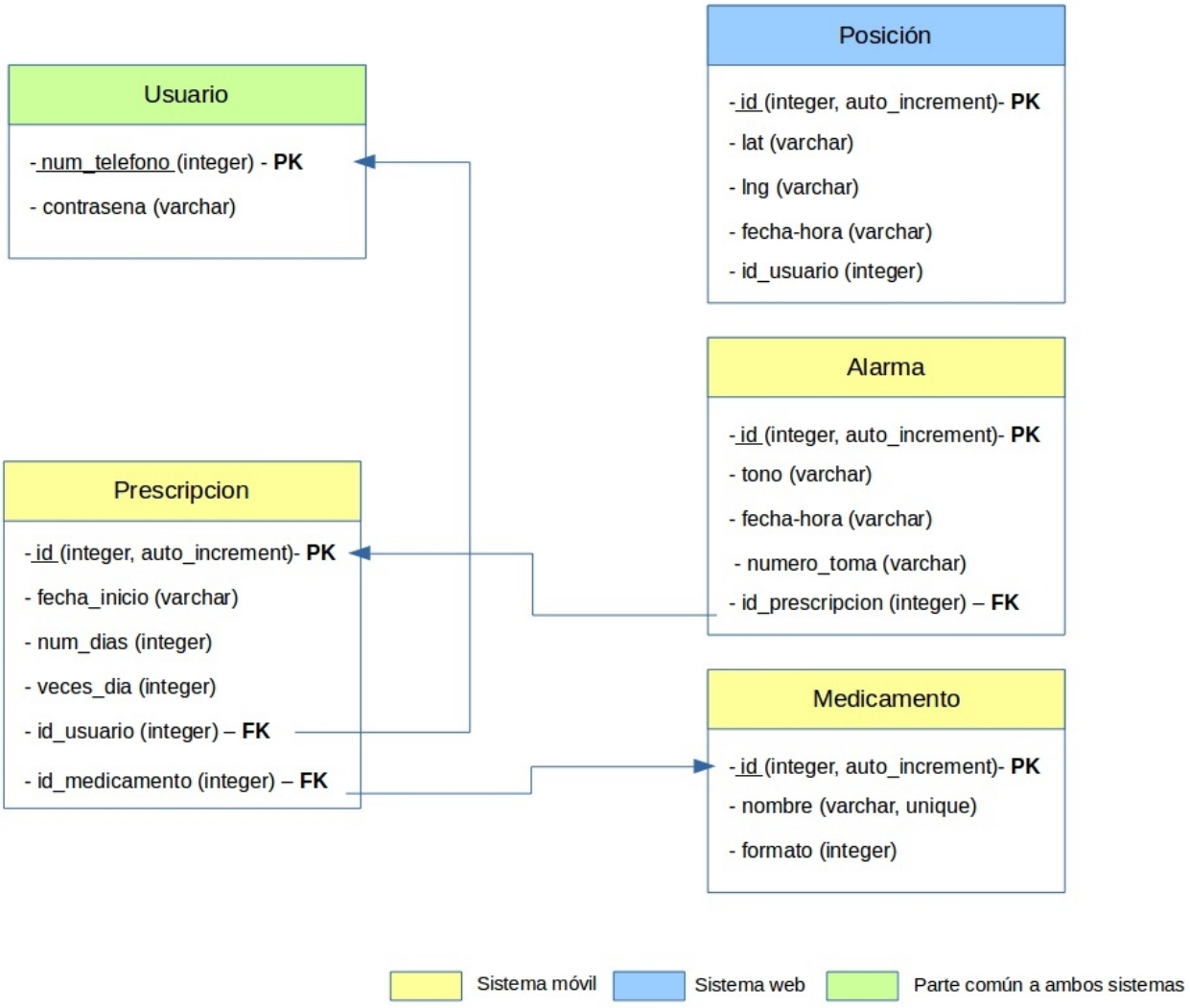


Figura 5.7: Modelo lógico de datos - Modelo Relacional

El modelo lógico de datos se ha extrapolado a partir del diagrama Entidad-Relación[2]. Se han generado cinco tablas relacionales, representadas en el E-R como entidades, las que heredan los mismo atributos que las entidades equivalentes. Los casos especiales son aquellos en los que la relación se ha transformado en una clave foránea o *foreign key* como en los siguientes casos:

- En el caso de la tabla Prescripción hereda el *id_usuario* de la tabla Usuario. Todas las prescripciones deberán tener asignado un *id_usuario*. También el campo *id_medicamento* hereda de la tabla medicamento, atributo *id*, ya que todas las prescripciones deben tener asignado un medicamento obligatoriamente.

[2]Figura 4.2 Diagrama Entidad-Relación(E-R)

- La tabla Posiciones hereda el *id_usuario* de la tabla Usuario, para guardar una relación entre las posiciones almacenadas y el usuario que las registra.

- En la tabla Alarma, el campo *id_prescripcion* hereda del atributo *id* de la tabla prescripciones, es decir, todas las alarmas deberán hacer referencia a alguna prescripción.

5.6. Diseño de la interfaz

El diseño de la interfaz de usuario fue uno de los puntos que más se tuvieron en cuenta a la hora de plantear el proyecto, ya que además de buscar una interfaz agradable a la vista, se tenía que tener en cuenta al público al que va dirigida la aplicación. Otro de los puntos de vista que se tuvieron en cuenta a la hora de pensar en la interfaz, fue que el primer desarrollo se realiza para Android, pero pensando en un futuro su aplicación en iOS también, lo que genera problemas en la gestión de pantallas y la renderización de una plataforma a otra, por lo que se opto por una visualización con más facilidades para la posterior adecuación a iOS. A continuación se indican los primeros prototipos que se plantearon al inicio del desarrollo, aunque en un principio se desarrollaron a mano alzada, se adaptaron a formato digital para mejorar su calidad.

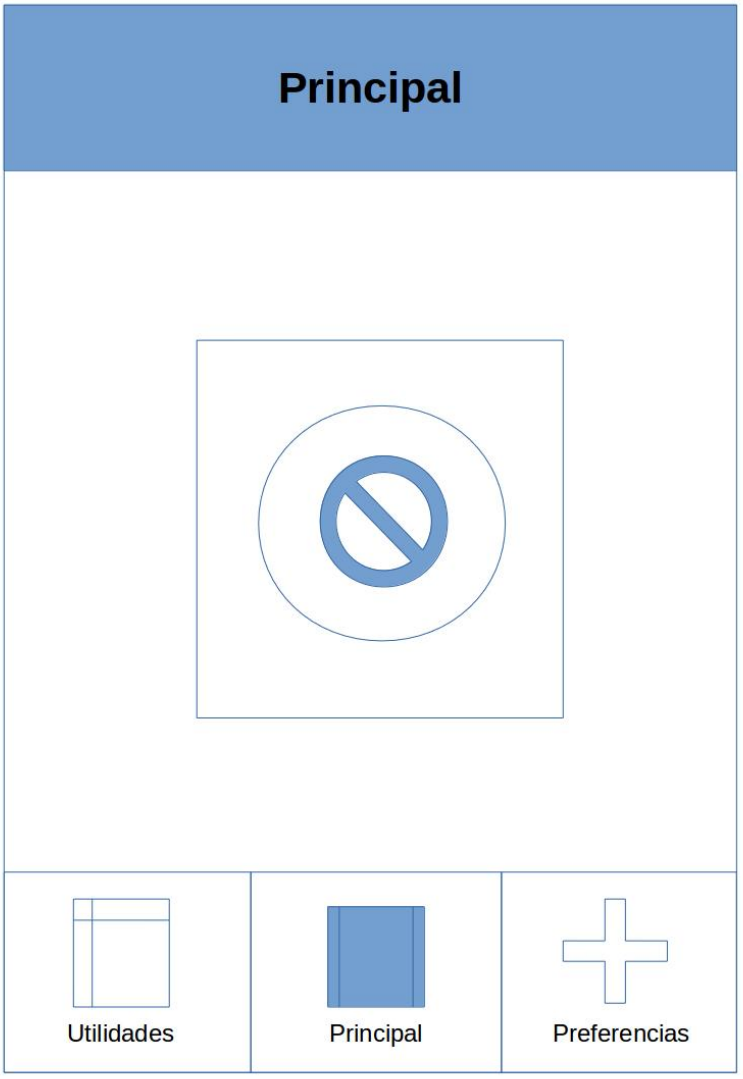
Principal	
Descripción	Página de inicio de la aplicación, contiene un botón que permite el aviso de emergencia instantáneo por parte del usuario, y un pequeño saludo.
Activación	Arranque de la aplicación.
Boceto	
Eventos	Permite el aviso de emergencia instantáneo, y el acceso a las otras dos vistas principales de la aplicación.

Tabla 5.1: Diseño de interfaz: principal.

Utilidades	
Descripción	Página que muestra datos importantes para el usuario, el número de teléfono de contacto y su nombre, datos sobre la siguiente alarma y un acceso a las prescripciones actuales y la pantalla de medicamentos.
Activación	Clic en **Utilidades** en la barra de navegación.
Boceto	
Eventos	Permite modificar el número de teléfono de contacto o el nombre dentro de los label. También permite pulsar al botón "Prescripciones" para ir a la vista con las prescripciones que tiene, y pulsar al botón de Çrear medicamento"para acceder a la vista de medicamentos.

Tabla 5.2: Diseño de interfaz: Utilidades

Preferencias	
Descripción	Vista que muestra los ajustes y preferencias de la aplicación, muestra los datos de usuario y también un botón para abandonar la aplicación. Tiene un apartado con los distintos niveles de detección de caídas a activar.
Activación	Clic en **Preferencias** en la barra de navegación.
Boceto	
Eventos	Permite cerrar sesión al usuario. También permite elegir entre tres niveles distintos para la detección de caídas de usuario pulsando un botón.

Tabla 5.3: Diseño de interfaz: Preferencias

Medicamentos	
Descripción	Ofrece un formulario para dar de alta un nuevo medicamento y su prescripción, introduciendo el formato, y la información de la toma necesaria del mismo
Activación	Click en el botón **Crear nuevo medicamento** en la pantalla de Utilidades.
Boceto	
Eventos	Permite añadir nuevos medicamentos. Permite seleccionar un formato en un RadioListButton, introducir el nombre en un InputLabel, y seleccionar la fecha y las tomas de unos desplegables. También permite retroceder a la pantalla anterior pulsando en la flecha superior izquierda.

Tabla 5.4: Diseño de interfaz: Medicamentos

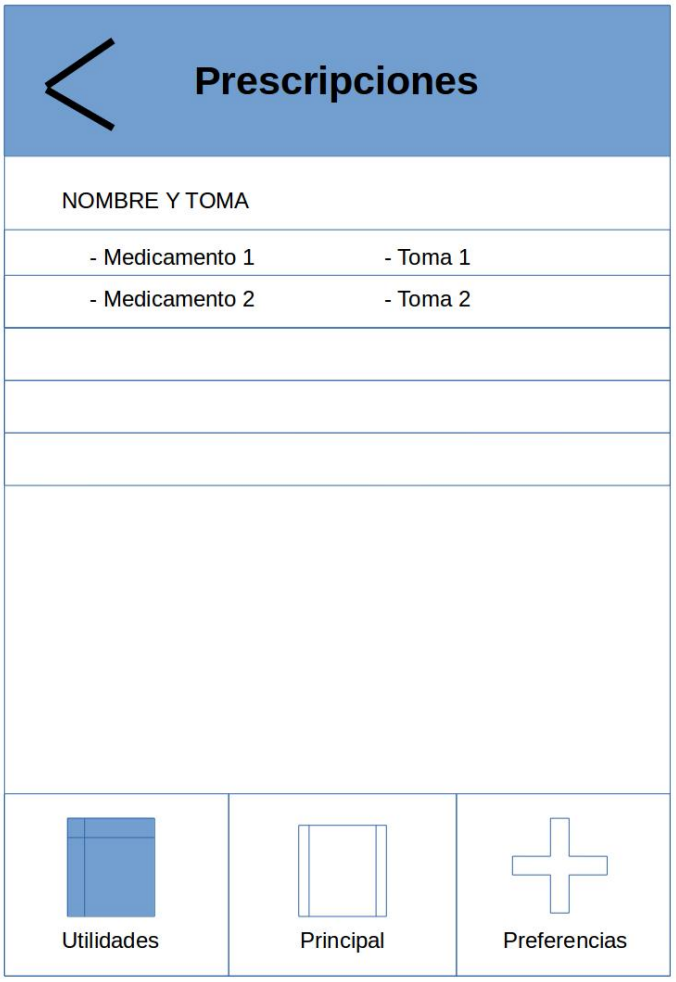
Prescripciones	
Descripción	Muestra un breve listado con los nombres y la siguiente toma de todas las prescripciones que el usuario tiene almacenado en su dispositivo.
Activación	en el botón **Mis Prescripciones** en la pantalla de Utilidades.
Boceto	
Eventos	Muestra un listado de todas las prescripciones del usuario. Únicamente se permite volver a la vista anterior pulsando la flecha de la parte superior izquierda.

Tabla 5.5: Diseño de interfaz: Prescripciones

Capítulo 6

Implementación

En este capítulo se explicarán las diferentes técnicas o tecnologías usadas para poder desarrollar el sistema. Qué es y en que consiste cada una, y la forma en la que se ha implementado la funcionalidad necesitada.

6.1. Phonegap

Phonegap es un paquete de librerías que permite encapsular aplicaciones HTML5 de manera que puedan ser utilizadas como aplicaciónes móviles o WebApps. Se trata de un framework que permite ejecutar aplicaciones desarrolladas en HTML, CSS y JavaScript como si fueran aplicaciones nativas en distintos dispositivos móviles. Phonegap es un producto derivado de Apache Cordova[1], aunque el nombre original fuera Phonegap al ser comprado por Adobe se liberó como Apache Cordova.

Su funcionamiento consiste en un conjunto de librerías o plugins que permiten usar las funcionalidades del teléfono, es decir, no se utiliza lenguaje nativo en la aplicación sino que mediante JavaScript se realizan llamadas a las distintas librerías de Phonegap y estas son las que se ejecutan de manera nativa.

Para poder utilizar Phonegap deberemos realizar una instalación en nuestro sistema apoyado por el gestor de paquetes *npm* y luego dependiendo de la plataforma para la que vayamos a desarrollar deberemos instalar unos u otros componentes(SDK, XCode, etc) para poder compilar para las mismas, todo esto se explica con detalle en el Anexo 1, al final del documento.

6.1.1. SQLite3

La aplicación necesitaba almacenamiento local implementado en *SQLite3* por sus características de tamaño y gestión. Para que la aplicación cree y gestione bases de datos en *SQlite* necesita de una interfaz nativa, es decir, que a través de JavaScript no era posible, por lo que se necesita un plugin para el almacenamiento local: Cordova-sqlite-storage[2]. Este plugin da soporte a diferentes plataformas: Android, iOS, Windows Phone, Amazon FireOS. La conexión a través de base de datos Sqlite tiene sus complicaciones, a parte de por el

[1]https://cordova.apache.org/

[2]https://github.com/litehelpers/Cordova-sqlite-storage

aprendizaje de los comandos básicos para la creación y manejo de base de datos también por que su uso viene forzado por un evento de Phonegap, "deviceready"que salta cuando Phonegap esta cargado en la aplicación completamente, lo que genera problemas cuando nos encontramos con la aplicación en segundo plano. Para añadir el plugin a nuestro proyecto anteriormente creado necesitamos el siguiente comando por terminal dentro del proyecto:

```
$ phonegap plugin add cordova-sqlite-storage
```

Una vez añadido el plugin ya podemos comenzar a utilizar bases de datos SQLite en nuestra aplicación. Necesitaremos llamar al evento de *deviceready* cada vez que vayamos a realizar algun tipo de operación con la base de datos, como se muestra en el ejemplo a continuación:

```
document.addEventListener("deviceready", onDeviceReady, false);

function onDeviceReady() {
//Manejo de base de datos
}
```

A partir de aquí se debe crear una nueva base de datos o en el caso de que ya este creada abrirla, para ello crearemos una variable que sea una llamada al plugin donde indicaremos el nombre y la localización de la misma, que en mi caso es por defecto:

```
var myDB = window.sqlitePlugin.openDatabase({name: "mySQLite.db",
 location: 'default'});
```

Por último con nuestra variable de base de datos ya podemos crear tablas en la base de datos o insertar datos:

```
//Crear tablas
myDB.transaction(function(transaction) {
transaction.executeSql('CREATE TABLE IF NOT EXISTS usuario
 (_id integer primary key, nombre text, apellido text)', [],
function(tx, result) {
//manejas los resultados cuando la consulta tiene exito
alert("Tabla creada correctamente");
},
function(error) {
// la consulta tiene algun error
alert("Error en el proceso de creado de la tabla.");
});
});

//Insertar datos
var nombre="Víctor";
var apellido= "Rujas";

myDB.transaction(function(transaction) {
```

```
var executeQuery = "INSERT INTO usuario (nombre, apellido) VALUES  (?,?)";
transaction.executeSql(executeQuery, [nombre,apellido],
function(tx, result) {
alert('Dato insertado');
},
function(error){
alert('Error en la inserción');
});
});
```

El caso de insertar los datos es similar a una consulta del tipo *SELECT* o del tipo *UPDATE* por ejemplo.

6.1.2. Acelerómetro

El acelerómetro es un dispositivo que mide la aceleración y la fuerzas inducidas por la gravedad. Es un tipo de sensor que suele venir integrado en la mayoría de los smartphones. En la aplicación móvil necesitamos medir la aceleración de nuestro teléfono para poder detectar el momento en el que el usuario, con el móvil encima, sufre alguna caída o accidente. Para ello debemos acceder al sensor a través de un plugin de Phonegap: cordova-plugin-device-motion[3]. Este plugin necesita una configuración para poder acceder adecuadamente al sensor.

Primero debemos añadir el plugin al proyecto por la linea de comandos:

```
$ phonegap plugin add cordova-plugin-device-motion
```

Una vez añadido el plugin debemos configurar algunos archivos del proyecto para poder acceder al sensor, dependiendo de la plataforma será de una manera u otra. En el caso de Android e iOS se debe editar el archivo *config.xml* añadiendo lo siguiente:

Para Android:

```
 <feature name="Accelerometer">
    <param name="android-package"
     value="org.apache.cordova.devicemotion.AccelListener" />
</feature>
```

Para iOS:

```
 <feature name="Accelerometer">
    <param name="ios-package" value="CDVAccelerometer" />
</feature>
```

A partir de ahora ya se puede hacer la llamada a cualquiera de los tres métodos del acelerómetro. En mi caso uso el *watchAcceleration* que muestra a tiempo real la aceleración del dispositivo a tiempo real por eje(X, Y, Z). Primero es necesario dejar que el dispositivo cargue todas las librerías de la API de Phonegap mediante una llamada al evento *deviceready* y cuando este este cargado ya se puede acceder al sensor, este ejemplo de la documentación de la API de Phonegap muestra como hacerlo muy básicamente:

Variable que se encargar de rastrear el valor del método watchAcceleration.

[3]https://github.com/apache/cordova-plugin-device-motion/blob/master/doc/es/index.md

```
var watchID = null;
```

Llamada al evento *deviceready* y a la función que se ejecutará cuando este este correcto

```
document.addEventListener("deviceready", onDeviceReady, false);

function onDeviceReady() {
    startWatch();
}
```

La función *startWatch()* se encarga de establecer los distintos parámetros para el detectar el valor del acelerómetro y la frecuencia

```
function startWatch() {
var options = { frequency: 3000 };
watchID = navigator.accelerometer.watchAcceleration(onSuccess,
 onError, options);
}
```

La función *onSuccess()* mostrará los valores de la aceleración

```
function onSuccess(acceleration) {
var element = document.getElementById('accelerometer');
alert('Acceleration X: ' + acceleration.x
+ 'Acceleration Y: ' + acceleration.y
+ 'Acceleration Z: ' + acceleration.z) ;
}
```

En el caso de que surja algún tipo de error a la hora de activar el sensor del dispositivo también hay que capturar el error.

```
function onError() {
alert('Ha ocurrido un error con el acelerómetro');
}
```

6.1.3. GPS

El sistema de posicionamiento global GPS es un sistema que permite determinar la posición de un objeto con precisión de metros en el caso de los dispositivos móviles. Para registrar la posición GPS del dispositivo móvil mediante Phonegap debemos añadir el plugin: cordova-plugin-geolocation[4]. Para añadir el plugin al proyecto debe ser por la linea de comandos.

```
$ phonegap plugin add cordova-plugin-geolocation
```

Para poder utilizar ese plugin en nuestro proyecto debemos editar el archivo *config.xml*. En el caso de Android

[4]https://github.com/apache/cordova-plugin-geolocation

```
<feature name="Geolocation">
<param name="android-package"
 value="org.apache.cordova.geolocation.GeoBroker" />
</feature>
```

En el caso de iOS

```
<feature name="Geolocation">
<param name="ios-package" value="CDVLocation" />
</feature>
```

Android es una plataforma especial en este aspecto, por que lo que se debe añadir permisos a la aplicación en el archivo */platforms/android/AndroidManifest.xml*

```
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

El caso del GPS requiere del mismo evento que el acelerómetro, el *deviceready* ya que debe cargar todas las librerías de la API de Phonegap.

El método *watchPosition* registra los cambios en la posición del dispositivo. Para para lanzarlo cuando el evento *deviceready* este listo se configuran la frecuencia de actualización del método.

```
function onDeviceReady() {
var options = { timeout: 30000 };
watchID1 = navigator.geolocation.watchPosition(onSuccess,
 onError, options);
}
```

La llamada a la función *onSuccess()* mostrará los resultados de la llamada a watchPosition que se actualizará en función de los parámetros anteriormente fijados.

```
function onSuccess(position) {
alert('Latitude: '  + position.coords.latitude+
  'Longitude: ' + position.coords.longitude);
}
```

Para poder activar adecuadamente, y en el caso de que surja algún error, el GPS del dispositivo se captura el error.

```
function onError(error) {
alert('code: '    + error.code    + '\n' +
  'message: ' + error.message + '\n');
}
```

6.1.4. Alarmas

La aplicación debe gestionar las prescripciones de medicamentos que el usuario puede llegar a necesitar. Cada toma de medicamento por parte del usuario deberá ser notificada de alguna manera, en este caso mediante notificaciones push, para ello se añade al proyecto el plugin: de.appplant.cordova.plugin.local-notification[5]. Para añadir el plugin al proyecto se debe añadir mediante la línea de comandos.

```
$ phonegap plugin add de.appplant.cordova.plugin.local-notification
```

Este plugin genera notificaciones de manera local en el dispositivo, con soporte para plataformas Android, iOS, y Windows Phone. Hay que tener en cuenta que se debe registrar los permisos para generar push, posteriormente crear el evento de la notificación push y lanzarlo. El siguiente código muestra un ejemplo para generar y lanzar de manera simple y efectiva.

```
cordova.plugins.notification.local.schedule({
     id: 1,
     title: "Título del mensaje",
     message: "Texto del mensaje",
});

cordova.plugins.notification.local.on("click",
 function (notification, state) {
  window.location = url;
 }, this);
});
```

A partir de aquí el problema surge en la ejecución de la push en el momento en el que el usuario necesite la toma. Para poder lanzarla necesitamos recuperar el momento en el que se debe notificar al usuario y la información relativa al medicamento, por ejemplo el nombre. Ahí surge un problema ya que el acceso a base de datos viene gestionado por el evento de Phonegap *deviceready* que en el caso de que la aplicación se encuentre en segundo plano ese evento ya no esta activo y el estado pasa a *pause*. Hasta que no salte el evento *onresume* no se volverán a cargar las librerías de la API de Phonegap para poder acceder a SQLite. Ya que gestionar adecuadamente el acceso a base de datos tiene la complicación de tener activo el evento *deviceready*, hay que controlar el salto de esos tres eventos. En el momento en el que sea la hora de la notificación al usuario la aplicación a su vez debe realizar una consulta *UPDATE* en la base de datos por lo que el evento *deviceready* debe estar activo y capturado para poder lanzar la consulta SQLite.

El momento del lanzamiento de la notificación viene dato por fechas en formato *Date* de JavaScript guardado en milisegundos contando desde la media noche del día 1 de Enero de 1970, y las notificaciones se lanzan manejando la diferencia en milisegundos de las fechas.

[5]https://github.com/katzer/cordova-plugin-local-notifications

6.1.5. Llamadas

La ejecución de llamadas telefónicas mediante Phonegap se ha realizado mediante un plugin que permite la llamada directamente desde tu aplicación. El plugin es: call-number[6]. Para añadirlo al proyecto hay que hacerlo a través de la línea de comandos:

```
$Phonegap plugin add call-number
```

Es un plugin que permite un acceso muy fácil a la interfaz de llamada de cualquier dispositivo en una sentencia javascript.

```
window.plugins.CallNumber.callNumber(onSuccess, onError, number,
 bypassAppChooser);
```

6.2. Interfaz de usuario: ReactJS y touchstoneJS

A la hora de desarrollar interfaces de usuario para una aplicación Phonegap hay que tener en cuenta que se basan en páginas web embebidas en una aplicación. Para la realización de la interfaz en la aplicación había una gran variedad, pero la que mejores características de velocidad y potencia ofrecía según mi criterio era React JS[7]. ReactJS es una biblioteca escrita en JavaScript, desarrollada por Facebook para facilitar la creación de componentes interactivos y reutilizables para interfaces de usuario. Esta construido para hacer funciones que tomen las actualizaciones de estado de la página y que se traduzcan en una representación virtual de la página resultante. Cada vez que surge un cambio React modifica el DOM para reflejar la nueva representación de la página. Esto lo dota de una mayor agilidad que el JavaScript. Para gestionar interfaces de usuario existen varios frameworks basasdos en distintos lenguajes, siguiendo mi elección y basándome en ReactJS, he elegido TouchstoneJS[8], que se define como un framework de interfaces de usuario para desarrollar bonitas aplicaciones híbridas. TouchstoneJS ofrece una alta variedad de características: componentes de interfaz de usuario, gestión del estado de la aplicación, transición entre vistas, animaciones optimizadas, y lo más interesante, componentes customizables a gusto.

Aunque no existe ningún manual de manera oficial ni nada parecido, los creadores nos ofrecen la única forma de empezar a utilizar el framework, clonar el repositorio de *Github* del TouchstoneJS[9] proyecto starter de TouchstoneJS e ir modificándolo a gusto. Es un proyecto bastante elaborado y largo por lo que hay que invertir bastante tiempo para modificarlo a gusto.

6.3. Netbeans: JavaEE y GlassFish Server

Para que el sistema web y el servidor de base de datos pueda funcionar correctamente, deberá ser desplegado en un servidor de aplicaciones, que se encargar de albergar los distintos tipos de componentes que forman las aplicaciones web. Gestionan la mayor parte de la lógica de negocio y de acceso a datos de la aplicación.

[6]https://github.com/Rohfosho/CordovaCallNumberPlugin
[7]https://facebook.github.io/react/
[8]http://touchstonejs.io/
[9]https://github.com/touchstonejs/touchstonejs-starter.git

En este caso, se ha utilizado el servidor de aplicaciones *GlassFish Server*, la versión 4.1 que es la más estable. Se decidió por la utilización de dicho servidor a causa de que viene integrado con la herramienta NetBeans, que es un IDE utilizado para el desarrollo web con el que se ha tenido contacto con la asignatura de Plataformas de Software Empresarial de tercer año.

La aplicación web ha sido desarrollada en JavaEE, lo que permite multitud de ventajas y funcionalidades, ya que esta orientado a desarrollo empresarial. La parte correspondiente con la interfaz ha sido desarrollada con JSP(Java Server Pages) y JSF(con el complemento de primefaces), CSS y JavaScript. A la hora de acceder al la base de datos, no se ha hecho directamente, sino que se ha implementado unos servicios REST que facilitan el acceso y la manipulación de datos a través de peticiones HTTP, que a su vez permite el acceso a partir de la aplicación móvil.

Capítulo 7

Pruebas

En este capítulo se engloban las numerosas pruebas realizadas durante y a posteriori de la implementación de las dos aplicaciones que forman el sistema. Se han realizado dos tipos de pruebas: de caja blanca y de caja negra.

7.1. Pruebas de caja blanca

Las pruebas de caja blanca son un tipo de test estructurales, que se enfocan sobre todo en las funciones de cada script desarrollado por separado, gestionando pruebas a bajo nivel para asegurar el correcto funcionamiento en diferentes situaciones. Son pruebas que se han realizado simultáneamente a la implementación del sistema y se han basado en las siguientes comprobaciones:

- Verificación de la conexión a la base de datos SQLite mediante el plugin de Phonegap en la aplicación móvil.
- Verificación de la conexión de base de datos SQL en el servidor web.
- Comprobación mediante logs del funcionamiento de los servicios REST, mediante el uso de EJBs en la aplicación web y de peticiones HTTP en la aplicación móvil.
- Verificación de los posibles caminos surgidos a partir de bucles y llamadas recursivas en la ejecución del script.
- Comprobación de que no halla fallos en el encadenamiento de vistas en la aplicación móvil y de que no haya enlaces fallidos en la aplicación web.
- Verificación de la agilidad y un flujo correcto en la navegación a través de la aplicación en un dispositivo Android.

7.2. Pruebas de caja negra

Las pruebas de caja negra , también llamadas pruebas funcionales, se basan en verificar los valores producidos por el sistema cuando recibe distintos tipos de valores de entrada. Es decir, que buscan comprobar que las funcionalidades del sistema satisfacen correctamente los

distintos requisitos establecidos en el análisis previo. A continuación se detallan las pruebas que han resultado más relevantes:

PN-01: Registrar usuario	
Objetivo	Probar que el formulario de registro de usuario funciona correctamente.
Precondiciones	El usuario no debe estar identificado.
Datos de entrada	Campos del formulario de registro.
Acción esperada	El usuario queda registrado en el sistema correctamente y ya puede iniciar sesión tanto en la aplicación móvil como web. Usuario creado en la base de datos del servidor.
Resultado	Correcto

Tabla 7.1: Prueba de Caja Negra 01 · Registrar usuario

PN-02: Registrar usuario incorrecto	
Objetivo	Probar que el formulario de registro de usuario muestra los errores oportunos.
Precondiciones	El usuario ya registrado con anterioridad.
Datos de entrada	Campos del formulario de registro.
Acción esperada	Se muestra una alerta que avisa al usuario sobre que los datos introducidos son incorrectos ya que el usuario ya existe.
Resultado	Correcto

Tabla 7.2: Prueba de Caja Negra 02 · Registrar usuario incorrecto

PN-03: Login de usuario	
Objetivo	Probar que el formulario de login de usuario funciona correctamente en ambas plataformas.
Precondiciones	Usuario ya registrado con anterioridad.
Datos de entrada	Campos del formulario de login.
Acción esperada	Se muestra la pantalla principal de la aplicación móvil y de la aplicación web, con la sesión ya iniciada.
Resultado	Correcto

Tabla 7.3: Prueba de Caja Negra 03 · Login de usuario.

PN-04: Login de usuario incorrecto	
Objetivo	Probar que el formulario de login de usuario devuelve adecuadamente los errores.
Precondiciones	Usuario no autenticado con anterioridad
Datos de entrada	Campos del formulario de login.
Acción esperada	Se muestra una pantalla donde se avisa que los datos son incorrectos
Resultado	Correcto

Tabla 7.4: Prueba de Caja Negra 04 · Login de usuario incorrecto.

PN-05: Actualizar datos de contacto de usuario	
Objetivo	Comprobar que la información de usaurio
Precondiciones	Usuario logueado correctamente en la aplicación móvil.
Datos de entrada	Un número de teléfono Un nombre de contacto.
Acción esperada	Que se registren dichos datos en las preferencias compartidas de la aplicación móvil
Resultado	Correcto

Tabla 7.5: Prueba de Caja Negra 05 · Actualizar datos de contacto

PN-06: Elegir nivel de detección de caídas	
Objetivo	Modificar los valores de rastreo del sensor del acelerómetro.
Precondiciones	Usuario logueado correctamente en la aplicación móvil.
Datos de entrada	Tres valores, x, y, z.
Acción esperada	Que se modifique la detección de caídas y sus variables
Resultado	Correcto

Tabla 7.6: Prueba de Caja Negra 06 · Elegir nivel de detección de caídas

PN-07: Cerrar sesión	
Objetivo	Probar que la función de cerrado de sesión funciona correctamente tanto en la aplicación móvil o web.
Precondiciones	Usuario abandona la sesión correctamente
Datos de entrada	Ninguno
Acción esperada	Que se modifique la detección de caídas y sus variables
Resultado	Correcto

Tabla 7.7: Prueba de Caja Negra 07 · Cerrar sesión

PN-08: Crear medicamentos	
Objetivo	Probar que el formulario de creación de medicamentos funciona correctamente.
Precondiciones	Sesión iniciada por el usuario en la aplicación móvil.
Datos de entrada	Los requeridos por el formulario.
Acción esperada	Prescripción, medicamento y alarma creado
Resultado	Correcto

Tabla 7.8: Prueba de Caja Negra 08 · Crear medicamento

PN-09: Crear medicamento incorrecto	
Objetivo	Probar que el formulario de creación de medicamentos funciona correctamente.
Precondiciones	Sesión iniciada por el usuario en la aplicación móvil.
Datos de entrada	El nombre facilitado para el medicamento ya esta dado de alta en la base de datos del dispositivo móvil.
Acción esperada	El sistema muestra una alerta y saca al usuario de la pantalla de medicamento
Resultado	Correcto

Tabla 7.9: Prueba de Caja Negra 09 · Crear medicamento incorrecto

Capítulo 8

Manuales

Este capítulo engloba el apartado de los manuales del sistema, inlcuyendo los manuales de instalación de la aplicación móvil y web, y de los manuales de usuario.

8.1. Manual de instalación

El sistema esta compuesto de dos aplicaciones: móvil y web, y para poder ejecutarlas adecuadamente necesitaremos los siguientes manuales de instalación.

8.1.1. Aplicación móvil híbrida

La aplicación móvil esta orientada para que sea multiplataforma y sea adaptable para varios tipos de plataformas. En este caso la principal es Android, para iOS se espera que sea posible en un futuro próximo. En el caso de Android solo será necesario la instalación del archivo con formato APK. Para ello el usuario debe permitir en los ajustes de seguridad de su teléfono la instalación de aplicaciones de orígenes desconocidos ya que no esta subida a una plataforma oficial como es la tienda de aplicaciones de Google. Durante el proceso de instalación deberá aceptar los permisos necesarios y finalizar. En casos excepcionales como en los dispositivos con Android 7.0 y superior te dejará elegir que permisos aceptar o no al mismo tiempo que irás usando la aplicación, se recomienda aceptar todos para que el funcionamiento de la aplicación sea correcto. Para que la aplicación pueda iniciar sesión correctamente se necesitará la instalación de la aplicación web y de su servidor de base de datos ya que requiere inicio de sesión la primera vez que inicies a la aplicación.

8.1.2. Aplicación web y servidor de aplicaciones

Una aplicación web no necesita instalarse como tal en el dispositivo en el que vaya a ejecutarse, pero requiere de otro tipo de instalaciones, como alojarse en un servidor web o servidor de aplicaciones. Para poder almacenar datos también requerirá de un servidor de base de datos. En este caso se ha utilizado de servidor de aplicaciones *GlassFish* y de servidor de base de datos *MySQL*. En primer lugar comenzaremos instalando MySQL Server. Según el sistema operativo se deben seguir unos pasos u otros, en mi caso la instalación sobre la que se han realizado las pruebas ha sido windows 10.

- Primero se debe descargar el archivo de instalación de MySQL Server, en mi caso recomiendo la descarga de XAMPP[1] que es un gestor de distintos tipos de base de datos y te realiza la instalación de MySQL Server, entre otros, de una manera fácil y rápida, y es de código abierto.

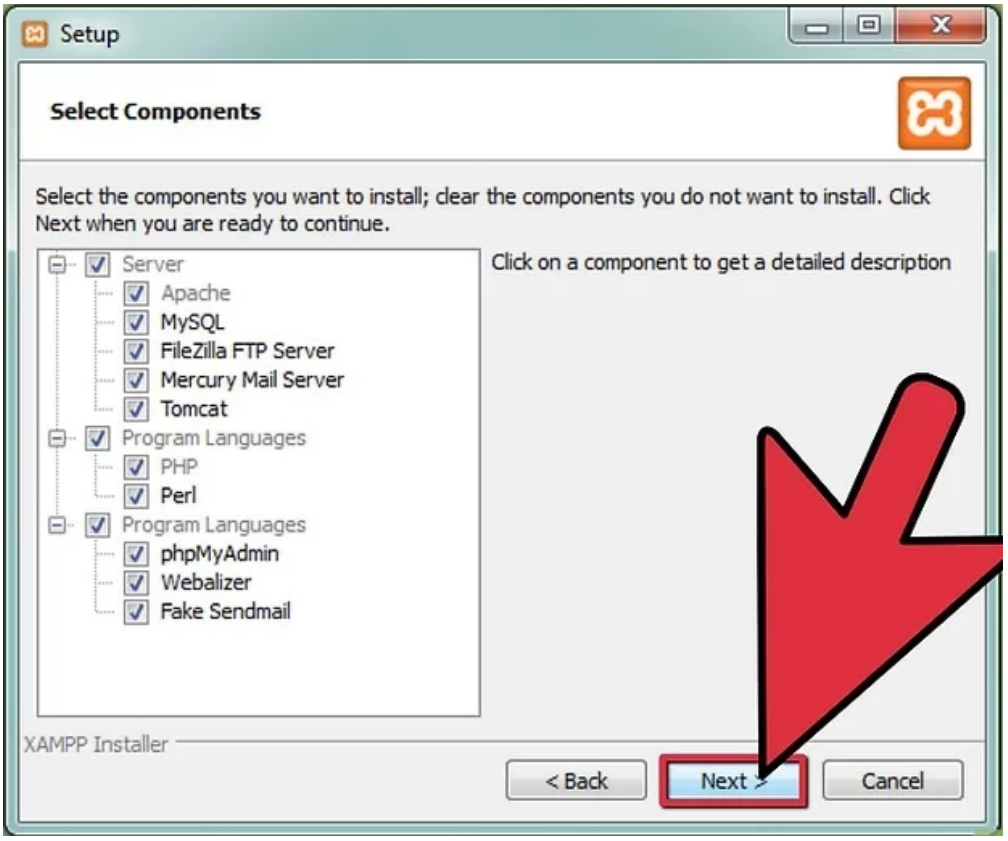


Figura 8.1: Instalación XAMPP - 1

[1]la descarga se realiza desde la página de Apache Friends http://www.apachefriends.org/en/xampp-windows.html

Figura 8.2: Instalación XAMPP - 2

- En el momento en el que XAMPP haya terminado la instalación, es hora de abrir el panel de control y ejecutar el servidor de MySQL y Apache para poder gestionar las bases de datos a través de la herramienta *phpMyAdmin*:

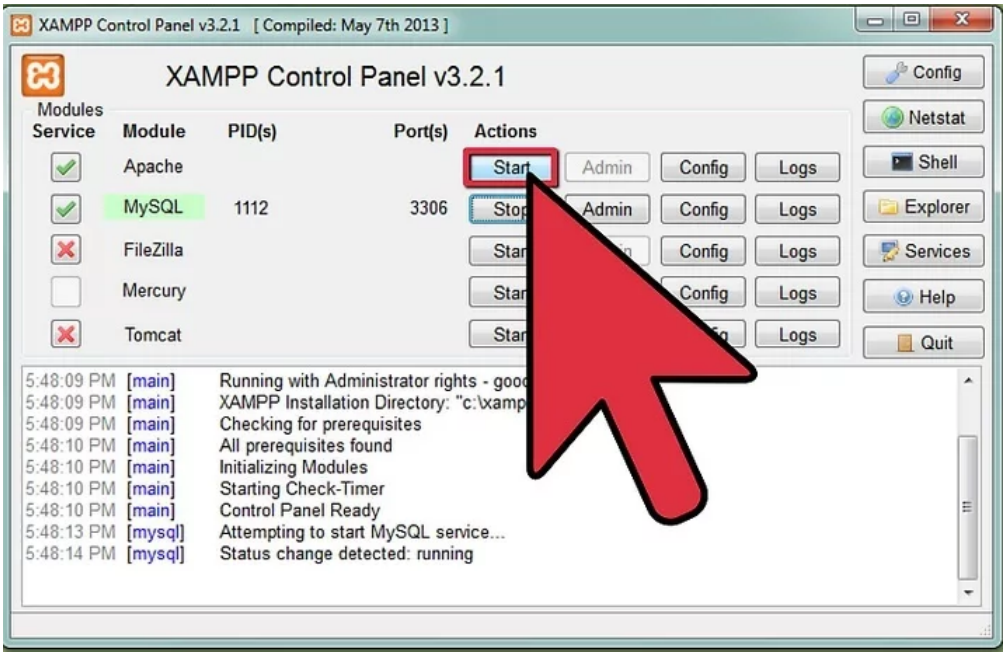


Figura 8.3: Instalación XAMPP - 3

- Se deberá tener en cuenta la contraseña asignada al usuario root en MySQL, ya que será importante para poder acceder al mismo.
- En el momento en el que el MySQL server este corriendo se podrá importar la base de datos en formato .sql.
- Para la instalación del servidor GlassFish deberemos descargar el paquete de instalación de su página, https://glassfish.java.net/download.html.
- Una vez descargado, se descomprime el fichero en nuestro sistema de archivos.
- Para ejecutar el servidor en Windows 10 es necesario ejecutar un par de órdenes desde la consola de comandos o cmd.
- Primero deberemos situarnos en el directorio donde se descomprimió el archivo.
- Para iniciar el servidor deberemos ejecutar la siguiente orden:

```
glassfish4.1\glassfish\bin asadmin start-domain
```

- Para configurar la aplicación web junto con el servidor GlassFish, deberemos ejecutar la herramienta de administración del servidor, que se encuentra en la siguiente dirección web: http://localhost:4848/. El login de inicio por defecto es *admin* tanto para el usuario como para la contraseña.

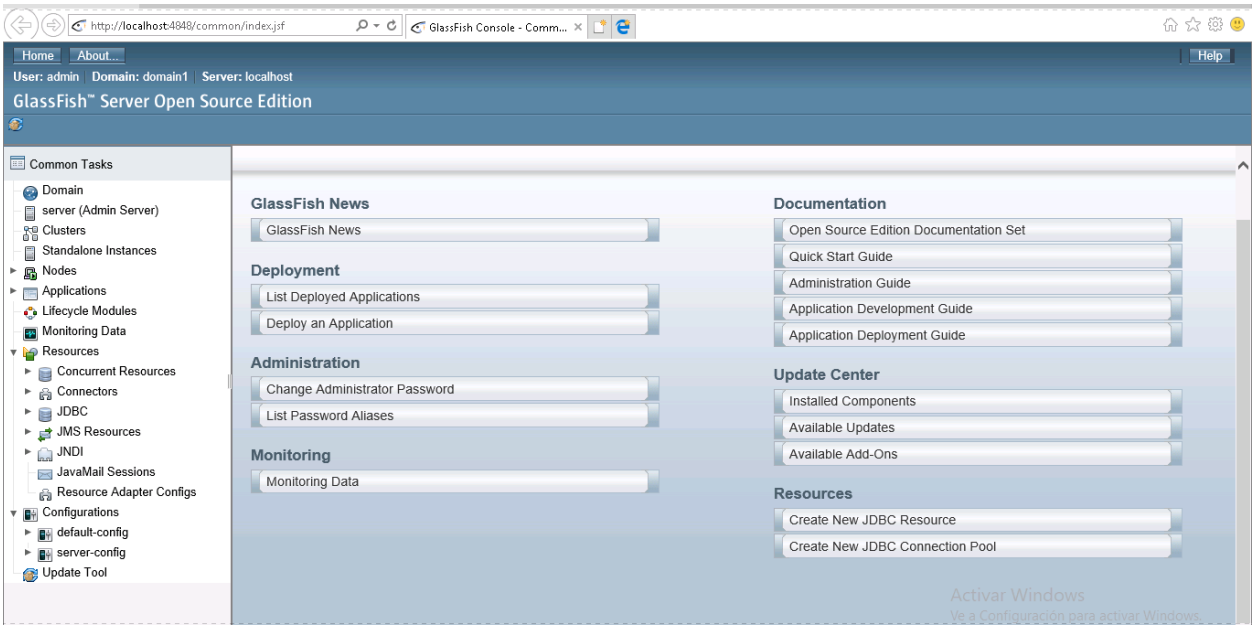


Figura 8.4: Instalación GlassFish - 1

- En esta pantalla se debe seleccionar la opción *Deploy* posteriormente la opción *Package File to be Uploaded to the server* y seleccionamos el archivo de aplicación web .war. Por último debe volver a la pantalla *Applications* y escribir el nombre del proyecto en el apartado de *Application Name* y guardar los cambios.

- A continuación se va a configurar la autenticación JAAS que se encarga de gestionar toda la seguridad de las comunicaciones de la aplicación web, para lo que deberemos aplicar las siguientes configuraciones:

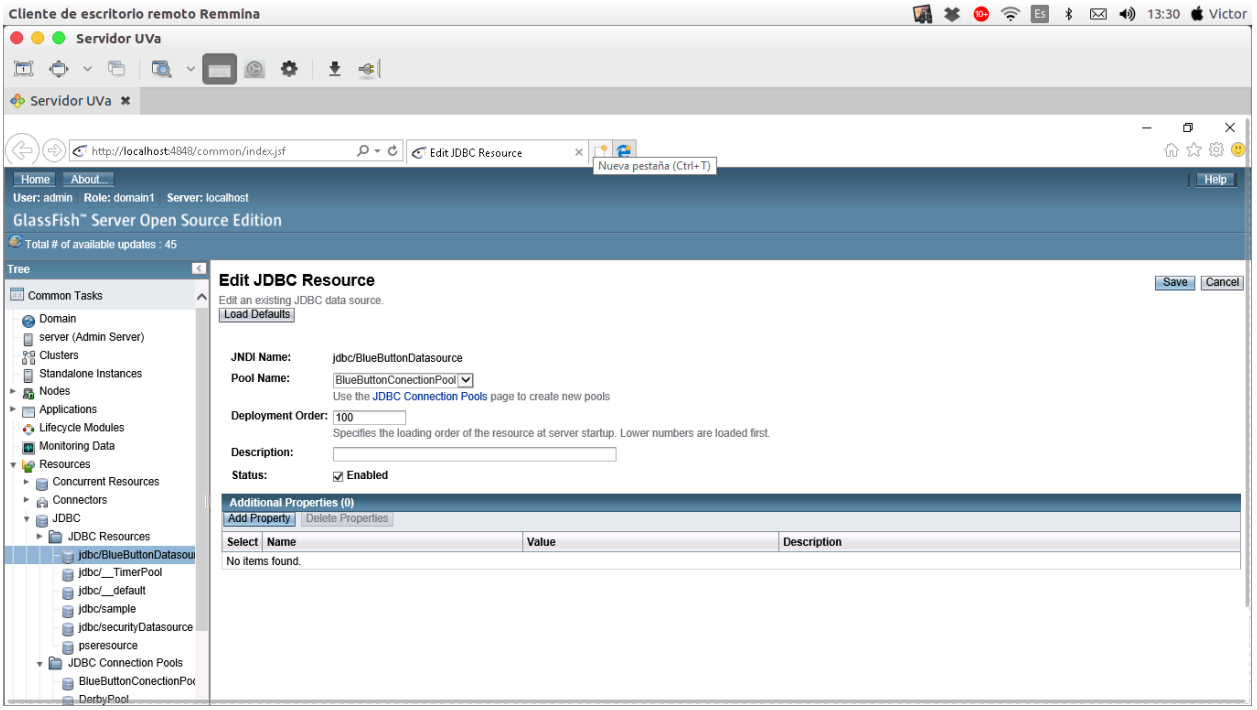


Figura 8.5: Configuración JAAS - Crear recurso JDBC

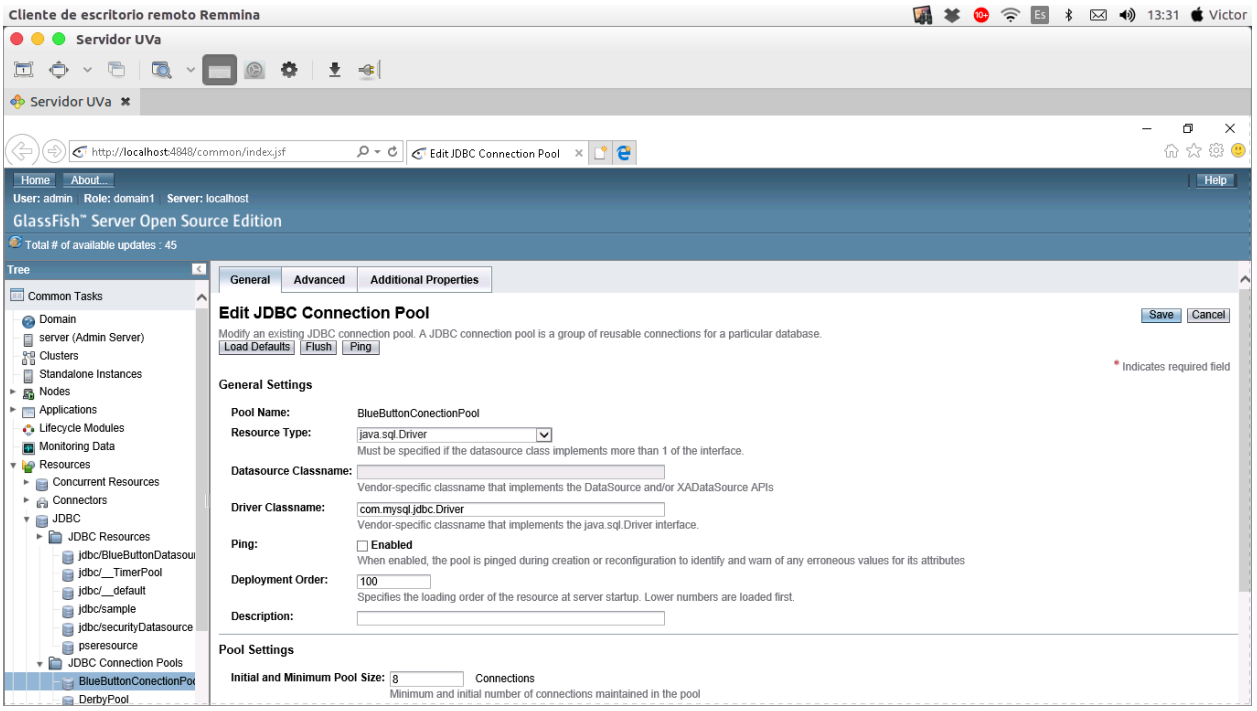


Figura 8.6: Configuración JAAS - Crear pool de conexión

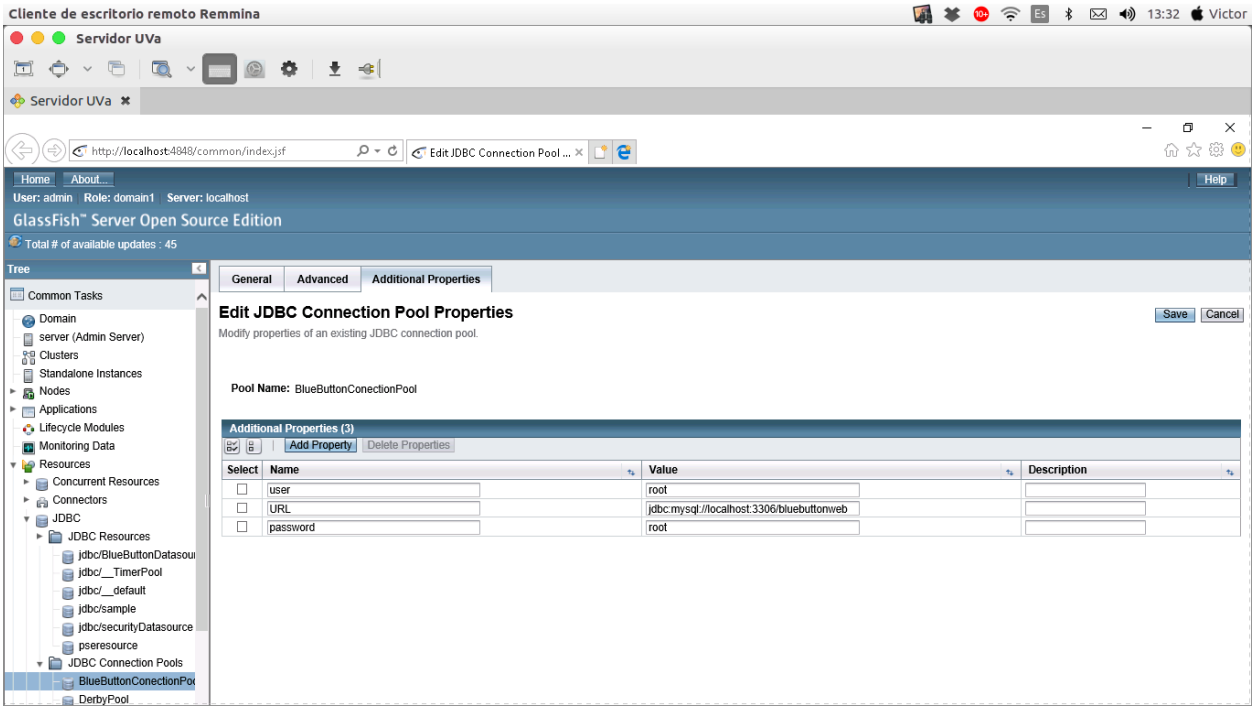


Figura 8.7: Configuración JAAS - Propiedades pool de conexión

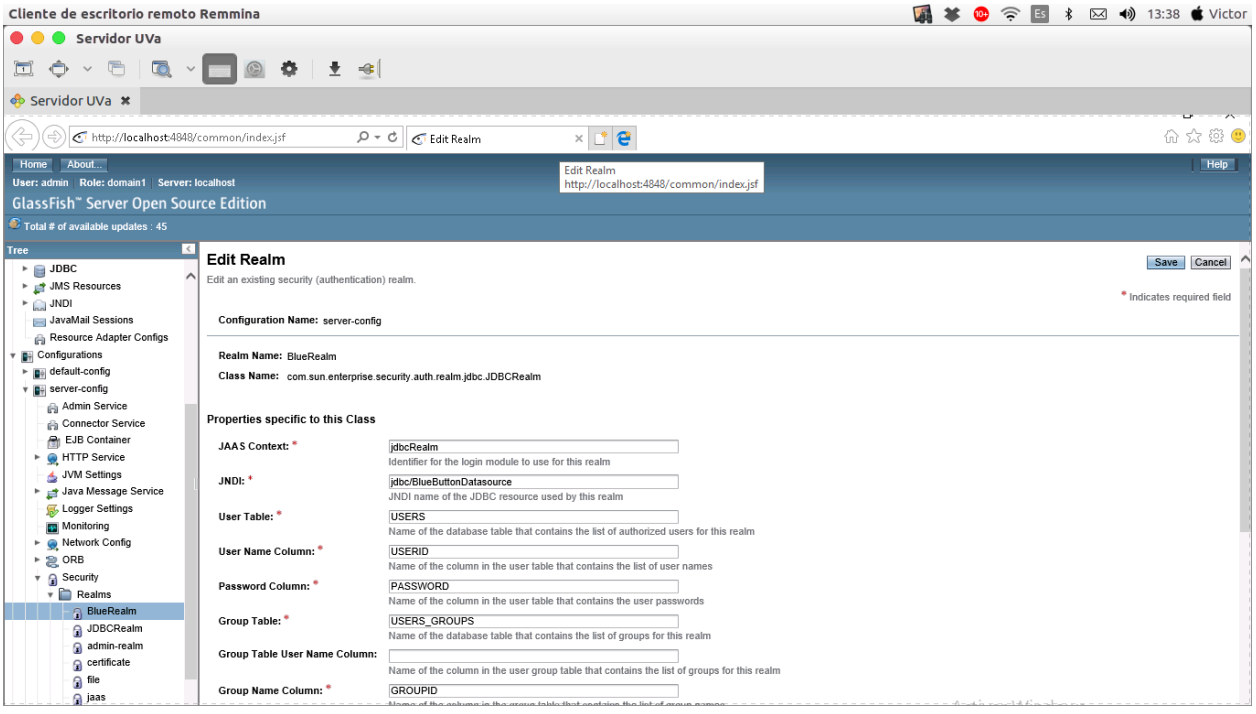


Figura 8.8: Configuración JAAS - Crear realm

- Para ejecutar la aplicación a través del servidor de aplicaciones se deberá dirigir a la dirección: http://localhost:8080/, seguido del nombre del proyecto.

8.2. Manual de usuario

Para facilitar el uso y el aprendizaje de los usuarios a la hora de manejar el sistema, existen dos manuales de usuario: el manual de usuario de la aplicación móvil y el de la aplicación web.

8.2.1. Manual de usuario aplicación móvil

Es muy recomendable que el futuro usuario móvil sea supervisado a la hora ejecutar la aplicación por primera vez para asegurar la correcta configuración de la misma, también a su vez es recomendable que también sea supervisado a la hora de añadir prescripciones al sistema. Es de alta importancia que al arranque de la aplicación en caso de tratarse de Android 7.0+ se le concedan todos los permisos solicitados. También el usuario deberá añadir la aplicación a la lista de excepciones para que aunque la pantalla se bloquee la aplicación en si no se cierre para permitir notificar de tomas de medicamento y almacenar la posición en el servidor web. El posible usuario que desee tener acceso a la aplicación móvil deberá estar registrado en la aplicación web con anterioridad, ese paso se mostrará en el punto 8.2.2[2]. En todo momento el usuario encontrará un elemento fijo en la aplicación, la barra de navegación inferior, que le permitirá moverse por las vistas principales de la aplicación:

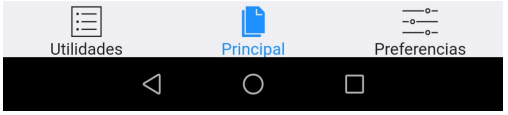


Figura 8.9: Barra de navegación - Aplicación móvil

[2]Manual de usuario aplicación web

Cuando el usuario acceda a la aplicación móvil se encontrará con la pantalla para iniciar sesión:

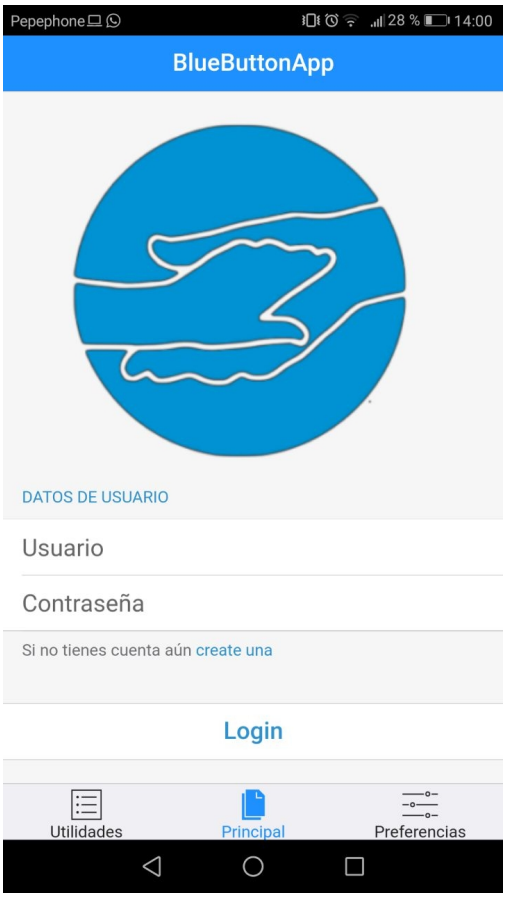


Figura 8.10: Pantalla de login - Aplicación móvil

En ella deberá rellenar los datos de usuario(correspondiente con su número de móvil) y su contraseña para después poder pulsar al botón de Login, que le permitirá iniciar sesión en la aplicación si lo datos introducidos son correctos, en el caso de que lo sean o no el sistema mostrará una alerta con el resultado. Si no dispone de cuenta justo debajo del campo de contraseña aparece un enlace que dice *créate una* que te dirigirá al navegador web para poder realizar el registro de usuario.

A partir del momento en el que el usuario ya haya iniciado sesión en el dispositivo móvil, no tendrá que volver a iniciar sesión, a no ser que la abandone él mismo. Cuando inicie la aplicación verá la siguiente pantalla:

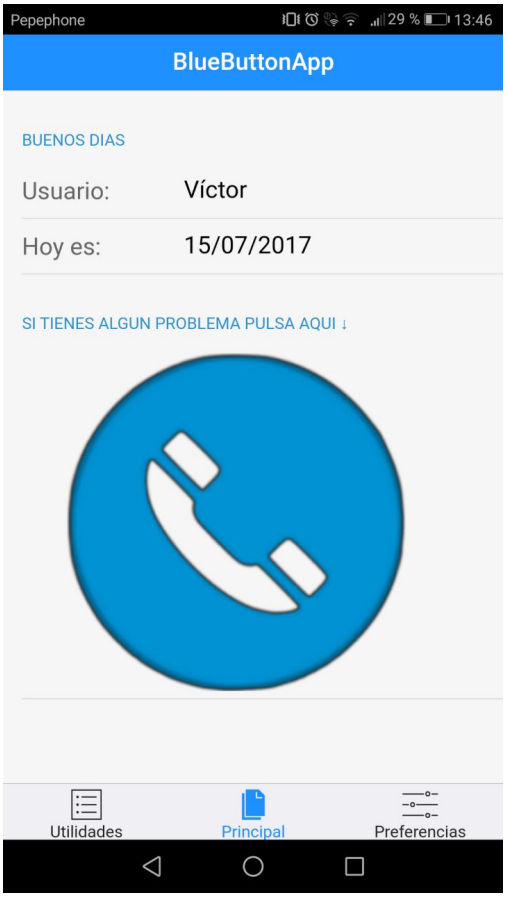


Figura 8.11: Pantalla Principal - Aplicación móvil

En la pantalla **Principal** de la aplicación, el usuario podrá realizar una llamada de emergencia a su usuario, previamente configurado, y ver un pequeño saludo indicando el día de hoy.

La primera vez que el usuario inicie sesión, será recomendable que se dirija a la pantalla de preferencias de utilidades de la aplicación e inserte los datos de su contacto de emergencia, para poder disponer de la funcionalidad de llamada de emergencia. Para ello deberá acceder a la pantalla de **Utilidades** a través de la barra de navegación, lo que le mostrará lo siguiente:

Figura 8.12: Pantalla de Utilidades - Aplicación móvil

En dicha pantalla se podrán editar los campos que aparecen en el grupo **Tu contacto**, tanto el nombre como el número de teléfono. Así cuando se realice alguna llamada de emergencia, la aplicación la dirigirá a dicho contacto. A su vez en esa misma pantalla aparece el nombre y la fecha de la próxima toma de medicamento que tenga el usuario registrada, y dos botones. El primer botón **Ver mis prescripciones** lanza una nueva vista en la que se muestra un pequeño listado de las prescripciones que tiene el usuario registradas en la aplicación y de la siguiente hora de toma de las mismas, sería la siguiente:

Figura 8.13: Pantalla Mis Prescripciones - Aplicación móvil

El segundo botón **Crear nuevo medicamento** carga una nueva vista en la que se muestra un formulario para la creación de prescripciones médicas:

Figura 8.14: Pantalla Medicamento - Aplicación móvil

Donde se podrán rellenar todos los datos, tanto como el formato de medicamento, información sobre la toma (nombre, número de tomas diarias y días de la medicación) y por último elegir una hora a la que inicie la prescripción, en caso de que se elija una hora anterior a la actual el sistema supondrá que la toma empieza al día siguiente. Cuando todos los campos estén rellenos correctamente pulsando en el botón de **Crear recordatorio** registraremos la nueva prescripción en la aplicación y crearemos la próxima alarma de toma relacionada con la prescripción, para asegurar que la creación se a realizado correctamente el sistema lo notificará con una notificación push:

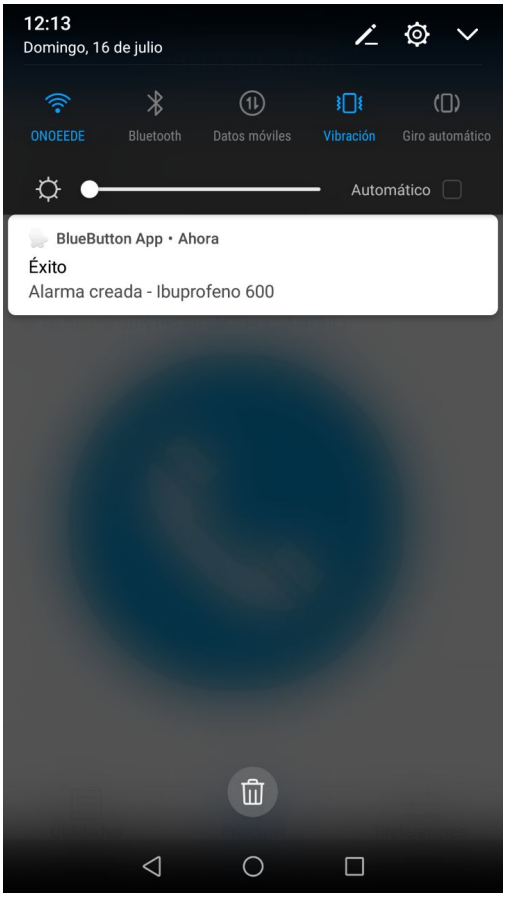


Figura 8.15: Notificación push creación de alarma - Aplicación móvil

Cada vez que se llegue la hora de alarma de alguna toma, el sistema lo notificará al usuario mediante notificaciones push:

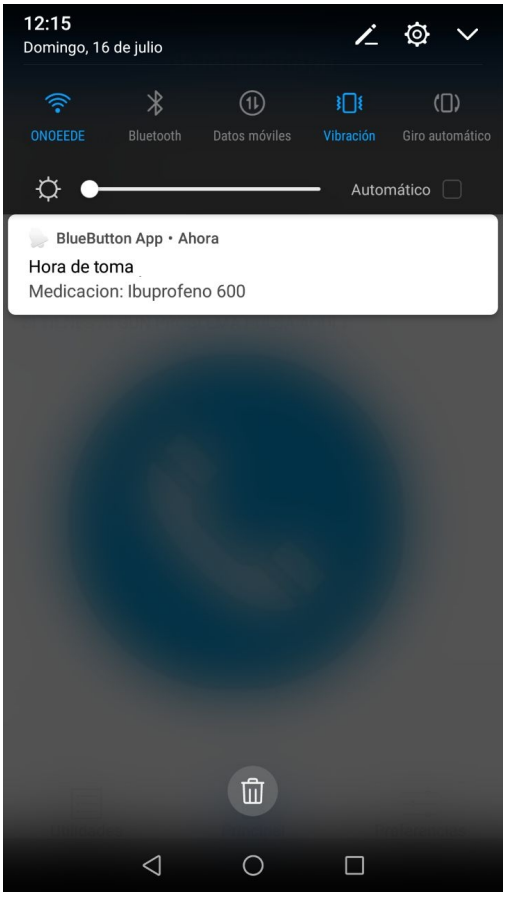


Figura 8.16: Notificación push hora de toma - Aplicación móvil

En la pantalla de **Preferencias**, a la que se accede a través de la barra de navegación, aparecen datos de usuario, tales como el nombre y el número de teléfono, la opción de cerrar sesión, y la opción de elegir un la sensibilidad de detección de caídas en función de las preferencias del usuario.

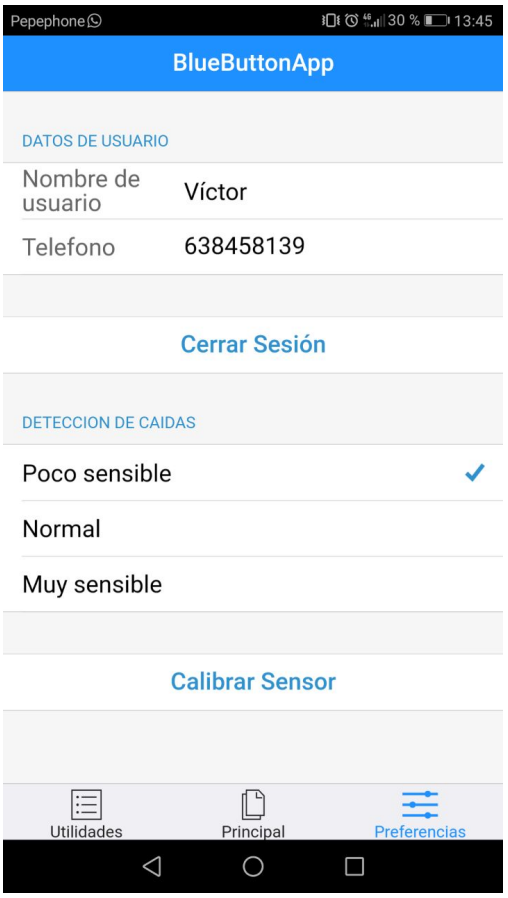


Figura 8.17: Pantalla Preferencias - Aplicación móvil

Para poder calibrar adecuadamente el nivel de detección de caída es necesario elegir el nivel pulsando en el nivel deseado y después calibrar el sensor pulsando el botón **Calibrar sensor** para que los valores del acelerómetro se reseteen adecuadamente.

A partir de aquí ya se puede utilizar correctamente la aplicación desarrollada y disfrutar de las funcionalidades que ofrece. Es recomendable que el primer uso y su configuración se realice con supervisión y/o ayuda de la persona de contacto, ya que será ella la que mejor pueda comprender sus necesidades para con la aplicación.

8.2.2. Manual de usuario aplicación web

La aplicación web será el lugar donde se realice el registro de usuarios en la aplicación. Al acceder a la misma se obtendrá una ventana en la que el usuario podrá acceder a su sesión, si ya ha sido registrado con anterioridad, o en el caso contrario pulsar el botón de registro, lo que le enviará a la pantalla de registro. Podrá acceder desde un navegador convencional o desde el mismo navegador del móvil:

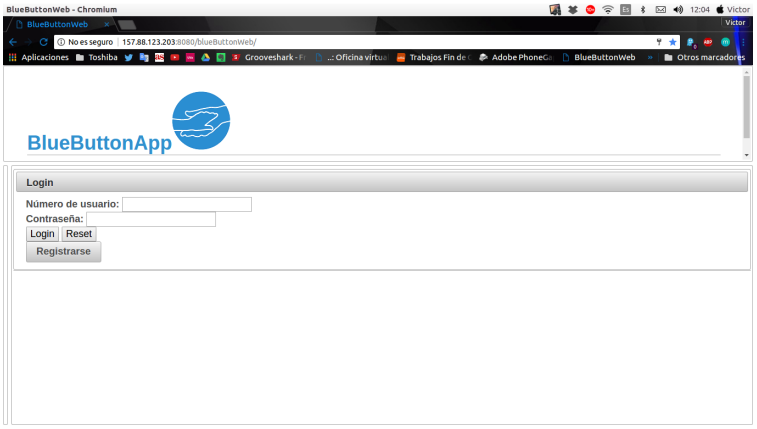


Figura 8.18: Pantalla Login - Aplicación web

Figura 8.19: Pantalla Login - Aplicación web - Dispositivo móvil

En el caso de que el usuario no esté registrado, podrá registrarse en la plataforma, accediendo a la ventana de registro pulsando el botón de **Registro de usuario**:

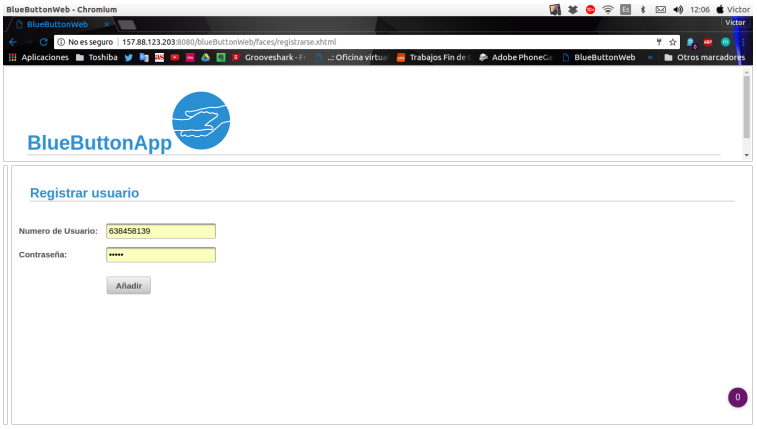


Figura 8.20: Pantalla Registro - Aplicación web

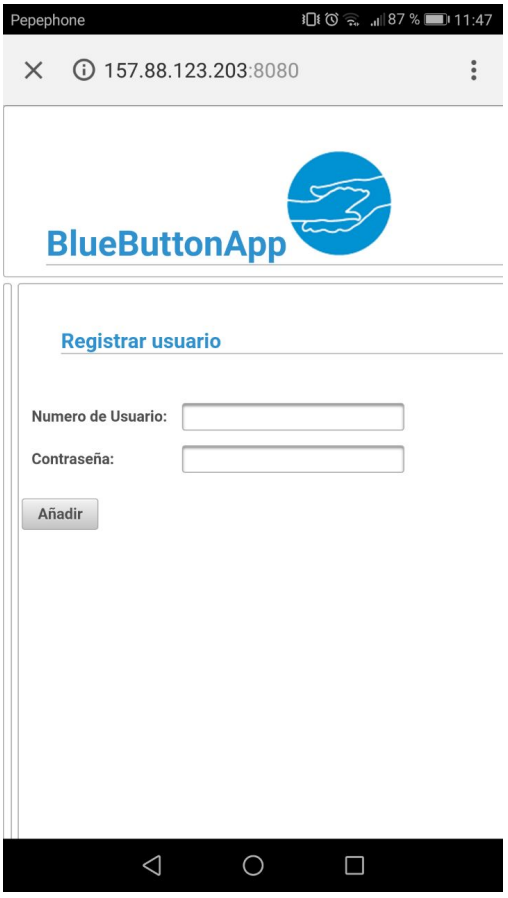


Figura 8.21: Pantalla Registro - Aplicación web - Dispositivo Móvil

El usuario deberá introducir sus datos para el registro, y pulsar en el botón de **Añadir** que devolverá un mensaje con el resultado del registro y si la misma ha sido correcta redirigirá a la página de login para que el usuario pueda iniciar sesión, tanto en la aplicación web o móvil.

Para facilitar la supervisión del usuario de la aplicación móvil se tiene acceso a la plataforma web, donde un usuario registrado podría acceder y visualizar rápidamente la última

posición registrada por el dispositivo móvil en los últimos 15 minutos. Cuando el usuario haya iniciado sesión en la aplicación web introduciendo sus credenciales de usuario, podrá acceder a la pantalla principal, donde se le mostrará un mapa con la posición del dispositivo móvil resaltada con un marcador rojo.

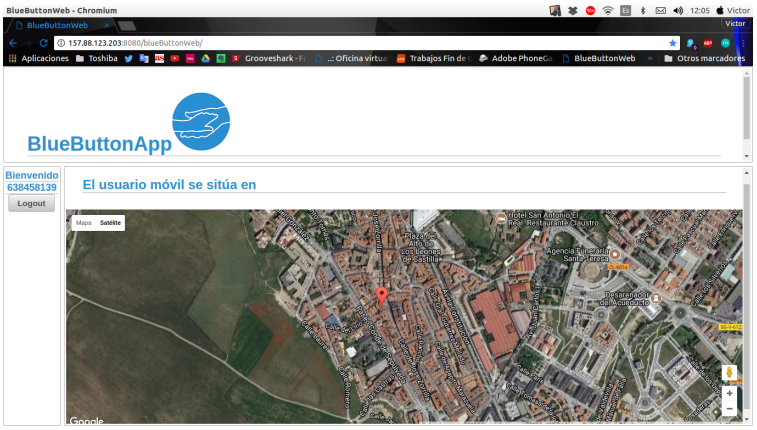


Figura 8.22: Pantalla Inicio - Aplicación web

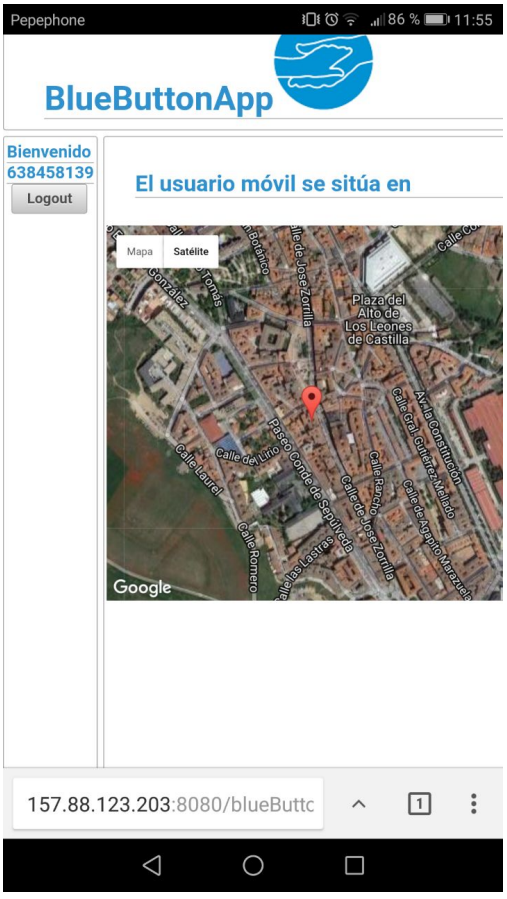


Figura 8.23: Pantalla Inicio - Aplicación web - Dispositivo Móvil

Capítulo 9

Conclusiones y líneas futuras

9.1. Conclusiones

El proyecto comenzó como una idea de mi tutor Aníbal, y sinceramente a simple vista me pareció una idea llamativa, ya que consistía en una aplicación que podría ser de ayuda a los demás y ser aplicado a bastantes ámbitos. A su vez generaba un interés en mi por el hecho de implementar aplicaciones móviles híbridas con Phonegap, ya que era algo que nunca había hecho y no nos enseñaban en el grado y me llamaba la atención, porque lo veía como una fusión de muchas cosas, aplicaciones en Android[1] y en otras plataformas como puede ser iOS o Windows Phone para las que nunca había desarrollado nada.

Ha sido una experiencia gratificante ya que ha supuesto mucho esfuerzo, y un gran aprendizaje en distintos ámbitos. He conseguido aprender varias tecnologías nuevas(Phonegap, ReactJS,..) y aprender mucho de algunas que ya conocía(JavaScript).

Es cierto que no es un producto perfecto, quizá porque no se disponga de los medios o del tiempo suficiente, pero desde mi punto de vista me encuentro bastante satisfecho con el resultado.

Por último añadir que toda la realización de este proyecto, tanto la planificación de trabajo, como el desarrollo o las pruebas me ha supuesto un gran reto que ha concluido en un aprendizaje tanto para asentar la multitud de conocimientos adquiridos durante el grado como a nivel personal obligándome a organizar mi día a día entre mi trabajo y la realización de este proyecto, que me ha supuesto un notable crecimiento personal.

9.2. Líneas futuras

Tras el finalizar el desarrollo de este proyecto se puede concluir que se han incluido todas las funcionalidades planteadas en un principio, inclusive las planteadas como adicionales posteriormente. El producto final es totalmente funcional, aunque según ha ido avanzando el proyecto han surgido posibles características que pudieran ser añadidas para una versión posterior, ya que para los plazos actuales no era posible.

Las ideas de posibles mejoras o funcionales añadidas podrían ser las siguientes, por supuesto después de haberlas analizado más detenidamente: -Actualmente la aplicación móvil

[1]impartido en la asignatura de Plataformas Software Móviles de 4º Curso

no permite al usuario, que se tratará de una persona dependiente, modificar las prescripciones que tenga creadas para evitar que las elimine sin darse cuenta y eso pueda ocasionar problemas en su tratamiento médico. Por lo que se podría plantear la implementación de un pequeño usuario móvil que permitiera supervisar las prescripciones del usuario dependiente asignado y modificarlas en el caso que fuera necesario. -El desarrollo inicial se orientó a Android por su versatilidad, pero también a otras plataformas, por eso se eligió Phonegap como tecnología principal. La interfaz de la aplicación se ha realizado pensando en la futura portabilidad hacia iOS como segunda fuente de usuarios ya que iOS es una plataforma menos flexible que Android. Al principio dio muchos problemas aunque el resultado final de la interfaz desde mi punto de vista es bastante agradable a la vista. -La actual interfaz de la aplicación web esta un poco descuidada ,posiblemente ocasionado la falta de tiempo causada en parte por los múltiples problemas que había con la máquina de Valladolid o quizá otros factores, una de las posibles mejoras sería mejorar la imagen de la aplicación web para que resulte más agradable al usuario.

Bibliografía

- Temario y documentación de la asignatura de Plataformas Software Empresariales(curso 2015-2016) - [01/07/2017] - Aníbal Bregón Bregón.
- Temario y documentación de la asignatura de Plataformas Software Móviles(curso 2015-2016) - [15/06/2017] - Aníbal Bregón Bregón.
- Temario y documentación de la asignatura de Administración de Base de Datos(curso 2015-2016) - [20/06/2017] - Miguel Ángel Martínez Prieto.
- Temario y documentación de la asignatura de Modelado de Software de Sistemas de Informáticos(curso 2015-2016) - [10/07/2017] - Miguel Ángel Martínez Prieto.
- Temario y documentación de la asignatura de Gestión de Proyectos basados en las Tecnologías de Información(curso 2014-2015) - [30/06/2017] - Francisco J. González Cabrera.

Webgrafía

- Stack Overflow Community - http://stackoverflow.com/ - [Última visita: 12/07/2017].
- PrimeFaces, Ultimate UI Framework for Java EE - http://www.primefaces.org/ - [Última visita: 14/07/2017].
- Calcula tu Salario.es - - [Última visita: 01/07/2017].
- Google Maps API Documentation - API Google Maps - https://developers.google.com/maps/docu 419 - [Última visita: 01/07/2017].
- Apache Cordova - https://cordova.apache.org/ - [Última visita: 01/06/2017].
- GitHub, Cordova-sqlite-storage - https://github.com/litehelpers/Cordova-sqlite-storage - [Última visita: 25/06/2017].
- GitHub, cordova-plugin-device-motion - https://github.com/apache/cordova-plugin-device-motion/blob/master/doc/es/index.md - [Última visita: 12/06/2017].
- GitHub, cordova-plugin-geolocation - https://github.com/apache/cordova-plugin-geolocation - [Última visita: 15/06/2017].
- GitHub, cordova-plugin-local-notifications - https://github.com/katzer/cordova-plugin-local-notifications - [Última visita: 17/06/2017].
- GitHub, CordovaCallNumberPlugin - https://github.com/Rohfosho/CordovaCallNumberPlugin - [Última visita: 16/06/2017].
- React JS - https://facebook.github.io/react/ - [Última visita: 20/04/2017].
- TouchStone JS - http://touchstonejs.io/ - [Última visita: 17/06/2017].

- TouchStoneJS Starter - https://github.com/touchstonejs/touchstonejs-starter.git - [Última visita: 15/06/2017].
- GlassFish - https://glassfish.java.net/ - [Última visita: 01/01/2017].
- Aprendiendo Latex - http://minisconlatex.blogspot.com.es/ - [Última visita: 15/07/2017].
- Introducción a Phonegap - http://alfonsomarin.com/desarrollo-movil/tutoriales/introduccion-a-phonegap-apache-cordova - [Última visita: 01/05/2017] - Alfonso Marín.
- Que es y como funciona ReactJS - https://platzi.com/blog/intro-react-js/ - [Última visita: 05/05/2017].
- Seis apps para personas dependientes http://www.gerosol.com/seis-apps-para-personas-dependientes-o-con-alguna-discapacidad/ - [Última visita: 01/04/2017].
- Crea tu primera aplicación ReactJS con create-react-app - http://www.enrique7mc.com/2017/02/crea tu-primera-aplicacion-en-react-con-create-react-app/ - [Última visita: 05/05/2017] - Enríque Munguía.
- Sample Phonegap app with ReactJS - http://devgirl.org/2015/09/22/sample-phonegap-app-with-reactjs/ - [Última visita: 15/05/2017] - Holly Schinsky.
- Phonegap API Documentation - http://docs.phonegap.com/en/edge/ - [Última visita: 15/05/2017].
- Phonegap Spain - http://www.phonegapspain.com/ - [Última visita: 15/06/2017]

Apéndice A

Creación de aplicaciones móviles híbridas

Para poder comenzar con la realización de una aplicación móvil que sea híbrida hay que instalar varios programas y realizar las configuraciones pertinentes. En mi caso, partiré con un ordenador Toshiba Satellite L750/755, con un procesador intel core i5 a 2,4GHz, con una memoria RAM 8GB DDR3. En él corre un sistema operativo Ubuntu 16.04 LTS de 64bits. Después de esto comenzamos con la configuración del sistema para poder realizar aplicaciones híbridas con un framework, que en mi caso será PhoneGap[1]. En la página oficial se encuentra una guía bastante completa de cómo realizar la instalación, la que yo he realizado es de la siguiente manera:

- En primer lugar se debe instalar nodejs:

  ```
  $ sudo apt-get install nodejs
  ```

- También se debe instalar npm para gestionar los paquetes de Node.js:

  ```
  $ sudo apt-get install npm
  ```

- Teniendo instalado npm podemos dirigirnos a la instalación de Phonegap con el siguiente comando:

  ```
  $ npm install -g phonegap@latest
  ```

  Es un proceso que tardará, ya que descargará todos los paquetes necesarios e irá configurándolos para el correcto funcionamiento de la aplicación en el sistema.

- Es recomendable instalar algunas librerías de las que depende Phonegap para su correcto funcionamiento:

  ```
  $ sudo apt-get install lib32z1 lib32ncurses5 lib32bz2-1.0 lib32stdc++6
  ```

[1] http://Phonegap.com/

- Para poder crear aplicaciones para el sistema operativo Android debemos tener instalado el SDK de Android, para ello la manera más simple es instalar la aplicación de escritorio de AndroidStudio[2] que te descarga automáticamente el SDK, las SDK-tools[3] y las SDK-build-tools[4] a la vez que realiza su instalación y además te permite crear emuladores para probar tus aplicaciones.

  - Primero debemos descargarnos el paquete y descomprimirlo en la ubicación */usr/local/*.

  - A continuación nos dirigimos a través de un terminal al directorio *android-studio/bin/* y ejecutamos *studio.sh*

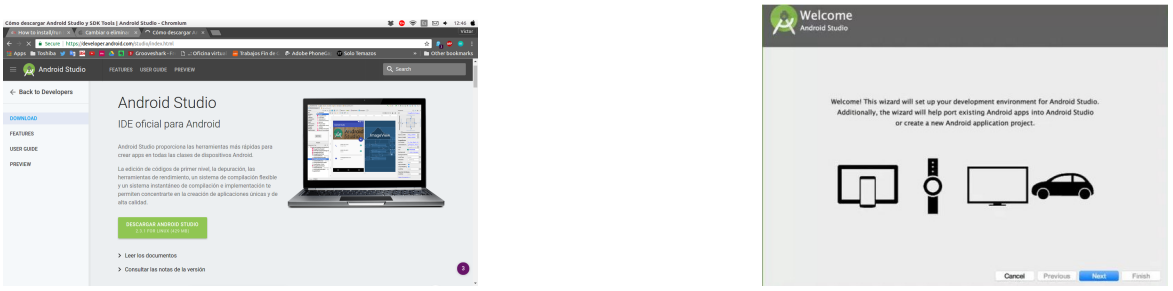


Figura A.1: Descarga e Instalación de AndroidStudio

- En el caso de querer desarrollar aplicaciones para iOS es necesitamos la aplicación de xCode, para poder ejecutar dicha aplicación es obligatorio tener un ordenador con sistema operativo macOS. Para descargarlo accedemos al Apple Store y lo descargamos, en el caso de que no lo tengamos ya instalado:

[2]http://developer.android.com/sdk/index.html
[3]http://tools.android.com/recent/androidsdktoolsrevision2530feb2017
[4]https://developer.android.com/studio/releases/build-tools.html

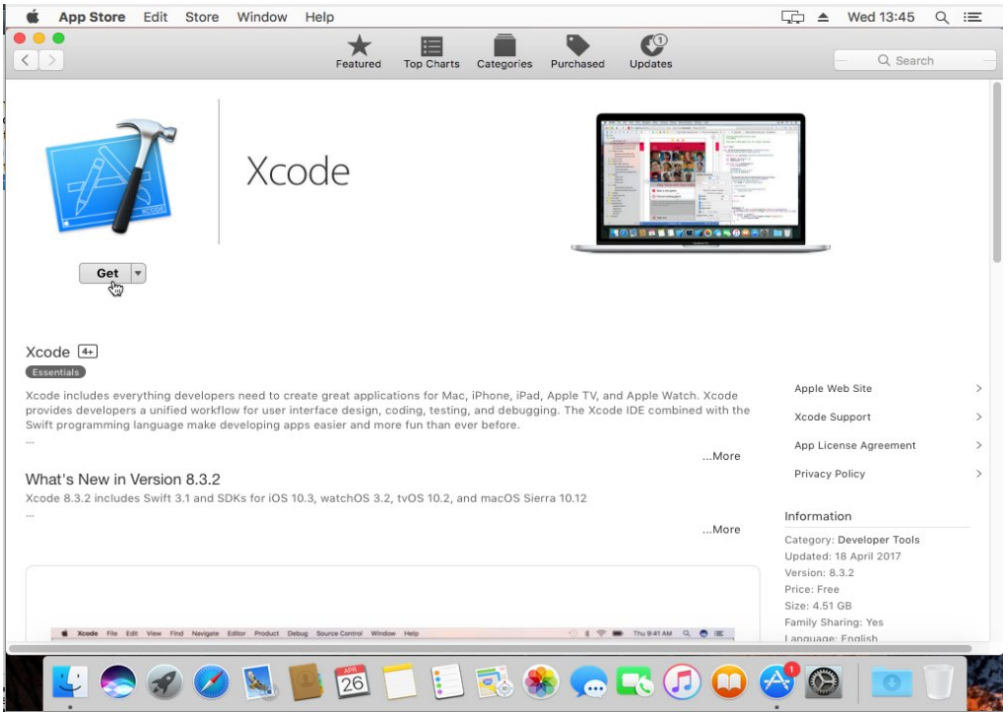


Figura A.2: xCode Apple Store.

- Para poder realizar pruebas de nuestras aplicaciones creadas en Android es muy recomendable usar un emulador, en mi opinión uno de los que mejor funciona gracias a su rapidez de reacción es Genymotion[5].

  - Para instalarlo es necesario hacerse una cuenta en su página que nos permitirá la descarga. En el caso de tratarse de uso personal únicamente es gratuito, el link de descarga es el siguiente: descarga[6].

[5]https://www.genymotion.com/
[6]https://www.genymotion.com/download/

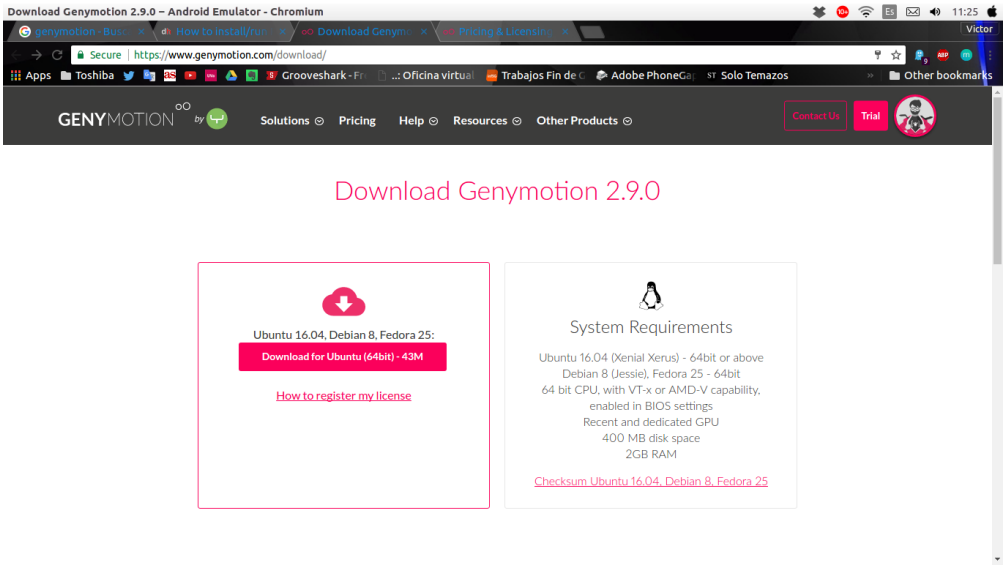


Figura A.3: Página de descarga de Genymotion.

- Cuando haya finalizado la descarga del archivo de Genymotion se le asignarán permisos de ejecución al archivo y después se le ejecutará para su instalación:

```
$ chmod u+x genymotion-2.8.1_x64.bin
./genymotion-2.8.1_x64.bin
```

- Para poder ejecutar Genymotion correctamente se debe tener instalado VirtualBox, de la siguiente manera:

```
$ sudo apt-get install dkms
$ sudo apt-get install virtualbox-5.3
```

- Por último se debe configurar un dispositivo Android que se quiera emular, de una manera bastante simple:

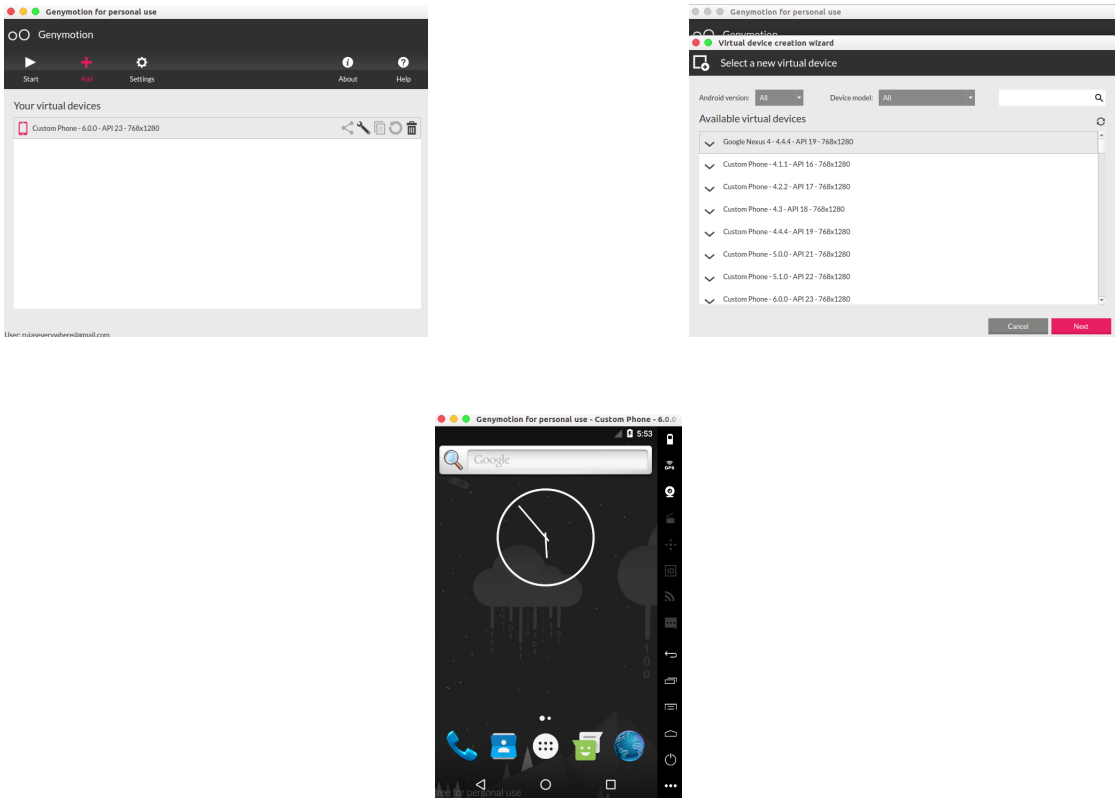


Figura A.4: Creación de un emulador Android con Genymotion.

- Ya esta todo listo para crear una aplicación con Phonegap. Los pasos básicos son los siguientes:

  - Creamos un nuevo proyecto:

    ```
    $ phonegap create HolaMundo com.rujass.holamundo HolaMundo
    ```

  - A continuación a declaramos la ruta de la variable *ANDROID_HOME* y de las *tools*, *build-tools* y las *platform-tools*, que en mi caso sería lo siguiente:

    ```
    $ export ANDROID_HOME=~/Android/Sdk
    $ PATH=$PATH:$ANDROID_HOME/tools
    $ PATH=$PATH:$ANDROID_HOME/platform-tools
    $ PATH=$PATH:$ANDROID_HOME/build-tools
    ```

  - Ahora podemos añadir la plataforma en la que queramos que se ejecute la aplicación, por ejemplo Android:

    ```
    $ phonegap platform add android
    ```

  - Construimos el proyecto:

    ```
    $ phonegap build
    ```

  - Por último ya podemos ejecutar la aplicación en nuestro emulador:

    ```
    $ phonegap emulate android
    ```