Grado en Ingeniería Informática

Escuela Politécnica Superior de Jaén

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior (Jaén)

Trabajo Fin de Grado

GESTIÓN Y ACCESO A LOS SERVICIOS DE UNA INSTITUCIÓN O EMPRESA MEDIANTE APLICACIÓN MÓVIL

Alumno: Pablo Mantas Torres

Tutor: Prof. D. Pedro J. Sánchez Sánchez
Dpto: Informática

Noviembre, 2015

Universidad de Jaén
Escuela Politécnica Superior de Jaén
Departamento de Informática

Don Pedro J. Sánchez Sánchez , tutor del Proyecto Fin de Carrera titulado: Gestión y Acceso a los Servicios de una Institución o Empresa mediante Aplicación Móvil, que presenta Pablo Mantas Torres, autoriza su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, Noviembre de 2015

El alumno: Los tutores:

Pablo Mantas Torres Pedro J. Sánchez Sánchez

Agradecimientos

En primer lugar todo dar las gracias a todas aquellas personas que me han apoyado y animado a terminar este proyecto porque siempre han creído en mi y en que podía conseguirlo.

Mis padres que siempre han confiado en mí y en que podía con todo lo que me propusiera.

A mis amigos que empatizaban conmigo sabiendo lo duro que puede ser trabajar y gestionarse el tiempo para terminar un proyecto así. Y a mis más que amigos por pincharme y “obligarme” a no ir dejandolo día tras día.

Y por último a mi tutor del proyecto Pedro José Sánchez Sánchez que siempre ha estado atento a mis dudas y a mis correos y me ha ayudado tanto para que el trabajo estuviera a tiempo.

¡Muchas gracias por todo!

Índice

1 INTRODUCCIÓN

1.1 Introducción al trabajo

Este proyecto nace a partir de un caso específico dentro del Trabajo Fin de Grado genérico denominado “Gestión y Acceso a los Servicios de una institución o empresa mediante Aplicación móvil”. El caso específico del que hablamos es el control de posición de empleados y uso de la flota de vehículos de una empresa.

La empresa elegida para este proyecto es ENDESA, una empresa eléctrica en la que los técnicos tienen que disponer de vehículos preparados para desplazarse por distintas instalaciones eléctricas.

En el caso concreto que vamos a desarrollar, tomaremos como “Servicios de una empresa” el uso de cualquiera de los vehículos que la empresa pone a disposición de los empleados para que estos realicen desplazamientos con motivo laboral. La gestión y acceso a los servicios será registrada por una aplicación movil que trabajará en conjunto con un servicio web y una base de datos que se alojarían en un servidor de la propia empresa.

Para el desarrollo de todo el proyecto vamos a necesitar la combinación de múltiples herramientas y tecnologías que se describirán a lo largo de esta memoria.

1.2 Motivación

A día de hoy la gran parte de la población dispone de algún tipo de dispositivo movil con conectividad a internet y a señales GPS. No solo ayudan en el día a día de personas particulares sino que también son cada vez más las empresas que se unen a esta tendencia de dotar a sus trabajadores con algún tipo de dispositivo movil con dicha conectividad para aportar soporte a las funciones de sus empleados.

En este proyecto me centraré en el desarrollo de una Aplicación Android para la geolocalización de dispositivos móviles con el fin de conocer la situación de un empleado cuando este use un vehículo de la flota de la empresa. En concreto de una empresa eléctrica en la que los técnicos tienen que realizar habitualmente desplazamientos a instalaciones en las que se deben de reunir con distintas personas del equipo.

Así se pretende gestionar de una manera más eficiente la disposición de los vehículos de la empresa por parte de los empleados usuarios. En muchas ocasiones empleados destinados en distintas delegaciones necesitan de un automóvil para desplazarse a la Sede Central o a distintas instalaciones eléctricas, por ello saber en qué situación se encuentran los vehículos, esto les ayuda a gestionar su tiempo de trabajo ya que pueden conocer la ubicación exacta de los vehículos en los que se desplazan los empleados con los que van a reunirse o realizar cualquier tarea en el ámbito laboral. A su vez conocerán la disponibilidad de los vehículos para poder hacer uso de los mismos durante sus jornadas.

1.3 Objetivos

La idea principal de este proyecto es la de gestionar de una manera más eficiente el tiempo a los técnicos de una empresa eléctrica, ya que a través de ésta aplicación podrán conocer la geolocalización del empleado con el que va realizarse la reunión. Para ello los empleados podrán acceder a su lista de contactos en la aplicación y comprobar a que distancia se encuentra cada miembro del equipo del lugar acordado y así poder calcular el tiempo que deben invertir.

Además, por otro lado podrán conocer la disponibilidad de los vehículos de la flota de la compañía para hacer uso de los mismos.

Otra parte importante del objetivo es la de gestionar el uso de los vehículos de la empresa que se encuentran a disposición de los técnicos ya que en la mayoría de los casos estos deben de turnarse para hacer uso de los mismos y con esta aplicación los técnicos pueden ver los coches disponibles antes de acordar un encuentro con su equipo.

Y por último cabe mencionar que otra parte importante en una empresa es la del control de uso que hagan sus empleados de los vehículos de la empresa por lo que quedarán registradas toddas las rutas que haga cada técnico para llevar un posible control de kilómetros o ante posibles auditorías internas e incluso saber si hay demasiados vehículos o por el contrario la flota no es suficiente para las necesidades de la empresa.

Para realizar este proyecto es necesario estudiar previamente las tecnologías de las cuales nos valdremos para su desarrollo. Dichas tecnologías se describirán más adelante.

1.4 Resumen

En los siguientes capítulos podremos encontrar una descripción más detallada de las tecnologías y principales elementos del proyecto.

En el punto dos, descubriremos la definición de Servicio Web, las ventajas e inconvenientes del uso de un servicio web y las plataformas donde podremos utilizarlo, descripción detallada de Axis2 como motor del servicio, conoceremos que es MySQL como gestión de base de datos, descripción de Tomcat 7 como servidor de nuestro servico web y finalmente explicaremos los que es un servicio REST y que debe cumplir nuestro servicio para ello.

En el punto tres, encontraremos información de interés sobre Android. En primer lugar haremos una breve descripción, aportaremos sus principales carácterísticas, su estructura, la relación que existe y de la que haremos uso entre Android y los servicios REST y finalmente algunas notaciones sobre el posicionamiento GPS del que nos valdremos en este proyecto.

En el punto 4 se desarrolla el análisis del sistema, una descripción detallada y síntesis del problema planteado desglosándolo en diseño, requerimientos, casos de uso y posibles escenarios. Hablaremos también de la interfaz e incluiremos un storyboard, describiremos en detalle la implementación tanto del servicio web como de la aplicación y su funcionamiento y por último se incluye una pequeña campaña de test sobre el funcionamiento del sistema y sus resultados .

Finalmente se incluirán como anexos a la memoria un manual de instalación del Servicio Web y de las herramientas necesarias (Anexo A), un manual de instalación de la aplicación Android (Anexo B) y una guía de usuario (Anexo C).

2 SERVICIO WEB

2.1 ¿Qué es un Servicio Web?

Un Servicio Web [04] es una tecnología constituida por un conjunto de protocolos y estándares con los que se permite el intercambio de información entre aplicaciones. Estas aplicaciones podrían estar programadas en distinto lenguaje o plataforma a través de internet o red de dispositivos. Esto se consigue mediante estándares abiertos regulados por las organizaciones OASIS[05] (Organization for the Advancement of Structurad Information Standards) y W3C[06] (World Wide Web Consortium) para garantizar la interoperabilidad sin importar donde se desarrolle.

2.1.1 Estándares que se emplean

Algunos de los principales estándares[07] que encargan de esta tarea son:

Web Services Protocol Stack: Conjunto de los servicios y protocolos pertenecientes a los servicios web.

XML[19] (Extensible Markup Language): Este estandar define el formato que deben seguir los paquetes de datos que se intercambian.

SOAP (Simple Object Access Protocol): Protocolos en los que se basan los intercambios de datos.

WS-Security [03] (Web Service Security): Protocolo de seguridad estandard por OASIS el cual garantiza la autenticación y confidencialidad de los actores implicados en el intercambio.

2.1.2 Ventajas e inconvenientes

Ventajas:

- La principal ventaja es que permiten la interoperabilidad de aplicaciones independientemente de su lenguaje o plataforma en las que se ejecuten.
- Motivan el desarrollo de protocolos y estándares basados en texto que facilitan la accesibilidad a los mismos y su entendimiento.
- Permiten que aplicaciones, software y servicios de distintos tipos y con distintas funciones se aunen para potenciarse mutuamente y crear nuevas aplicaciones.

Inconvenientes:

- En cuanto a los inconvenientes cabe destacar que el rendimiento sea menor en comparación con otros modelos de computación distribuida como CORBA[08] (Common Object Request) o RMI[09] (Remote Method Invocation), provocado en contraprestación a aceptar un formato basado en texto.
- Algunas medidas de seguridad del *firewall* basadas en bloquear la comunicación entre programas podrían ser evitadas.

2.1.3 Plataformas

Todo servicio web necesita una plataforma donde ejecutarse para lo que se emplean Servidores de aplicaciones. Algunos de ellos son:

- **BEA WebLogic.**
- **JBoss.**
- **Oracle WebLogic Server.**
- **Axis2.**
- **Tomcat (Apache).**

2.1.4 Justificación

Nuestro servicio web hará de interfaz entre la base de datos y nuestra aplicación por lo que buscaremos la mayor compatibilidad y simplicidad de entegración para este servicio en los sistemas de cualquier organización.

La principal razón para elegir un Servicio Web es la compatibilidad con cualquier ordenador por lo que no habría que invertir en equipos específicos y otra de las razones es la facilidad para lograr realizar la comunicación entre equipos dentro de la red de cualquier organización ya que estas normalmente protegen sus redes filtrando el tráfico de datos mediante *firewalls* cerrando la mayoría de los puertos TCP a excepción del 80 que es el que utilizaremos pues es el empleado por los navegadores para HTTP (Hypertext Transfer Protocol). De esta manera evitaremos bloqueos en nuestra comunicación.

Otra razón por la que escogemos utilizar un Servicio Web es la independencia que este tiene con respecto a cualquier software o aplicación que haga uso de él por lo que el desarrollo del mismo se realiza completamente a parte.

2.2 Axis2

La plataforma elegida para alojar el servicio web que desarrollaremos en este proyecto ha sido Axis2.

Axis2 es un motor para servicios web bastante potente y diseñado completamente a partir de la reimplementación de la extendida pila SOAP “Apache Axis”.

2.2.1 Características Axis2

- Puede funcionar como **servidor autónomo**.
- Cuenta con **licencia de código abierto** Apache License 2.0 y es **multiplataforma**.
- Da soporte a estándares como:
    - WS - ReliableMessaging
    - WS - Coordination

  - WS - AtomicTransaction
  - WS - SecurityPolicy
  - WS - Security
  - WS - Trust
  - WS SecureConversation
  - SAML 1.1
  - SAML 2.0
  - WS – Addressing
- **Mejora de la velocidad** respecto a versiones anteriores de Axis gracias al análisis sintáctico basado en StAX[11] (Streaming API for XML).
- Diseño basado en **minimizar el uso de memoria**.
- Uso de un **modelo de objetos propio, AXIOM**[12], optimizado para mejorar su rapidez y su uso a los desarrolladores.
- Permite **añadir nuevos servicios y *handlers* al sistema sin necesidad de reiniciar** el servidor tan solo con copiar los ficheros necesarios al directorio que aloja los servicios.
- Permite la **invocación asíncrona de servicios** web por medio de clientes.
- Las **APIs** públicadas suelen tener un periodo de funcionamiento bastante elevado consiguiendo así una estabilidad bastante decente.
- **Integra y utiliza los senders y listeners para SOAP** mediante distintos protocolos como SMTP, FTP, etc, siendo este el núcleo para el motor de los mecanismos de transporte.
- **Soporta WSDL** (Web Services Description Languaje) por lo que es más sencillo contruir accesos a servicios remotos.
- Los módulos de Axis2 **soportan nuevas especificaciones WS-#** de forma sencilla aunque no permite su despliegue instantáneo para evitar alterar el funcionamiento general del sistema

2.2.2 Justificación

Es una evolución de Axis1 con una nueva arquitectura más flexible, configurable y eficiente e incorpora soporte para SOAP y REST.

2.3 MySQL

MySQL fue desarrollado por MySQL AB[13], una empresa subco de Sun Microsistems en 2008, la cual lo era de Oracle Corporation desde 2009 y fue desarrollado como un software libre con licencionamiento dual.

Los derechos de autor del código pertenecen a una empresa privada que tiene el copyright de la mayor parte del mismo.

En la actualidad MySQL es utilizado por muchos de los sitios más famosos de internet tales como Facebook, Google, Twitter, Youtube, etc.

2.3.1 Características MySQL

- MySQL es un sistema de gestión de bases de datos **relacional (datos en tablas separadas), multihilo y multiusuario** muy extendido globalmente y con una **amplia comunidad**.
- Tiene **licencia GNU GPL** aunque para las empresas que deseen incorporarlo a productos privados pueden comprar una licencia específica para su uso.
- Desarrollado en su mayor parte en **ANSI C**.
- Hay una **amplia variedad de APIs específicas** que permiten a las aplicaciones accedera las bases de datos dependiendo del lenguaje en que estén programadas C, C++, C#, Pascal, Delphi, Eiffel, Smalltalk, Lisp, Perl, PHP, Phyton, Ruby, ABAP (SAP), java (mediante implementación del diver de java).
- Funciona en **multitud de plataformas** (FreeBSD, GNU/Linux, Windows, Mac OS X…).
- Transacciones y **claves foráneas**.
- **Indexación** y búsqueda de campos de texto.
- Permite **agrupar transacciones** de distintas conexiones para mejorar la eficiencia.

2.4 Tomcat

Apache Tomcat o Jakarta Tomcat es un contenedor de pequeñas aplicaciones que se ejecutan en un navegador web denominados *servlets* y con su propio contenedor llamado *Catalina*[15]. También soporta JSPs ya que incluye un compilador (Jasper) que los convierte en servlets.

Ha sido desarrollado y está mantenido por los miembros de Apache Software Foundation[14] y la comunidad independiente. Posee licencia Apache License 2.0.

Puede funcionar como servidor web independiente y puede funcionar en cualquier sistema operativo que disponga de máquina virtual Java.

2.4.1 Árbol de directorios

La estructura de directorios de Tomcat es la siguiente:

- *bin:* inicio, parada, scripts y ejecutables
- *common:* clases que pueden usar las aplicaciones web y *Catalina*.
- *conf:* configuración de Tomcat, ficheros XML y DTD (Definición de Tipo de Documento).
- *logs:* logs de las aplicaciones y Catalina.
- *server:* clases que usa Catalina.
- *shared:* clases que pueden usar todas las aplicaciones web que alojamos.
- *webapps:* aplicaciones web alojadas.
- *work:* almacenamiento temporal de ficheros.

2.5 RESTful

REST[02] (REpresentational State Transfer) es un servicio sin estado (stateless), lo cual provoca que pierda todos sus datos de una llamada a otra por lo que será el cliente el encargado de recordar el estado y pasarlo en cada una de las llamadas al servicio. Esto conlleva que tengamos que reservar memoria para almacenar todos estos estados y por tanto supone una desvantaja pues según aumente el número de clientes también aumentará la memoria necesaria.

Para compensar la falta de estado se utilizan las “sesiones” que almacenan los datos de cada usuario en memoria RAM del propio servidor lo que conlleva que podamos quedarnos sin memoria si almacenamos demasiados datos en cada sesión o demasiados usuarios. Para conseguir una buena escalabilidad en aplicaciones web se intenta no utilizar la sesión.

2.6 SOAP

SOAP (Simple Object Access Protocol) es el protocolo muy complejo ideado par solucionar diferentes necesidades propias de las comunicaciones, seguridad, trensferencia segura de mensajes, etc.

La evolución de los servicios web ha hehco que SOAP se haga enorme intentando abarcar la mayoría de necesidades provocando que la gran parte de sus capacidades no se usen durante la mayor parte del tiempo.

2.6.1 SOAP vs REST

La principal diferencia entre los sevicios SOAP y REST es que SOAP se orienta a invocar métodos sobre un servicio remoto RPC[16] (Remote Procedure Call), miestras que REST lo hace sobre los recursos.

En una API REST tendremos recursos accesibles por URIs, identificadores, en los que podremos realizar diferentes acciones o CRUD (Create, Read, Update and Delete) entre los que se incluyen los POST, GET, PUT y DELETE mediante HTTP teniendo así acceso a la lista de objetos correspondientes a dicho identificador y obtener o crear objetos mediante POST o PUT con HTTP.

En SOAP por el contrario tendríamos que definir un proceso hacia el que realizar la llamada que nos devolviese un objeto en concreto pasándole su correspondiente identificador.

3 ANDROID

Para ponernos en antecedentes con la base de nuestro proyecto, el cual es la aplicación movil, vamos a contar algunos datos de interés sobre el origen y evolución del sistema operativo sobre el que se ejecuta la aplicación.

3.1 Android

En 2005 Google compró una pequeña y joven empresa llamada Android Inc. dedicada a producir aplicaciones para dispositivos móviles. Fue en ese año cuando se empezó a desarrollar una máquina virtual Java para dispositivos móviles (Virtual Machine Dalvik).

Posteriormente, en el año 2007, se creó el consorcio Handser Alliance[23] con el fin de establecer los estándares abiertos para dispositivos móviles y promover la difusión de Android[24] como plataforma. Para ello se comprometieron a publicar gran parte de su propiedad intelectual bajo la Licencia Apache v2.0 como código abierto.

Android es un Sistema Operativo que vio la luz por primera vez junto con el primer smartphone[01] (T-Mobile G1) desarrollado por Google y fabricado por HTC en 2008. Con la aparición de los smartphones y las distintas plataformas para desarrollar aplicaciones la sociedad ha cambiado en pocos años de forma significativa.

A día de hoy las grandes plataformas que compiten por el mercado de los dispositivos móviles son iOS[17] de Apple, Windows Phone[18] de Microsoft y Android de Google aunque existen muchas más que en la actualidad buscan hacerse un hueco en el mercado.

Una de las causas del éxito de Android ha sido la combinación, en un mismo sistema, de propiedades optimizadas para dispositivos móviles con otras ya estandarizadas mejorando así la compatibilidad entre aplicaciones y dispositivos.

3.1.1 Características

Las características pricipares que acabamos de mencionar son las siguientes:

- **Es adaptable al hardware**: No ha sido diseñado para un dispositivo en concreto.
- **Desarrollo libre**: Basado en Linux y de código abierto.
- **Portabilidad**: Puede ser ejecutado en cualquier CPU gracias a las máquinas virtuales Java.
- **Iterfaz basada en XML**: Se puede ejecutar en cualquier tipo de pantalla sin importar su resolución o tamaño.
- **Siempre conectado:** Filosofía basada en la conectividad permanente a internet para multiplicar sus posiblilidades.
- **Amplia gama de servicios:** Gracias a las características que incorpora tales como el soporte para localización GPS, Bases de datos SQL (SQLite[20]), giroscopio, etc.
- **Gran nivel de seguridad**: Los programas tienen su propio campo donde actuar limitado por una serie de permisos y se encuentran aislados unos de otros por el concepto de ejecución dentro de una caja heredado de Linux.
- **Optimización para dispositivos con poca memoria y potencia**: Gracias al uso de la Máquina Virtual Dalvik (Máquina Virtual Java implementada por Google) diseñada principalmente para dispositivos móviles.
- **Gráficos**: Posibilidad de utilizar tanto gráficos vectoriales (Flash[21]) como gráficos 3D basados en OpenGL[22].
- **Audio y video**: Incorpora codecs estándar tales como H.264(AVC), MP3, AAC, etc.

3.1.2 Arquitectura

La arquitectura de Android se basa en un sistema de capas (pila), que permite la comunicación vertical entre ellas para favorecer la interacción del usuario con el dispositivo.

Android es una plataforma que contiene una pila de software donde se incluye un sistema operativo, middleware[35] (Librerías y Runtime) y aplicaciones para el usuario.

En las siguientes líneas se dará una visión global por capas de cuál es la arquitectura empleada. Cada una de estas capas utiliza servicios ofrecidos por las anteriores, y ofrece a su vez los suyos propios a las capas de niveles superiores, tal como muestra la imagen propiedad de Google:

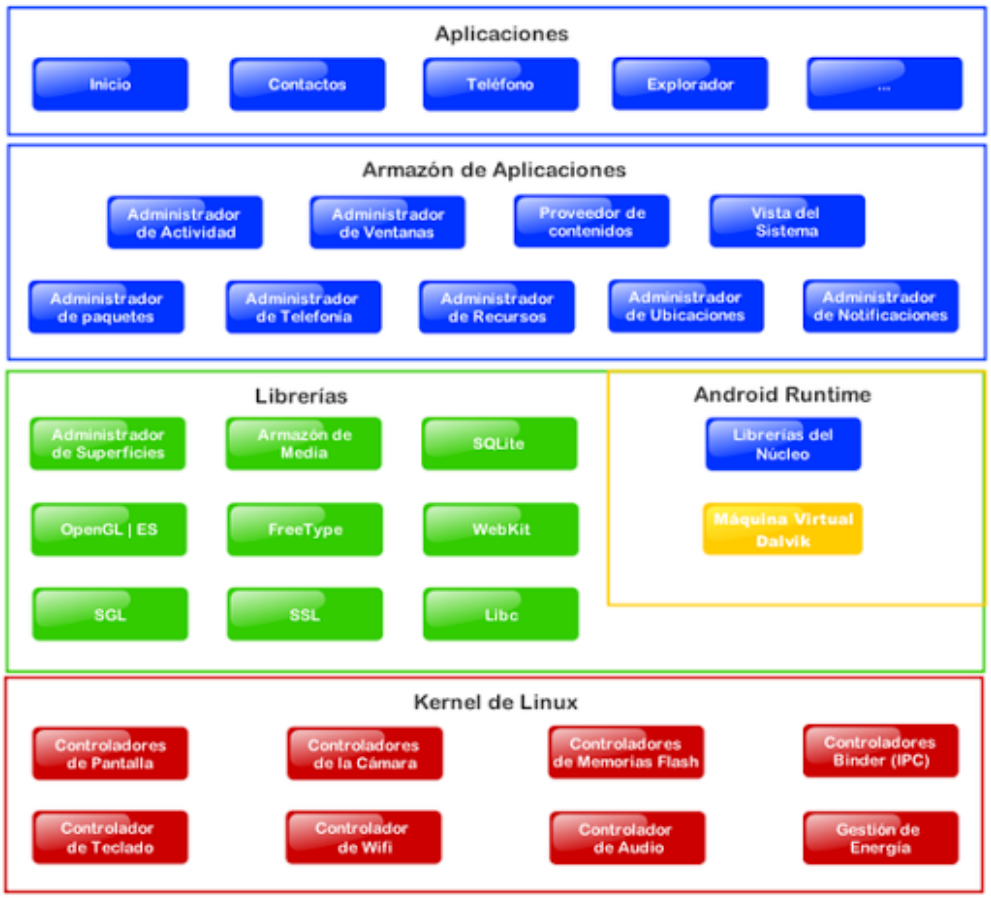


Figura 3.1: Arquitectura Android (© Google)

- **Capa Aplicaciones:** Contiene las aplicaciones básicas que incorpora por defecto Android más las que instale el usuario.
- **Capa Entorno:** Conjunto de herramientas de desarrollo de cualquiera de las aplicaciones. Las aplicaciones usan el mismo conjunto de APIs y "framework" [10]. Algunas de las más importantes son:

- *Activity Manager:* Para gestionar el ciclo de vida de una aplicación.
- *Window Manager:* Utiliza la Librería Surface Manager y gestiona las ventanas de las palicaciones.
- *Telephone Manager:* Todo lo relacionado conla funcionalidad propia de la telefonía, llamadas, sms, etc.
- *Content Provider:* Para que las aplicaciones puedan compartir sus datos con el resto de aplicaciones.
- *View System:* Contiene los elementos necesarios para construir la interfaz de usuario (botones, teclado, listas…).
- *Location Manager:* Para utilizar los servicios de localizsción y posicionamiento GPS.
- *Notification Manager:* Para la comunicación estre el usuario y las aplicaciones mediante eventos. Si el evento lleva asociado alguna acción se denominan “Intent” y se activan con un click.
- *XMPP Service:* Para utilizar el intercambio de mensajes basado en XML.

- **Librerías:** Creadas en C/C++ y gran parte de la funcionalidad característica de Android. Algunas de las más importantes son:
  - *Libc:* Cabeceras y funciones según el estándar de C.
  - *Surface Manager:* Gestiona las ventanas de las aplicaciones activas y une los diferentes componentes de navegación de la pantalla.
  - *OpenGL/SL y SGL:* Todo lo perteneciente a gráficos y a potenciar el hardware encargado de ellos.
  - *Media Libraries:* Codecs necesarios par contenido multimedia.
  - *FreeType:* Para trabajar con los distintos tipos de fuentes.
  - *SSL:* Para establecer conexiones seguras mediante protocolo.
  - *SQLite:* Para el uso de base de datos relacionales en las aplicaciones.
  - *WebKit:* Núcleo del navegador nativo de Android y el motor para las aplicaciones basadas en navegador.

- **Runtime:** Capa correspondiente al entorno de ejecución donde se encuentran las clases Java y la máquina virutal Dalvik la cual ejecuta los ficheros .dex diseñados para optimizar el ahorro de memoria y basada en registros. Cada aplicación corre su propio proceso linux con su propia intancia de máquina vitual.
- **Capa Kernel:** Capa que contiene los drivers necesarios para que los componentes hardware puedan ser utilizados mediante llamadas, así como la gestión de memoria, miltiproceso, pila de protocolos y seguridad.

3.2 REST en la práctica

Ya se ha hablado anteriormente de los servicios REST pero no se ha comentado cómo interactua nuestra aplicación con este tipo de servicio por lo que a continuación nos centraremos en el desarrollo y explicacion de una forma genérica.

Con REST no tenemos que añadir ninguna capa adicional a la pila de protocolos de nuestra aplicación ya que empleamos directamente el de HTTP. Para ello tendremos que seguir los principios dictados por la World Wide Web cambiando que en lugar de solicitar webs, solicitaremos servicios. Dichos principios son:

- Uso de las operaciones GET, PUT,POST y DELETE para el transporte de datos mediante HTTP.
- Las llamadas a los servicios se realizan mediante el uso de URIs, siendo este uno de los pilares de WWW.
- Los datos se codifican con los tipos MIME aunque el tipo de codificación predominante es el XML.

Algunas de las ventajas del uso de REST es la disminución de sobrecarga, una mejora de tiempos de respuesta por ambas partes (Cliente-Servidor) y estabilidar frente a cambios.

Para poder utilizar este tipo de servicio con nuestra aplicación tendremos que hacer varias cosas:

- Definir la URL específica del servicio y añadir la ruta de la función que deseemos utilizar y los parámetros necesarios de entrada que necesitemos.
- Establecer la conexión con la URL (con POST normalmente).
- Esperar la respuesta que será en formato XML y la procesaremos mediante un objeto de la clase SAXParser.
- Por último tendremos que construir nuestro propio manejador web para tratar la respuesta de manera correcta.

Gracias al uso de estos servicios web externos podremos descentralizar el procesamiento de la aplicación haciendo que Android sólo se encargue de la parte de cliente para el acceso y gestión.

3.3 Posicionamiento GPS

La principal carácterística de nuestra aplicación se basa en la localicación geográfica por lo que necesitamos algún método para solicitar nuestra posición al hardware de nuestro dispositivo que nos permita obtenerla y lo logramos gracias a una API, ***Google Maps Android API[26]*** , con la cual podemos realizar dichas peticiones para conocer nuestra posición geográfica siempre y cuando el dispositivo disponga del hardware necesario para ello. La localización se basa principalmente en la conexión GPS, pero también podemos obterner una localización aproximada mediante la triangulación de redes wifi y/o de telefonía movil.

Estos distintos tipos de geolocalización se complementan de tal manera que el posicionamiento GPS nos indique la posición con mayor precisión (hasta 15 metros de margen de error) en exteriores y que por el contrario la posición dentro de edificios la obtengamos gracias a la triangulación de torres de telefonía movil y de puertos de acceso wifi ya que generalmente la conexión con satélites GPS suele ser nula en espacios cerrados.

4 ANÁLISIS DEL PROYECTO

4.1 Descripción

Tras la exposición de los objetivos del proyecto como la motivación del mismo se procederá a definir el proceso de desarrollo de Ingeniería del Software[25] que infunda todo este proyecto.

Para empezar hay que dejar claro que es la Ingeniería de Software :

El estudio y aplicación de un modelo sistemático, disciplinado y cuantificable en proceso de desarrollo y mantenimiento del software.

Las fases necesarias para realizar una buena Ingeniería del Software son:

- Especificación de requerimientos: Identificar el tema principal por el cual iniciamos este estudio y desarrollo, incluyendo tanto recursos humanos como materiales.
- Análisis del sistema: Entender y documentar que debe hacer exactamente un sistema.
- Diseño del sistema: Proceso en el que se aplican distintas técnicas con el propósito de crear un dispositivo, proceso o sistema lo suficientemente detallado como para su reutilización física.
- Implementación del sistema: Fase en la que el sistema previamente definido pasa a código.
- Pruebas y Validación: Proceso en el que tras la realización del sistema se realizarán distintas pruebas para validar y verificar su correcto funcionamiento.

Ahora se procederá a aplicar cada una de estas fases a nuestro proyecto.

4.2 Especificación de requerimientos

Este es el primer paso que debemos realizar en la Ingeniería de Software ya que en él se establecen las necesidades y servicios que el cliente demanda en un sistema junto con las restricciones y preferencias de funcionamiento bajo las que será desarrollado.

En este caso el fin de este proyecto es puramente académico por lo que el objetivo del mismo queda definido con antelación.

"Gestión y Acceso a los Servicios de una Empresa de Telefonía mediante Aplicación Móvil."

Al tratarse de un proyecto genérico hay que concretar un poco más en la aplicación de nuestra solución que en este caso se trata de gestionar el uso de la flota de vehículos de una empresa y poder acceder a la localización de dichos vehículos y por consiguiente al empleado que lo está usando mediante una aplicación movil.

Ya que tenemos claro el propósito del proyecto nos queda definir las propiedades o restricciones en más detalle, que nuestro sistema debe cumplir. Existen dos tipos de requerimientos:

- Requerimientos funcionales: describen la funcionalidad del sistema con el comportamiento y tareas que este mismo puede realizar.
- Requerimientos no funcionales: describen el grado de cumplimiento del sistema relacionado con los requisitos funcionales.

Una vez hechas estas pequeñas aclaraciones procedo a definir todos los requerimientos necesarios para nuestro proyecto.

4.2.1 Requerimientos funcionales

Después del estudio de las distintas posibilidades de uso por parte de los usuarios se definen las siguientes funcionalidades necesarias para que el uso del sistema sea correcto y su consiguiente detalle de funcionamiento:

- **Iniciar sesión:** función necesaria para acceder al contenido de la aplicación y todo lo que esta nos ofrece.
- **Cerrar sesión:** tras el uso de la aplicación y cuando el usuario haya decidido que no va a realizar ninguna tarea más sobre ella deberemos utilizar esta función para evitar que la sesión siga activa.
- **Recuperar contraseña:** en el caso de que hayamos olvidado la contraseña por el motivo que sea deberemos hacer uso de esta función que nos enviará nuestra contraseña al correo que tengamos registrado en nuestra cuenta.
- **Registrarse:** función que deberemos usar en el caso de que sea la primera vez que utilizamos la aplicación para crearnos un perfil con nuestros datos personales.
- **Modificar datos**: una vez creado nuestro perfil tendremos la oportunidad de modificar los datos introducidos en el caso de que detectemos algún error o que deseemos cambiar alguno de ellos.
- **Eliminar cuenta:** en el caso de que la estancia en la empresa haya concluido deberemos eliminar nuestra cuenta del sistema.
- **Ver mis rutas:** posibilidad de consultar el historial de rutas realizado.
- **Añadir usuario a mi lista:** posibilidad de añadir otros usuarios a mi lista de contactos para establecer una relación en la que podrán consultar mutuamente el seguimiento de las rutas.
- **Eliminar usuario de mi lista:** cuando la relación haya concluido con cualquier otro empleado podemos eliminarlo de nuestra lista para evitar el seguimiento mutuo.
- **Empezar ruta:** a la hora de realizar un trayecto en horario laboral deberemos de comenzar a retransmitir nuestra posición mediante esta función.
- **Finalizar ruta:** en el momento en que alcancemos nuestro destino final deberemos hacer uso de esta función para terminar la retransmisión de nuestra posición GPS y almacenar el recorrido en nuestro historial de rutas.

- **Seguir usuario:** función que utilizaremos cuando deseemos consultar la situación geográfica de cualquier usuario de nuestra lista.
- **Añadir coche:** posibilidad de añadir los distintos coches de la flota de vehículos que la empresa nos proporciona para realizar las rutas.
- **Eliminar coche:** cuando el contrato de renting sobre un vehículo haya concluido procederemos a eliminarlo de la base de datos.

4.2.2 Requerimientos no funcionales

Estos requerimientos son muy importantes ya que especifican las necesidades software, hardware o de cualquier otro tipo, impuestos normalmente por el cliente, estándares o legislación vigente.

Al tratarse de de un proyecto académico, sólo nos tenemos que preocupar de las necesidades hardware y software.

4.3 Requerimientos del sistema

Hablaremos de los requisitos tanto del equipo que realiza el papel de servidor como los del dispositivo móvil en el que se instala la aplicación.

Los requisitos de equipo para los clientes sería únicamente la utilización de un smartphone Android con una versión del Sistema igual o superior a la 4.0 Ice Cream Sandwich API level: 17[27] y capacidad de conexión a internet y con GPS.

4.3.1 Hardware

- **Capacidad de procesamiento**: el servidor debe de ser capaz de soportar la carga de peticiones simultáneas que se le realizarán. Se recomienda al menos un procesador multinúcleo especializado para servidores.
- **Capacidad gráfica**: sin requisitos mínimos pues el único procesamiento gráfico que tiene que realizar el servidor será mostrar la interfaz del sistema por pantalla

- **Memoria**: en el caso de que el servidor sea únicamente utilizado por nuestro servicio se recomendaría únicamente 3GB de memoria RAM para evitar cualquier tipo de detención por falta de memoria.
- **Almacenamiento**: los requisitos mínimos de almacenamiento vienen dictados por el sistema operativo que en nuestro caso es Linux y al menos necesitaríamos unos 20GB para su correcto funcionamiento.
- **Conexión a internet**: es imprescindible la conexión a internet para que nuestro servicio web esté operativo la mayor cantidad de tiempo posible, preferiblemente en un porcentaje superior al 99%.
- **Pantalla**: uso único para la instalación, configuración y mantenimiento del sistema por lo que puede ser tan básica como queramos.

4.3.2 Software

- **Sistema Operativo**: el sistema sobre el que trabajaremos en este proyecto será Ubuntu 12.04[28], basado el Linux. Dicho sistema se encuentra instalado en una máquina virtual de la Universidad de Jaén. No obstante en este proyecto las herramientas utilizadas tanto Tomcat para montar el servicio web como la conexión con Base de Datos MySQL con PhpMyAdmin[29], son compatibles con Sistemas operativos de Microsoft.
- **Gestor de Bases de Datos**: la base de datos será de MySQL utilizando la herramienta phpMyAdmin funcionando en conjunto con Apache[30].
- **Navegador**: el navegador queda a nuestra elección bajo preferencias personales, aconsejo Mozilla Firefox o Google Chrome en sus últimas versiones para asegurarnos de que soporta sobradamente las tareas a realizar.
- **Resto de software**: nos referimos al software necesario para el correcto funcionamiento del servicio Weben el servidor y su correspondiente desarrollo en programación.
    - Funcionamiento del servidor:
        - Java JRE v7[31].
        - Tomcat 7 (instalado a través del terminar. Ver anexo II.)

- MySQL Java Driver Connector[32].

- Necesario para el desarrollo :
  - Axis2 (unido al IDE Eclipse para el desarrollo del servicio. Ver anexo I.)
  - Android SDK junto con Android Studio[36].

4.4 Requerimientos de la interfaz para la aplicación Android

En este punto hablaremos de cómo el desarrollo de la aplicación afecta directamente a la experiencia final del usuario por lo que integraremos una buena Usabilidad[33] para que los clientes encuentren una forma cómoda y eficiente de trabajar.

Dependiendo de la ISO que utilicemos como referencia la Usabilidad se define como:

"*La usabilidad se refiere a la capacidad de un software de ser comprendido, aprendido, usado y ser atractivo para el usuario, en condiciones específicas de uso". ISO 9126*

"*Usabilidad es la eficacia, eficiencia y satisfacción con la que un producto permite alcanzar objetivos específicos a usuarios específicos en un contexto de uso específico*". *ISO 9241*

Nos centraremos en los siguientes principios para integrar la Usabilidad en nuestra aplicación:

- **Facilidad de aprendizaje**: es el tiempo mínimo necesario desde el desconocimiento de una aplicación hasta su uso eficiente. Dicho sistema debe ser Sintetizable y Familiar.
- **Flexibilidad**: se considera como las múltiples formas de intercambiar información entre el usuario y el sistema.
- **Adaptabilidad**: correcta adaptación al usuario de manera automática.

- **Consistencia**: un sistema es consistente si todas las funciones del mismo se utilizan de la misma manera siempre que se usen y sin cambiar con el tiempo.
- **Robustez**: es la capacidad del sistema de permitir al usuario conseguir sus objetivos utilizando la aplicación sin tener problemas. Soporte a fallos.
- **Recuperabilidad**: capacidad de corregir un error del usuario.
- **Tiempo de respuesta**: para que el usuario tenga una buena experiencia los tiempos del sistema en mostrar los cambios deben de ser adecuados.
- **Adecuación de las tareas**: medida en que el sistema permite al usuario hacer las tareas que desee de la manera que él elija.
- **Disminución de la carga cognitiva**: una aplicación debe de ser lo más intuitiva posible para que el usuario se sienta cómodo y utilice el reconocimiento en lugar del recuerdo.

4.5 Análisis del sistema

Tras la especificación de los requisitos pasamos a realizar un Análisis de Sistema en el que definiremos un modelo del mismo lo más claro, consistente y completo posible.

Este análisis consta de distintas partes que definirán de la mejor manera posible el modelado del proyecto.

4.5.1 Perfiles

En este caso no existe diferenciación en los perfiles de los distintos usuarios que utilicen la aplicación y se definirán simplemente como “Empleados”.

El único requisito que deben de cumplir es el de ser empleado de la empresa pues este proyecto no implica la asignación de tareas ni la influencia de ningún empleado sobre otro sino el mero control de la situación de cada uno para posibles encuentros o controles a la hora de utilizar uno de los vehículos de empresa.

En este caso la función de un empleado “Administrador” sería únicamente la de gestionar la base de datos por lo que no se ve necesaria la implementación de una jerarquía dado que su uso es completamente independiente del puesto del empleado.

4.5.2 Casos de uso

Se definen como los pasos o tareas que deberán realizarse para llevar a cabo un objetivo concreto. Las entidades que intervienen en un caso de uso se denominan actores.

Para especificar la comunicación y el posible comportamiento del sistema entre ambos haremos uso de los *diagramas de casos de uso.* Nos sirven para mostrar cómo los requerimientos del sistema reaccionan ante los distintos eventos que se producen en el propio sistema.

El procedimiento a seguir para definir un caso de uso comienza por aclarar quienes harán el papel de actores en cada caso y para ello hay que definirlos con un nombre único para cada uno.

Los diferentes elementos que componen los casos de uso son los siguientes:

- **Actores**: aquellos usuarios que interaccionan con el sistema.
- **Casos de uso**: representados mediante una elipse y hacen referencia a las distintas tareas que se pueden llevar a cabo en el sistema.
- **Asociaciones**: la relación que se establece entre un actor y un caso de uso en el que ambos interactúan.
- **Relaciones**: representan las interacciones existentes entre distintos casos de uso y pueden ser de dos tipos:
    - **<<include>>**: aquella cuya dirección va desde la tarea elegida por el actor hacia otra tarea resultante de la misma y que representa una funcionalidad común de un caso de uso.
    - **<<extend>>**: aquella cuya dirección va desde una función opcional del sistema hasta una función común del caso de uso.

En nuestro proyecto únicamente tenemos un tipo de usuario que es el de "Empleado" pero con la diferencia de que cada uno de ellos ejerce un rol distinto en cada caso.

A partir de ahora definiremos a estos dos actores como:

- *Empleado en ruta*: es la persona que va a realizar un trayecto y activa la aplicación para retransmitir su posición.
- *Empleado observador*: es la persona que accede a la aplicación para consultar la localización de algún compañero.

Ahora ya podemos definir los diferentes casos de uso en los que intervienen los distintos requerimientos funcionales del sistema. Es usual que al aparezcan nuevos requerimientos como consecuencia de detallar el funcionamiento del sistema.

El primer diagrama de uso que deberemos de realizar es aquel en el que se muestra el funcionamiento general del sistema al que llamaremos *diagrama frontera.*

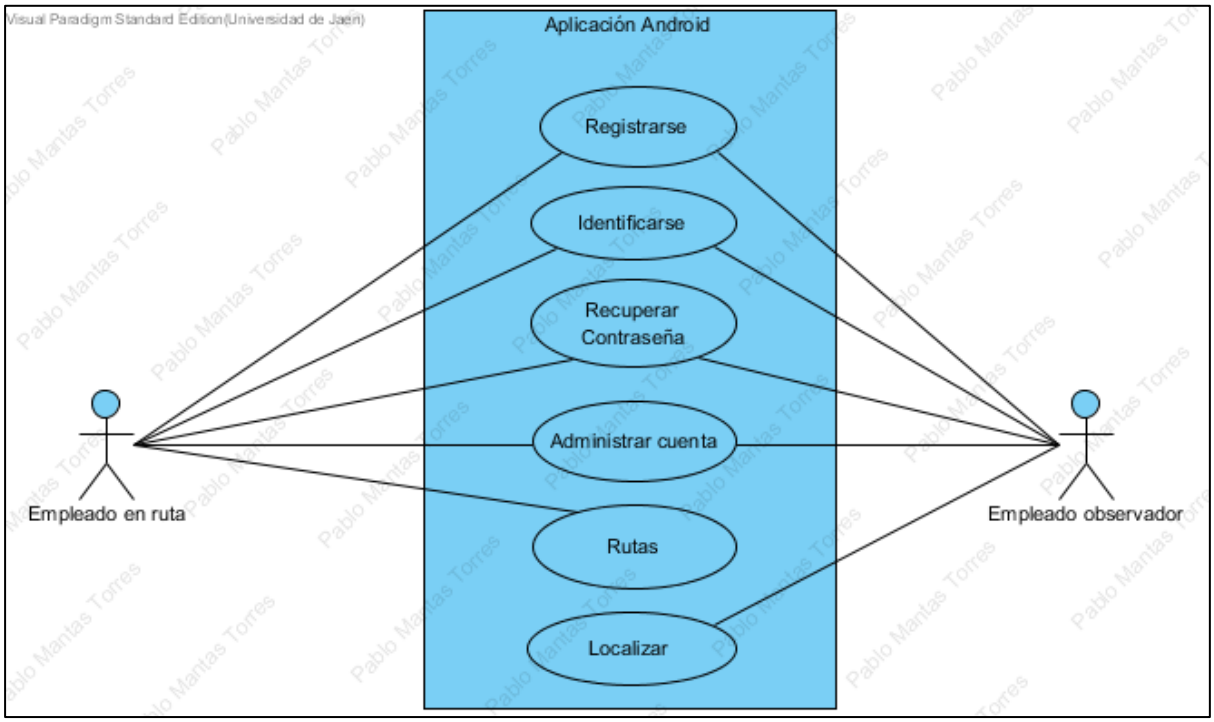


Figura 4.1: Diagrama frontera

- **Caso de Uso 1: Registrarse**
  Actores participantes: Empleado en ruta, Empleado observador

Sistema: Aplicación Android
Condiciones previas: La aplicación se está ejecutando en primer plano
Operaciones básicas:

1. El usuario introduce sus datos personales en todos los campos obligatorios. (A-1)(B-2)(C-3)

Alternativas:

A-1: Campo obligatorio en blanco.
B-2: Contraseña no coincide.
C-3: Correo electrónico no coincide.

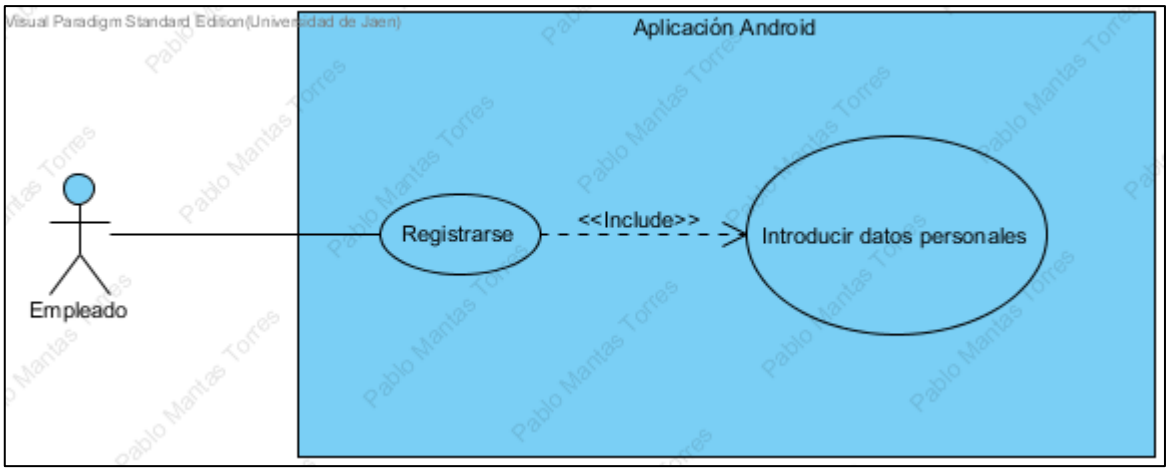


Figura 4.2: Caso de uso Registrarse

- **<u>Caso de Uso 2: Identificarse</u>**
  Actores participantes: Empleado en ruta, Empleado observador
  Sistema: Aplicación Android
  Condiciones previas: La aplicación se está ejecutando en primer plano y estar previamente registrado en el sistema
  Operaciones básicas:
  1. El usuario introduce su usuario(email) y contraseña y pulsa el botón “Entrar”. (A-1)

  Alternativas:

  A-1: Email y/o contraseña erróneo/a.

- **<u>Caso de Uso 3: Recuperar contraseña</u>**
  Actores participantes: Empleado en ruta, Empleado observador
  Sistema: Aplicación Android
  Condiciones previas: La aplicación se está ejecutando en primer plano y estar previamente registrado en el sistema
  Operaciones básicas:
  1. Introducir el nombre de email y pulsar en el botón “Enviar”. (A-1)

  Alternativas:

A-1: El email no se encuentra.

- **Caso de Uso 4: Administrar cuenta**
  Actores participantes: Empleado en ruta, Empleado observador
  Sistema: Aplicación Android
  Condiciones previas: Estar identificado en el sistema
  Operaciones básicas:
  1. Ver mis datos personales.
     1.1. Modificar datos.
     1.2. Eliminar cuenta.
  2. Ver lista de usuarios. (A-1)
     2.1. Añadir usuario. (A-1)
     2.2. Eliminar usuario. (A-1)
  3. Ver la lista de coches en el sistema. (B-2)
     3.1. Añadir coche. (B-2)
     3.2. Eliminar coche. (B-2)

  Alternativas:
  A-1: No existe ningún usuario en tu lista.
  B-2: No hay ningún coche en el sistema.

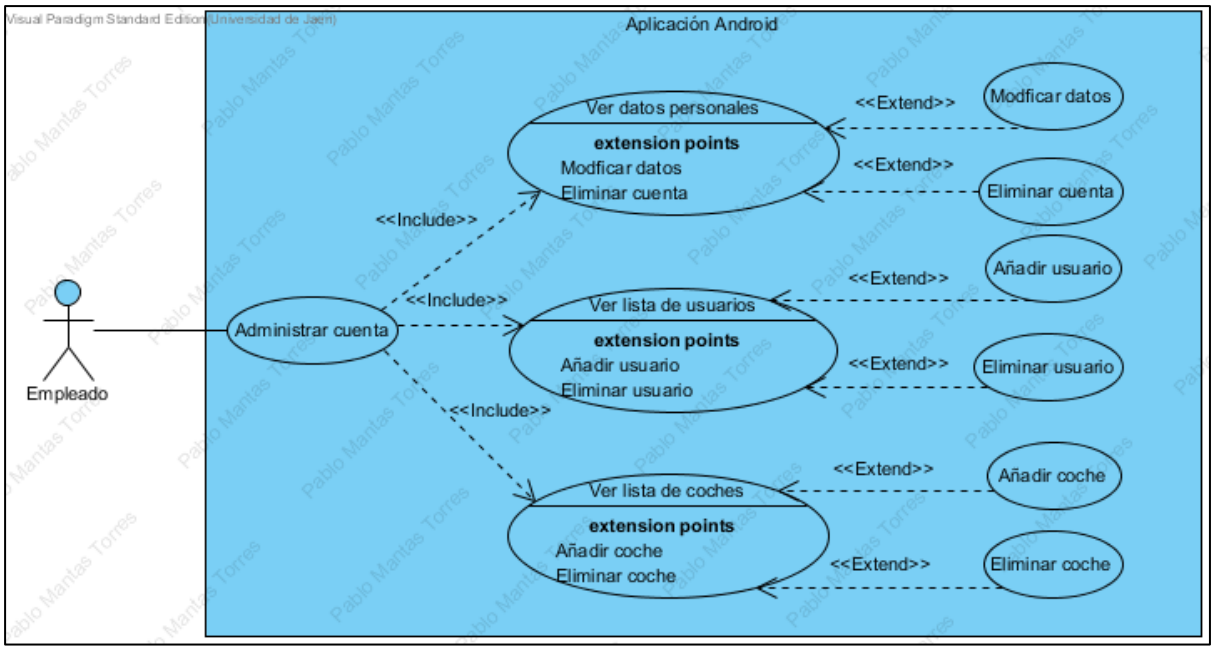


Figura 4.3: Caso de uso Administrar cuenta

- **Caso de Uso 5: Rutas**
  Actores participantes: Empleado en ruta
  Sistema: Aplicación Android
  Condiciones previas: Estar identificado en el sistema

Operaciones básicas:

1. Ver mi historial de rutas
2. Empezar ruta.
   2.1. Elegir coche. (A-1) (B-1)
   2.2. Finalizar ruta.

Alternativas:

A-1: No hay ningún coche disponible.
B-1: No se encuentra señal GPS.

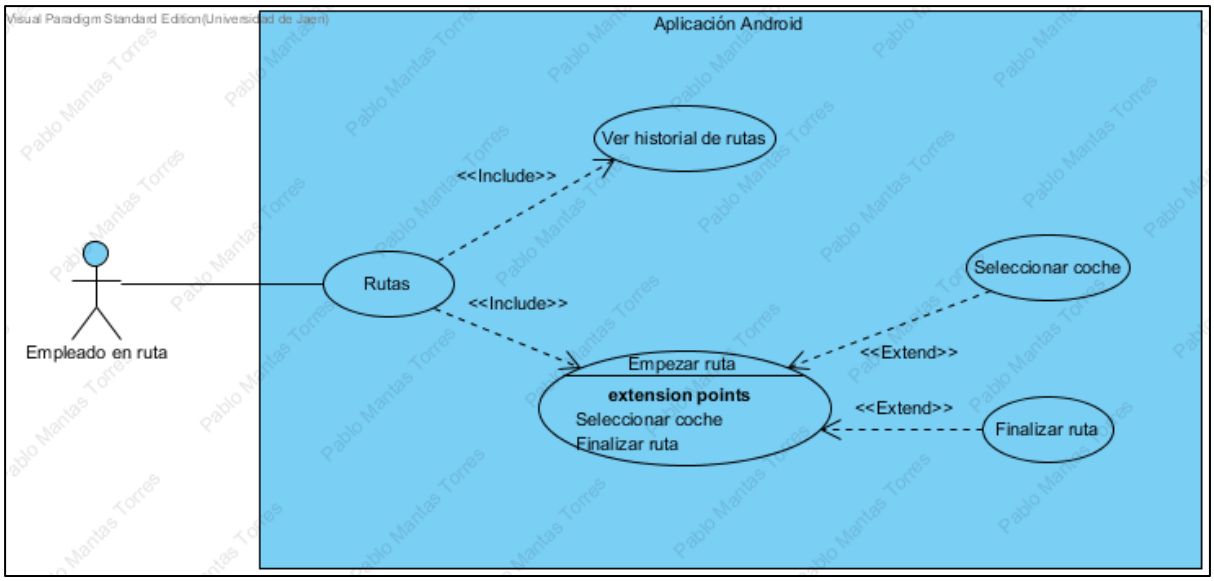


Figura 4.4: Caso de uso Iniciar ruta

- **Caso de Uso 6: Localizar**

  Actores participantes: Empleado observador
  Sistema: Aplicación Android
  Condiciones previas: Estar identificado en el sistema
  Operaciones básicas:

  1. Elegir el usuario de nuestra lista que deseemos localizar (A-1)(A-2)
     1.1. Dejar de seguir

  Alternativas:

  A-1: No hay ningún usuario en nuestra lista**.**
  A-2: No hay ningún usuario en ruta.

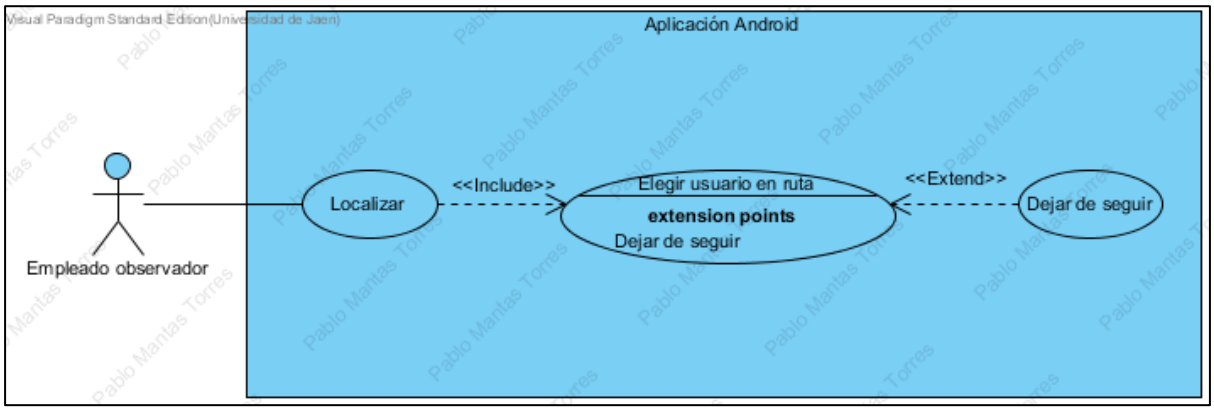


Figura 4.5: Caso de uso Localizar

4.5.3 Escenarios

Los casos recién descritos son representaciones abstractas de las funcionalidades del sistema. Para representar dichas funciones de una manera más concreta usaremos Escenarios, que no son más que "*unas historias de ficción con representación de personajes, sucesos, productor y entornos*".

Estos escenarios también nos servirán de ayuda para el desarrollo en cuanto a la resolución de problemas ya sean de interfaz o de funcionamiento interno del sistema, así como posibles carencias de cara al usuario que sean interesantes de incluir como funcionalidad.

Los componentes básicos de los escenarios serán:

- Nombre del escenario
- Descripción
- Actores principales
- Flujo de eventos

Realizaremos un Escenario por cada caso de uso

Nombre del escenario: RegistroUsuarioPablomantas

Descripción: Un nuevo usuario llamado Pablo se quiere registrar en el sistema **.**

Actores principales: Pablo, empleado de Endesa.

Flujo de eventos

1. El usuario inicia la aplicación Android.
2. Se muestra la primera actividad en la que podemos iniciar sesión, registrarlos en el sistema o recuperar contraseña.
3. El usuario elige la opción de registrarse.
4. El usuario introduce sus datos personales (Dni, Nombre, Apellidos, Edad, Email, Usuario y Contraseña); 77356749k, Pablo, Mantas Torres, 25, pablomantas@gmail.com, pablomantas, contraseña.
5. Si todo ha ido correctamente se mostrará nuevamente la pantalla de identificación.

Nombre del escenario: IdentificaciónPablomantas

Descripción: El usuario Pablomantas se quiere identificar en el sistema.

Actores principales: Pablo, empleado de Endesa.

Flujo de eventos

1. El usuario inicia la aplicación Android.
2. Se muestra la primera actividad en la que podemos iniciar sesión, registrarlos en el sistema o recuperar contraseña.
3. El usuario introduce su dirección de correo electrónico y contraseña en los campos correspondientes.
4. Si todo ha ido correctamente, se muestra la pantalla correspondiente a la administración de la cuenta de usuario.

Nombre del escenario: RecuperarContraseña

Descripción: El usuario Pablomantas se ha olvidado de su contraseña y quiere recuperarla.

Actores principales: Pablo, empleado de Endesa.

Flujo de eventos

1. El usuario inicia la aplicación Android.

2. Se muestra la primera actividad en la que podemos iniciar sesión, registrarlos en el sistema o recuperar contraseña.
3. El usuario elige la opción de recuperar contraseña.
4. El usuario introduce la dirección de correo electrónico.
5. Si la dirección existe indicará que efectivamente estaba registrado y el sistema enviará la contraseña al correo electrónico que el usuario introdujo cuando se registró.

Nombre del escenario: AdministrarCuenta

Descripción: Un empleado accede al sistema tras identificarse en él y quiere gestionar sus datos personales y usuarios de su lista.

Actores principales: Un empleado

Flujo de eventos

1. El usuario entra en la aplicación y se identifica correctamente en la misma.
2. El usuario entra en su lista de contactos.
3. Se mostrará por pantalla los empleados que este usuario tiene agregados actualmente en su lista.
4. El usuario quiere localizar a un compañero ya que próximamente tiene que reunirse con él pero no lo tiene aún en su lista de contactos por lo que introduce su nombre en el campo de búsqueda y pulsa el icono de buscar.
5. Si el nombre está correctamente escrito y dicho empleado existe, este se mostrará en pantalla.
6. Como el usuario quiere añadir a este empleado a su lista de contactos pulsará “Añadir” sobre el empleado.
7. Si todo ha ido correctamente se habrá enviado una notificación al empleado buscado y en cuanto este acepte la solicitud aparecerá en la lista de contactos del usuario.
8. Una vez finalizada la tarea decide cerrar sesión y salir de la aplicación.

Nombre del escenario: RutasEmpleado

Descripción: Un empleado accede al sistema tras identificarse en él y quiere gestionar sus datos personales y usuarios de su lista.

Actores principales: Un empleado

Flujo de eventos

1. El usuario entra en la aplicación y se identifica correctamente en la misma.
2. Decide entrar en la sección de Rutas del sistema.
3. El usuario va a salir en un coche de empresa para realizar una inspección a una instalación por lo que elige la opción de “Comenzar ruta”.
4. El usuario escoge de entre los coches disponibles actualmente el que va a utilizar para dicho viaje.
5. Si todo ha ido correctamente el sistema comenzará a actualizar la posición del Smartphone periódicamente.
6. El usuario cierra la aplicación durante su trayecto pero no la sesión.
7. Una vez finalizado el trayecto abre nuevamente la aplicación y finaliza la ruta.
8. Tras este paso cierra sesión en el sistema y sale de la aplicación.

Nombre del escenario: LocalizarEmpleado

Descripción: Un empleado accede al sistema tras identificarse en él y quiere gestionar sus datos personales y usuarios de su lista.

Actores principales: Un empleado

Flujo de eventos

1. El usuario entra en la aplicación y se identifica correctamente en la misma.
2. Decide entrar en la sección de Contactos.
3. Se mostrará en pantalla su lista de contactos y aparecerán habilitados aquellos que se encuentren en ruta.
4. El usuario selecciona aquel empleado de su lista que desea localizar pues tiene que tratar un asunto importante con él en persona y han en su propio despacho.
5. Si todo ha ido bien aparecerá en pantalla un mapa con la situación actual del empleado seleccionado.
6. Tras localizar al empleado sale de la localización, cierra sesión y la aplicación.

Con estos escenarios queda definido como funciona el sistema normalmente. Estos escenarios se han centrado en alguna de las tareas que puede llevar a cabo un usuario en el sistema.

4.6 Diseño

En esta sección definiremos el diseño empleado en el desarrollo de la aplicación. Para empezar entendemos el Diseño de Software como "*el proceso de aplicar distintas técnicas y principios con el propósito de definir un dispositivo, proceso o sistema con suficientes detalles como para permitir su realización física"* por Taylor en 1959.

El primer paso en esta fase es la construcción del sistema desde el punto de vista del dominio de la solución (sistema software) donde identificaremos cuales son los atributos principales de cada clase. Será el diseño el encargado de determinar cómo se introducen dichas clases en el sistema, su visualización y almacenamiento en la base de datos.

El diseño del proyecto entero se dividirá en varias fases para reducir su complejidad:

- Fase estructura del sistema.
- Fase estructura de los datos.
- Definición de la interfaz.

4.6.1 Estructura del sistema

Aquí definiremos las clases que intervendrán en el proyecto. Para el análisis de las mismas utilizaremos los Diagramas de Clases que son un tipo de diagramas orientado a objetos y que describen la estructura del sistema mediante sus clases.

Las partes de los diagramas de clases se componen de las siguientes partes:

- **Clases:** Conjunto de objetos similares en estructura, comportamiento específico, atributos y métodos y se denominan instancias de clase.
- **Relaciones:** Entenderemos relaciones como subconjuntos de clases iguales y existen dos tipos de relaciones:
    - De **especialización:** son subclases más detalladas queparten de otra más general.

- De **generalización:** son las clases de las que deriban las especializadas y se dice que son superclases.

Diagrama de clases de la aplicación:

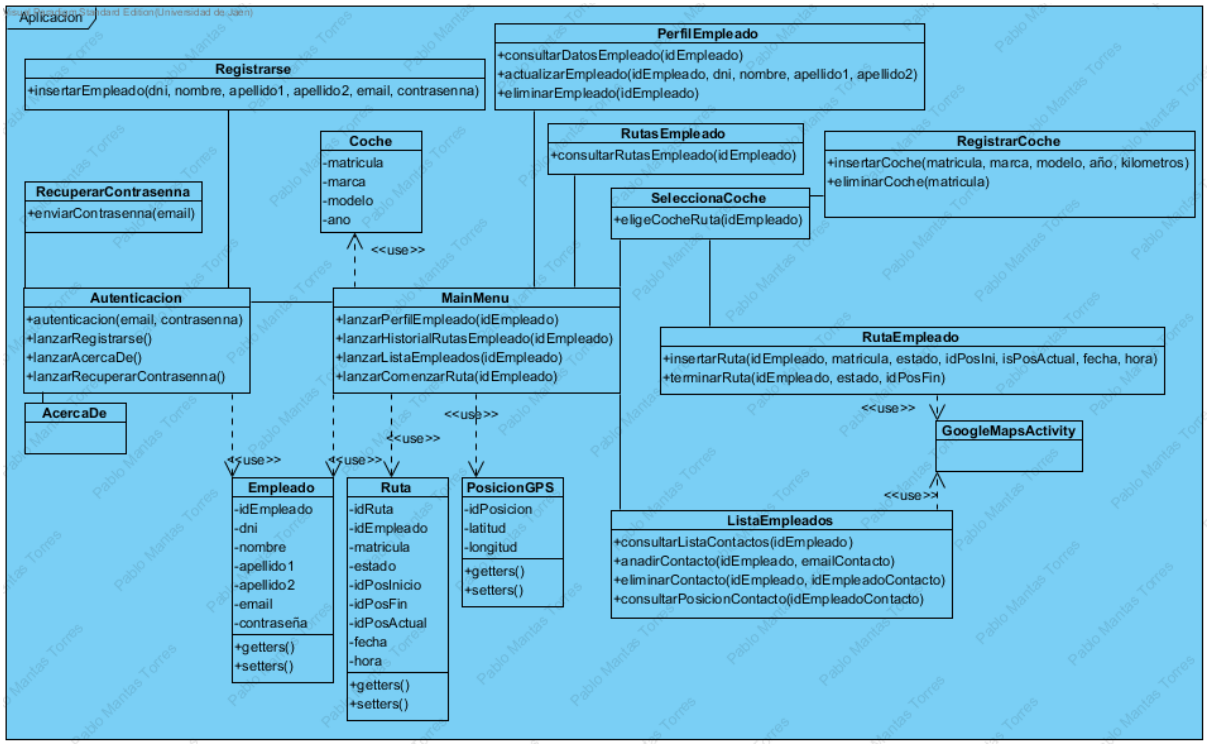


Figura 4.6: Diagrama de clases de la Aplicación

En este diagrama encontramos las clases necesarias para el correcto funcionamiento de nuestra aplicación y que comentaremos brevemente a continuación:.

Primero tenemos las clases posiciónGPS, empleado, coche, ruta y googleMapsActivity (necesaria para utilizar mapas y su posicionamiento GPS), las cuales son nuestro modelo de objetos.

Nuestra clasemadre o principal será la de autenticación pues será la que nos dé acceso a la funcionalidad de la aplicación.

Otra clase fundamental es el menuEmpleado y las que se asocian a ella (rutaEmpleado, listaContactos, listaCoches y modificarEmpleado).

Y por último tendremos la clase acercaDe que simplemente nos mostrará una ventana con información sobre autor y título del Trabajo Fin de Grado.

Diagrama de clases del servicio web:

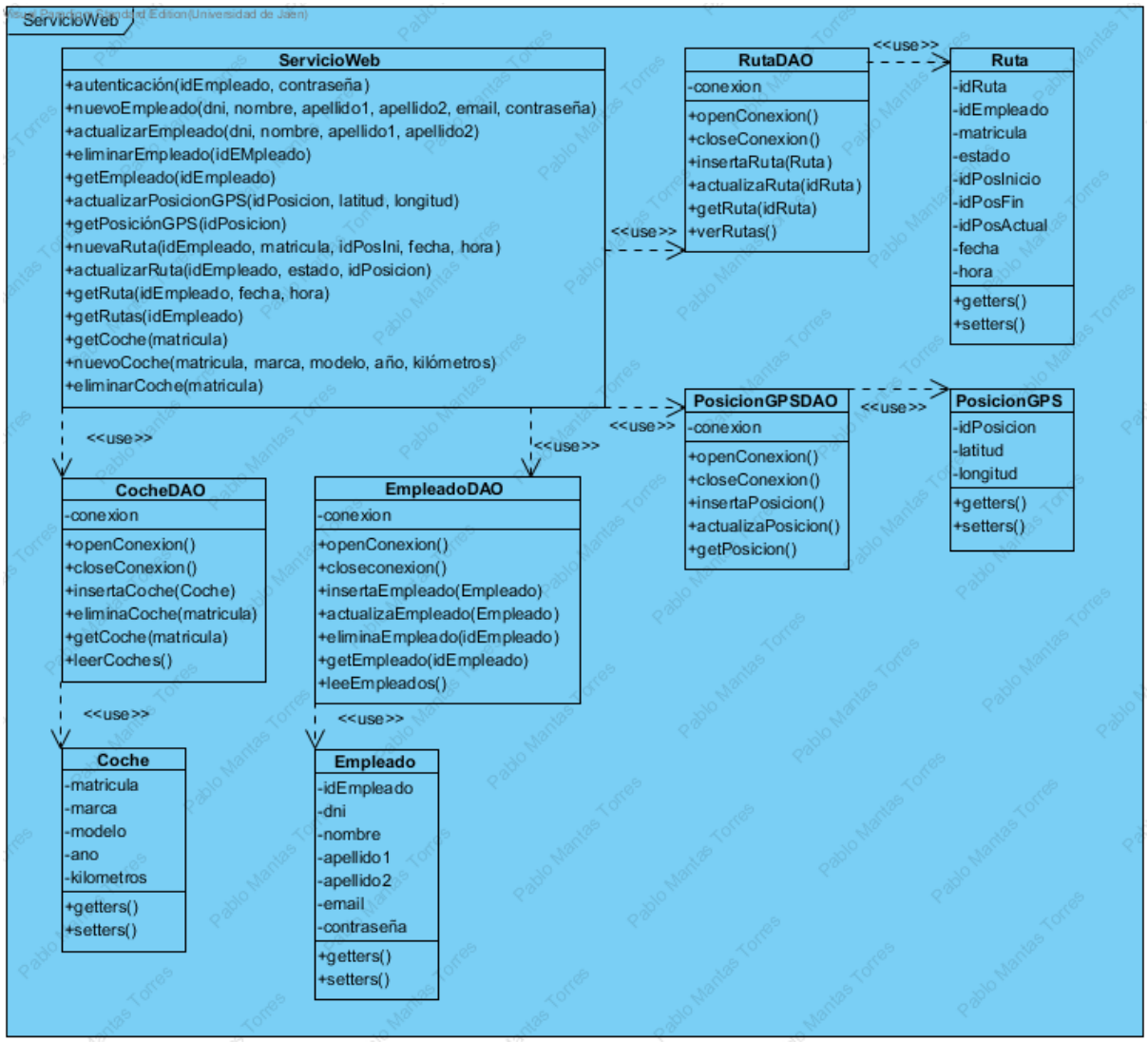


Figura 4.7: Diagrama de clase dl Servicio Web

En el diagrama encontramos al igual que en el anterior todo lo necesario para el funcionamiento del servicio web.

Las clases posiciónGPS, empleado, coche y ruta definen los tipos de objetos utilizados por el servicio.

Las clases posiciónGPSDAO, empleadoDAO, cocheDAO y rutaDAO albergan la interacción con los datos mediante funcionalidades (insertar, actualizar, consultar y eliminar) y la base de datos MySQL.

Y finalmente tenemos la clase servidorWeb que contiene nuestro main del servicio web con sus funcionalidades y sus correspondientes URIs para acceder a ellas.

4.6.2 Estructura de los datos

En la Ingeniería de Software se debe determinar la estructura de los datos almacenados en el propio sistema ya que con ellos trabajaremos una vez esté en funcionamiento el sistema.

En nuestro proyecto almacenaremos la información según tres tipos de elementos:

Los **empleados** cuyos atributos son**:** IdUsuario, Nombre, Apellidos, Edad, Email, Alias, DNI y provincia.

Las **rutas** cuyos atributos son: IdUsuario, Fecha, Hora, Coche, idPosInicio, idPosFinal, idPosActual.

Las **posiciones GPS** compuesta por un idPosición, latitud y longitud.

Los **coches** cuyos atributos son: Matrícula, Marca, Modelo y Km.

Modelo Entidad-Relación

Este modelo de datos se basa en una percepción del mundo real en el que hay una colección de objetos denominados entidades y en las relaciones entre ellos.

- **Entidades**: Son “objetos” del mundo real y únicos aunque sean del mismo tipo. Estos pueden ser físicos como un coche, una persona, etc ; o conceptuales como un nombre, una idea, etc. Además podemos encontrar dos tipos de entidades

  - **Fuertes**: Este tipo de entidades son todas aquellas que se identifican como únicas o independientes.
  - **Débiles**: Son aquellas que no pueden existir sin estar relacionada con otra entidad.
- **Atributos**: Características que definen a las entidades. Suelen ser sólo las más importantes de dichas entidades.
- **Relaciones**: Dependencias existentes entre entidades. Se representan mediante un rombo en los diagramas de entidad-relación con una etiqueta en el interior.
  - **Atributos en las relaciones**: Podemos encontrar atributos propios de estas mismas relaciones.
- **Claves**: Son algunos de los atributos comunes de una colección de entidades y permiten su identificación. Existen diferentes tipos:
  - **Superclave**: Subconjunto de atributos que distingue las distintas entidades de una colección.
  - **Clave candidata**: En el caso de que una superclave dejase de serlo al quitar uno de sus atributos, esta pasaría a ser una clave candidata.
  - **Clave primaria**: Una clave candidata que identifica un conjunto de entidades.
- **Cardinalidad**: Es una etiqueta en cada extremo de una relación y pueden ser de tipo uno a uno “1:1”, uno a muchos”1:N” y de muchos a muchos “N:M”.
- **Herencia**: Es una relación que se establece entre una entidad “padre” y una entidad “hijo” en la que la entidad hijo hereda los atributos de la entidad padre y se representa con triángulos.
- **Agregación**: Es una abstracción en la que las relaciones son tratadas como entidades y expresan una relación entre relaciones o entidades-relaciones. Se representan englobando el conjunto de objetos que intervienen en ella dentro de un rectángulo.

Tras definir los componentes de los diagramas de entidad-relación pasamos a desarrollar el modelo cumpliendo los siguientes pasos:

1. Incluir los elementos de nuestro sistema en el Esquema Conceptual.
2. Transformar dicho Esquema Conceptual en uno conocido como Esquema Conceptual Modificado.
3. Y finalmente definiremos las tablas de nuestra base de datos a partir del Esquema Conceptual Modificado y normalizarlas después.

Modelo Entidad-Relación y la Normalización

La normalización de una base de datos consiste en aplicar ciertas reglas a las relaciones del modelos entidad-relación de la misma.

Con este proceso evitamos que en nuestra base de datos se produzca la redundancia de datos, reduce los posibles problemas a la hora de actualizar datos y protege la integridad de los mismos.

En un modelo de base de datos relacional consideramos tablas a las relaciones pero para ello tienen que cumplir ciertas restricciones:

- Tienen que tener un nombre único.
- No pueden repetir filas.
- Los datos de una columna tienen que ser del mismo tipo.

Consideramos que una base de datos está en una forma normal N cuando todas sus tablas están en esa forma.

Para cubrir estas necesidades suele ser suficiente con las tres primeras formas normales, cuyo creador es Edgar F. Codd[34].

- **Primera Forma Normal (1FN)**: podemos considerar una tabla en 1FN si cada uno de sus atributos son atómicos (indivisibles), contiene una

clave primaria única(uno o varios atributos) y sin atributos nulos, el resto de campos de la tabla se identifican a partir de esta misma y debe existir independencia del orden de filas y columnas.

- **Segunda Forma Normal (2FN)**: consideramos que esta en 2FN si está en 1FN y los atributos que no son clave dependen completamente de la clave principal.
- **Tercera Forma Normal (3FN)**: finalmente nuestra tabla estará en 3FN si es 2FN y no existen dependencias funcionales transitivas entre atributos que no son clave.

Esquema Conceptual

Convertimos los componentes en entidades:

- Empleados (empleados).
- Rutas (rutas).
- Coches (coches).

Las relaciones entre ellos son:

- Los empleados tienen en su lista a otros empleados.
- Los empleados realizan rutas y una ruta es realizada por un empleado.
- Un coche es empleado en una ruta y en una ruta se emplea un coche.

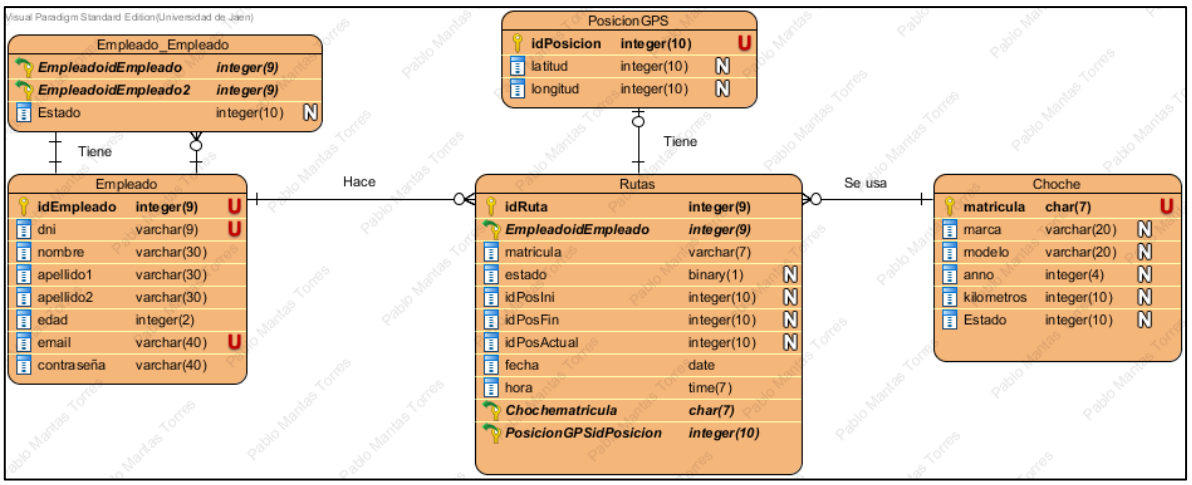


Figura 4.8: Esquema conceptual

Esquema Conceptual Modificado

Para obtener el Esquema Conceptual Modificado aplicaremos ciertos cambios al Esquema Conceptual que ya tenemos:

- Eliminar las entidades débiles.
- Eliminar las relaciones con atributos.
- Eliminar las relaciones de muchos a muchos.

Tablas de la Base de Datos del Sistema

CAMPO	TIPO	CLAVE	DECRIPCIÓN
idEmpleado	INT(9)	PRIMARY	Identificador de empleado
dni	VARCHAR(9)		Documento nacional de identidad del empleado
nombre	VARCHAR(30)		Nombre del empleado
apellido1	VARCHAR(30)		Primer apellido del empleado
apellido2	VARCHAR(30)		Segundo apellido del empleado
email	VARCHAR(40)		Correo electrónico del empleado
contraseña	VARCHAR(40)		Contraseña del empleado

Tabla 4.1: Tabla de Empleado

CAMPO	TIPO	CLAVE	DECRIPCIÓN
idRuta	INT(9)	PRIMARY	Identificador de ruta
idEmpleado	INT(9)	FOREING_KEY	Identificador de empleado
matricula	VARCHAR(7)		Matrícula del coche
estado	DEF(“0”,”1”)		Flag del estado de la ruta “0” para finalizada “1” para en curso
idPosIni	INT(10)		Identificador de la posición de inicio de la ruta
idPosFin	INT(10)		Identificador de la posición de en que finalizó la ruta
idPosActual	INT(10)		Identificador de la posición actual del vehiculo
fecha	DATE		Fecha en que se realiza la ruta
hora	TIME		Hora a la que se inicia la ruta

Tabla 4.2: Tabla de ruta

CAMPO	TIPO	CLAVE	DECRIPCIÓN
matricula	VARCHAR(7)	PRIMARY	Matrícula del coche
marca	VARCHAR(20)		Marca del coche

modelo	VARCHAR(20)	Modelo del coche
anno	INT(4)	Año de matriculación del coche
Kilometros	INT(6)	Número de kilómetros que tiene el coche

Tabla 4.3: Tabla de coche

CAMPO	TIPO	CLAVE	DECRIPCIÓN
idEmpleado 1	INT(9)	FOREING_KEY	Id del empleado que tiene la lista
idEmpleado2	INT(9)	FOREING_KEY	Id del empleado que pertenece a la lista
estado	DEF(“0”,”1”)		“1” para aceptado “0” Para pendiente

Tabla 4.4: Tabla lista de contactos

CAMPO	TIPO	CLAVE	DECRIPCIÓN
idPosicion	INT(10)	PRIMARY	Identificador de una posición GPS registrada
latitud	FLOAT		Latitud de la posición GPS
longitud	FLOAT		Longitud de la posición GPS

Tabla 4.5: Tabla posicionGPS

4.7 Interfaz

El diseño de la interfaz de nuestra aplicación es uno de las partes más importantes del desarrollo de la misma puesto que puede ser la diferencia entre usar la aplicación por parte de todos lo empleados y que quede en desuso por su complejidad. Tenemos que buscar siempre que la experiencia del usuario al utilizar cualquiera de las funcionalidades de nuestra aplicación sea la mejor posible.

4.7.1 Guía de estilo

Definiremos una guía de estilo para poder seguir un patron o estilo común según vaya evolucionando la aplicación y tenga coherencia su interfáz sea cual sea la versión o módulo que evolucionemos o desarrollemos.

La descripción de las normas de estilo a seguir son:

- Fuentes
    - Cabeceras
        - Tipo de letra: Arial

        - Tamaño: 22dp
        - Color: Negro
        - Formato: Normal
    - Contenidos
        - Tipo de letra: Arial
        - Tamaño: 20dp
        - Color: Negro
        - Formato: Normal
- Menús
    - Opciones
        - Tipo de letra:
        - Tamaño: 22dp
        - Color: Blanco
        - Formato: Normal
- Logotipo: Icono que se muestra en la pantalla de inicio del sistema Andoroid.

4.7.2 Storyboard

Una de las partes más importantes del diseño de la interfáz de nuestra aplicación ya que proporcionan un prototipo sin funcionalidad real de lo que podría ser la aplicación y se suelen usar para presentarlo al cliente o a los usuarios finales de la aplicación y poder evaluar y encontrar posibles mejoras antes de la implementación y despliegue final de la aplicación.

4.8 Implimentación

Este es el último paso que teóricamente se incluye en la Ingeniería del Software. Aquí procedemos a coger todo nuestro análisis, extraer el modelo obtenido y transformarlo mediante código escrito en el lenguaje de programación que hayamos elegido mediante herramientas de desarrollo.

4.8.1 Arquitectura del sistema

Nuestro sistema se divide en dos tipos diferentes de arquitectura, la del servicio web que será de tipo cliente-servidor y la de nuestra aplicación que será cliente-interfaz.

El usuario inicia la aplicación y realiza una petición al servidor y este tiene que utilizar las funciones definidas para poder encontrar la respesta adecuada a la petición.

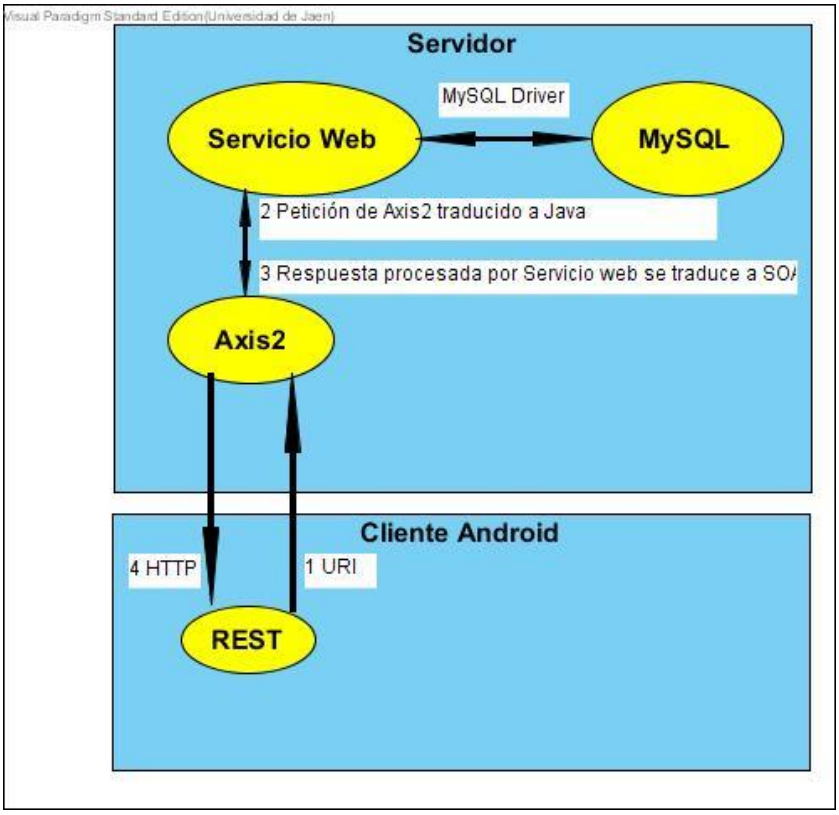


Figura 4.9: Arquitectura cliente-servidor

4.8.2 Lenguajes implicados

En este proyecto tanto la aplicación como el servicio web trabajan con Java aunque cada uno de ellos haga uso de recursos y librerías diferentes.

Java es un lenguaje de programación orientado a objetos, multiplataforma y con soporte nativo para el trabajo en red.

Cuando nos referimos a “orientado a objetos” estamos hablando del método de programación que se orienta a diseñar software de manera que los distintos tipos de datos estén asociados a sus propias funciones y así agruparlos en objetos (datos + funciones). El principio de esta metodología es la de separar las partes del estado de los datos de aquella que se encarga de alterarlos para poder modificar las estructuras de datos de manera estable.

Otra característica importante de Java es la recolección de basura, gracias a la cual evitamos las fugas de memoria siendo el entorno de ejecución de Java el que se encarga de gestionar el tiempo de vida de los objetos. Su funcionamiento consiste en que cuando un objeto deja de estar referenciado el recolector de basura se encarga de borrar el elemento y así liberar memoria.

4.8.3 Herramientas de desarrollo

Para nuestro servicio web he utilizado el IDE (entorno de desarrollo) Eclipse Kepler para Java Developers junto con librerias Axis2 1.6.1 y el driver de conexión Java a MySQL, *mysql-connector-java-5.1.31-bin.*

Para el dasarrollo de la aplicación he utilizado el SDK oficial de Google junto con Android Studio.

Para el soporte a la base de datos he utilizado PhpMyAdmin que maneja base de datos MySQL y funciona sobre Apache.

Para conseguir que el servicio web y la base de datos se comuniquenn por puertos distintos en el servidor he utilizado NGINX y así poder utilizar Apache para PhpMyAdmin y Apache Tomcat 7 para el servicio web.

4.9 Funcionamiento del sistema

El funcionamiento general de la aplicación para el tratamiento de cualquier acción se podría resumir como::

1. El usuario se autentica en la aplicación introduciendo su email y contraseña y posteriormente pulsa el botón de “Iniciar”.
2. La aplicación construye la petición con esos datos para que Axis2 pueda entenderla y la envía a una URI concreta para la autenticación.
3. Una vez la haya obtenido y entendido Axis2 es traducida de nuevo a Java para que el servicio web pueda entenderlo e interactuar con la base de datos para preparar la respuesta.
4. Una vez esté listo el resultado el servicio web se encarga de devolver la respuesta como listas (List<String>).
5. De nuevo Axis2 toma la respuesta y la envía en SOAP por medio de HTTP.
6. Aquí ya la aplicación recibe los datos en la función que ejecutóla petición y pasa la respuesta al manejador para convertirla a objetos Java.
7. Una vez en Java estos son devueltos a la función y tratados según el código considere oportuno.

4.10 Pruebas y validación

Para comprobar el correcto funcionamiento de nuestra aplicación y detectar posibles bugs definiremos distintos test que posteriormente aplicaremos.

4.10.1 Pruebas

Prueba 1: Registrarse

Tareas	Introducir nuestros datos de registro
Entradas	Email, contraseña, nombre, dni, apellido1 y apellido 2
Salidas	“OK” o “ERROR DE REGISTRO”
Roles	Empleado en ruta o empleado observador

Prueba 2: Autenticarse

Tareas	Autenticarse en la aplicación
Entradas	Email y contraseña del usuario con id=2
Salidas	“OK” o “ERROR DE AUTENTICACIÓN”
Roles	Empleado en ruta o empleado observador

Prueba 3: Recordar contraseña

Tareas	Pulsar en “Recuperar contraseña”
Entradas	Email
Salidas	Nuestra contraseña por email, Sistema responde 1 o 0
Roles	Empleado en ruta o empleado observador

Prueba 4: Modificar datos personales

Tareas	Introducir nuevos datos personales
Entradas	Nombre, Apellidos,…
Salidas	“OK” o “ERROR AL ACTUALIZAR”
Roles	Empleado en ruta o empleado observador

Prueba 5: Dar de alta un coche

Tareas	Introducir datos coche
Entradas	Matrícula, modelo y color
Salidas	“OK” o “ERROR DE REGISTRO”
Roles	Empleado en ruta o empleado observador

Prueba 6: Dar de baja un coche

Tareas	Pulsar en eliminar un coche
Entradas	Matrícula
Salidas	“OK” o “ERROR AL BORRAR”
Roles	Empleado en ruta o empleado observador

Prueba 7: Añadir usuario a nuestra lista

Tareas	Enviar una solicitud a otro usuario
Entradas	Email del usuario en cuestión
Salidas	“OK” o “ERROR DE ENVIO”
Roles	Empleado en ruta o empleado observador

Prueba 8: Eliminar usuario de nuestra lista

Tareas	Pulsar en eliminar usuario
Entradas	Email del usuario en cuestión
Salidas	“OK” o “ERROR AL BORRAR”
Roles	Empleado en ruta o empleado observador

Prueba 9: Comenzar ruta

Tareas	Pulsar en el botón de iniciar ruta
Entradas	Posición de inicio
Salidas	Posición actual o “ERROR DE POSICION”
Roles	Empleado en ruta

Prueba 10: Detener ruta

Tareas	Pulsar botón de finalizer ruta
Entradas	Posición final
Salidas	“OK” o “ERROR”
Roles	Empleado en ruta

Prueba 11: Comenzar a seguir a un usuario de nuestra lista

Tareas	Ver lista de usuarios
Entradas	Id del usuario en cuestión
Salidas	Posición actual del usuario o “ERROR DE POSICION”
Roles	Empleado observador

Prueba 12: Dejar de seguir a un usuario de nuestra lista

Tareas	Pulsar en “Dejar de seguir”
Entradas	Id del usuario en cuestión
Salidas	“OK” o “ERROR AL BORRAR”
Roles	Empleado observador

Prueba 13: Consultar el historial de rutas

Tareas	Ver el historial de rutas
Entradas	Id propio
Salidas	Lista de rutas realizadas por nuestro propio usuario
Roles	Empleado en ruta o empleado observador

Prueba 14: Darnos de baja del sistema

Tareas	Pulsar en “Darnos de baja”
Entradas	Id propio
Salidas	“OK” o “ERROR AL BORRAR”
Roles	Empleado en ruta o empleado observador

4.10.2 Resultados

Prueba 1	
Resultado	**<ns:return>1</ns:return>// “OK”**
Prueba 2	
Resultado	**<ns:return>2</ns:return>// “OK”**
Prueba 3	
Resultado	**<ns:return>1</ns:return>// “OK”**
Prueba 4	
Resultado	**<ns:return>1</ns:return>// “OK”**
Prueba 5	
Resultado	**<ns:return>1</ns:return>// “OK”**
Prueba 6	
Resultado	**<ns:return>1</ns:return>// “OK”**
Prueba 7	
Resultado	**<ns:return>1</ns:return>// “OK”**
Prueba 8	
Resultado	**<ns:return>1</ns:return>// “OK”**
Prueba 9	
Resultado	**<ns:return>2</ns:return> // idEmpleado <ns:return>9</ns:return> // idRuta <ns:return>13</ns:return> // idPosIni <ns:return>14</ns:return> // idPosActual**
Prueba 10	
Resultado	**<ns:return>1</ns:return>// “OK” <ns:return>14</ns:return>// idPosFin <ns:return>0</ns:return>// estado**
Prueba 11	
Resultado	**<ns:return>3</ns:return>// idEmpleado <ns:return>16</ns:return>// idPosActual <ns:return>1</ns:return>// estado**
Prueba 12	

Resultado	**<ns:return>1</ns:return>// "OK"**
Prueba 13	
Resultado	**<ns:return>1</ns:return>// "OK"**
Prueba 14	
Resultado	**<ns:return>1</ns:return>// "OK"**

Tabla 4.6: Tabla de resultados

5 CONCLUSIONES

Este proyecto se desarrolla a partir de una propuesta de trabajo genérico que consistía en la gestión y acceso a los servicios de una empresa mediante aplicación movil.

En un principio la idea era la de desarrollar una aplicación que permitiese a los padres, pareja o amigos conocer la posición de una persona cuando esta se encuentra de viaje para así poder despreocuparse de accidentes o poder saber la hora de llegada.

A partir de esta idea lo adapté al ámbito de una empresa, en concreto Endesa donde recientemente estuve realizando unas prácticas y vi una posible aplicación para conocer que vehículos están disponibles en cada momento en una oficina para los técnicos que necesiten usarlos para desplazarse puedan saberlo en cada momento y en el caso de que no haya ninguno disponible poder saber donde se haya el más cercano o cuanto le queda para volver.

A pesar de que los objetivos principales y las funciones definidas durante la memorian han sido desarrolladas y funcionan correctamente, aún queda mucho margen de mejora.

5.1 Posibles trabajos futuros

Una posible evolución de esta aplicación sería la de aplicar la misma al ámbito social de una familia o grupos de amigos donde se podría seguir el trayecto en coche de un hijo por parte de sus padres hasta crear grupos de reunión ya sea de

amigos o de trabajo y saber donde se encuentra cada miembro con respecto al lugar de encuentro preestablecido.

6 BIBLIOGRAFÍA

[01] "***El primer smartphone de la historia***" http://techyzmundo.com/sabes-cual-fue-el-primer-smartphone-android-de-la-historia/

Visitado el: 13/10/2015 12:32

[02] "***RESTful Web Services***" O'Reilly. Leonard Richardson & Sam Ruby. 2007

Visitado el: 22/05/2015 12:05

[03] Página oficial de **WS-I** http://www.ws-i.org

Visitado el: 24/05/2015 10:22

[04] "***Servicio Web***" http://www.w3c.es/Divulgacion/GuiasBreves/ServiciosWeb

Visitado el: 25/05/2015 11:02

[05] Página de **OASIS** https://www.oasis-open.org

Visitado el: 27/05/2015 16:55

[06] Página oficial de **la World Wide Web** http://www.w3.org

Visitado el: 23/09/2015 10:40

[07] ***"Estándares de los servicios Web información"*** http://www.eumed.net/tesis-doctorales/2007/cavl/Estandares%20de%20los%20servicios%20Web.htm

Visitado el: 05/06/2015 17:30

[08] ***"Información útil sobre COBRA"*** http://web.archive.org/web/20070517212806/http://agamenon.uniandes.edu.co/~revista/articulos/corba/corba.htm

Visitado el: 06/06/2015 10:02

[09] ***"Remote Method Invocation Home"*** http://www.oracle.com/technetwork/java/javase/tech/index-jsp-136424.html

Visitado el: 20/09/2015 9:30

[10] ***"Spring Framework"*** http://projects.spring.io/spring-framework/

Visitado el: 25/09/2015 11:21

[11] ***"Streaming API for XML"*** http://docs.oracle.com/cd/E17802_01/webservices/webservices/docs/1.6/tutorial/doc/SJSXP.html

Visitado el: 25/09/2015 12:00

[12] "**AXIOM**" https://ws.apache.org/axiom/

Visitado el: 22/09/2015 12:45

[13] ***"Panorámica de MySQL AB"*** http://dev.mysql.com/doc/refman/5.0/es/what-is-mysql-ab.html

Visitado el: 22/09/2015 7:25

[14] ***"Página oficial de la Apache Software Foundation"*** http://www.apache.org/

Visitado el: 01/09/2015 11:00

[15] ***"Meet Tomcat Catalina"*** https://www.mulesoft.com/tcat/tomcat-catalina

Visitado el: 15/09/2015 11:20

[16] ***"What is RPC?"*** http://technet.microsoft.com/en-us/library/cc787851(v=ws.10).aspx

Visitado el: 15/09/2015 13:00

[17] Página oficial de **iOS** https://www.apple.com/es/iphone/

Visitado el: 13/10/2015 11:44

[18] Página oficial de **Windows Phone** (España) http://www.windowsphone.com/es-ES

Visitado el: 19/10/2015 11:46

[19] Página sobre **XML** en la W3 http://www.w3.org/XML/

Visitado el: 22/05/2015 17:50

[20] Sitio oficial de **SQLite** http://www.sqlite.org/

Visitado el: 20/09/2015 9:22

[21] Página oficial de **Adobe Flash** http://www.adobe.com/es/products/flashplayer.html

Visitado el: 02/09/2015 8:22

[22] Página oficial de **OpenGL** http://www.opengl.org/

Visitado el: 06/09/2015 9:55

[23] Página oficial de **Handset Alliance** http://www.openhandsetalliance.com/

Visitado el: 05/05/2015 8:00

[24] Página oficial de **Android** http://www.android.com/

Visitado el: 24/09/2015 12:05

[25] ***"Ingeniería del Software"*** "Iam Sommerville. Addison Wesley"

Visitado el: 04/04/2015 10:00

[26] ***"Google Maps Android API"*** https://developers.google.com/maps/documentation/android-api/

Visitado el: 03/10/2015 21:50

[27] ***"Android Ice Cream Sandwich 4.2 API 17"*** http://developer.android.com/about/versions/android-4.2.html

Visitado el: 01/09/2015 20:21

[28] "**Ubuntu 12.04**" http://releases.ubuntu.com/12.04/; Visitado el: 09/04/2015 22:21

[29] "**PhpMyAdmin**" http://www.phpmyadmin.net/home_page/index.php

Visitado el: 07/09/2015 12:55

[30] "**Apache**" http://httpd.apache.org/

Visitado el: 07/09/2015 13:40

[31] ***"Java JRE v7 Downloads"*** http://www.oracle.com/technetwork/java/javase/downloads/java-se-jre-7-download-432155.html

Visitado el: 24/07/2015 11:00

[32] ***"MySQL Java Connector"*** http://dev.mysql.com/downloads/connector/j/

Visitado el: 14/07/2015 11:45

[33] ***"The Invisible Computer"*** "Donald A. Norman. 1999" ; Visitado el: 04/04/2015 12:14

[34] Edgar F. Codd http://amturing.acm.org/award_winners/codd_1000892.cfm

Visitado el: 04/04/2015 13:22

[35] ***"Middleware información"*** http://www.oracle.com/lad/products/middleware/cloud-app-foundation/weblogic/overview/index.html

Visitado el: 23/09/2015 10:55

[36] Página oficial **Android Studio Developers** https://developer.android.com/sdk/index.html

Visitado el: 01/10/2015 22:46

ANEXO A: Manual de instalación del Servicio Web

En primer lugar necesitaremos las siguientes herramientas para poder poner en funcionamiento nuestro servicio web:

- Java
- MySQL-Server
- Nginx
- PhpMyAdmin
- Apache2
- Tomcat 7
- MySQL Java Connector

El método más sencillo y el que usaremos en la mayoría de los casos es el de instalación mediante terminal (shell Ubuntu 12.04) y otra manera será la de descarga directa y posterios instaliónen el lugar correspondiente.

Ya que tendremos a tres herramientas funcionando por defecto en el mismo puerto 80, tendremos que cambiar a Apache2 al puerto 81 y a Tomcat al puerto 82, dejando a Nginx trabajar en el puerto 80 para que los redireccione (“/a/” y “/t/”)y evitando de esta manera las interferencias entre herramientas. Así tendremos que para acceder localmente a Apache2 sería “localhost/a/” y para acceder a Tomcat sería “localhost/t/” o desde internet (externamente) “kefren.ujaen.es:6901/a/” o “kefren.ujaen.es:6901/t/”.

A continuación describiremos los pasos uno a uno:

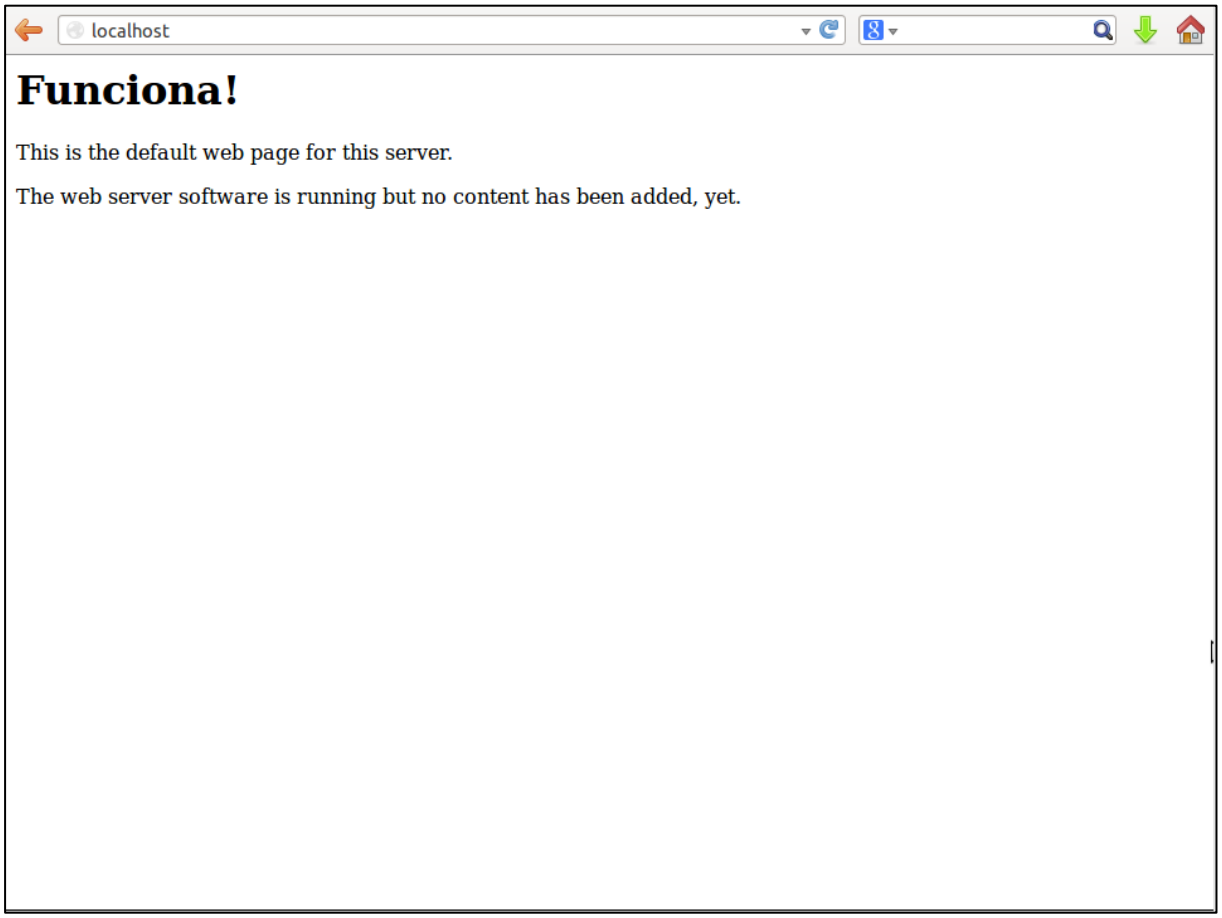
Paso 1: Instalación y configuración de Apache2

Para poder utilizar MySQL utilizaremos PhpMyAdmin y para ello necesitamos instalar Apache2 que será el que nos permita ejecutarlo en remoto.

En este caso utilizaremos como ya he mencionado anteriormente el método de intalación por consola.

```
sudo apt-get install apache2
```

Tras la instalación deberemos de introducir una contraseña para el usuario root que será el administrador y una vez esté terminado podremos comprobar que se ha instalado correctamente accediendo a “localhost” desde el navegador.



Ahora toca configurar Apache2 para que funcione por el puerto 81 en lugar del puerto 80 que es el que viene por defecto y para ello haremos las siguientes modificaciones en estos ficheros:

En “*etc/apache2/ports.conf*” cambiaremos:

- “NAmeVirtualHost *:80” a **“NameVirtuaHost *:81”**
- “Listen 80” a **“Listen 81”**

En "*/etc/apache2/sites-available/default*" cambiaremos:

- "<VirtualHost *:80>" a **"<VirtualHost *:81>"**

Paso 2: Instalación de Java

En este paso intalaremos Java para que todo funcione de manera correcta y volveremos a hacer uso de la consola de comandos.

```
sudo apt-get install python-software-properties

sudo add-apt-repository ppa:webupd8team/java

sudo apt-get update

sudo apt-get install oracle-java7-installer
```

Y para comprobar que la instañlación ha sido satisfactoria utilizaremos el siguiente comando:

```
java –version
```

Paso 3: Instalación MySQL-Server

Para esta instalación ejecutamos el siguiente comando:

```
sudo apt-get install musql-server
```

Paso 4:Instalación y configuración de PhpMyAdmin

En este paso instalaremos nuestro gestor de base de datos y crearemos todo lo necesario para que esta se quede funcionando.

Para la instalación usaremos el siguiente comando:

```
sudo apt-getinstall phpmyadmin
```

A continuación nos pedirá el lugar de instalación y le indicaremos que en Apache2.

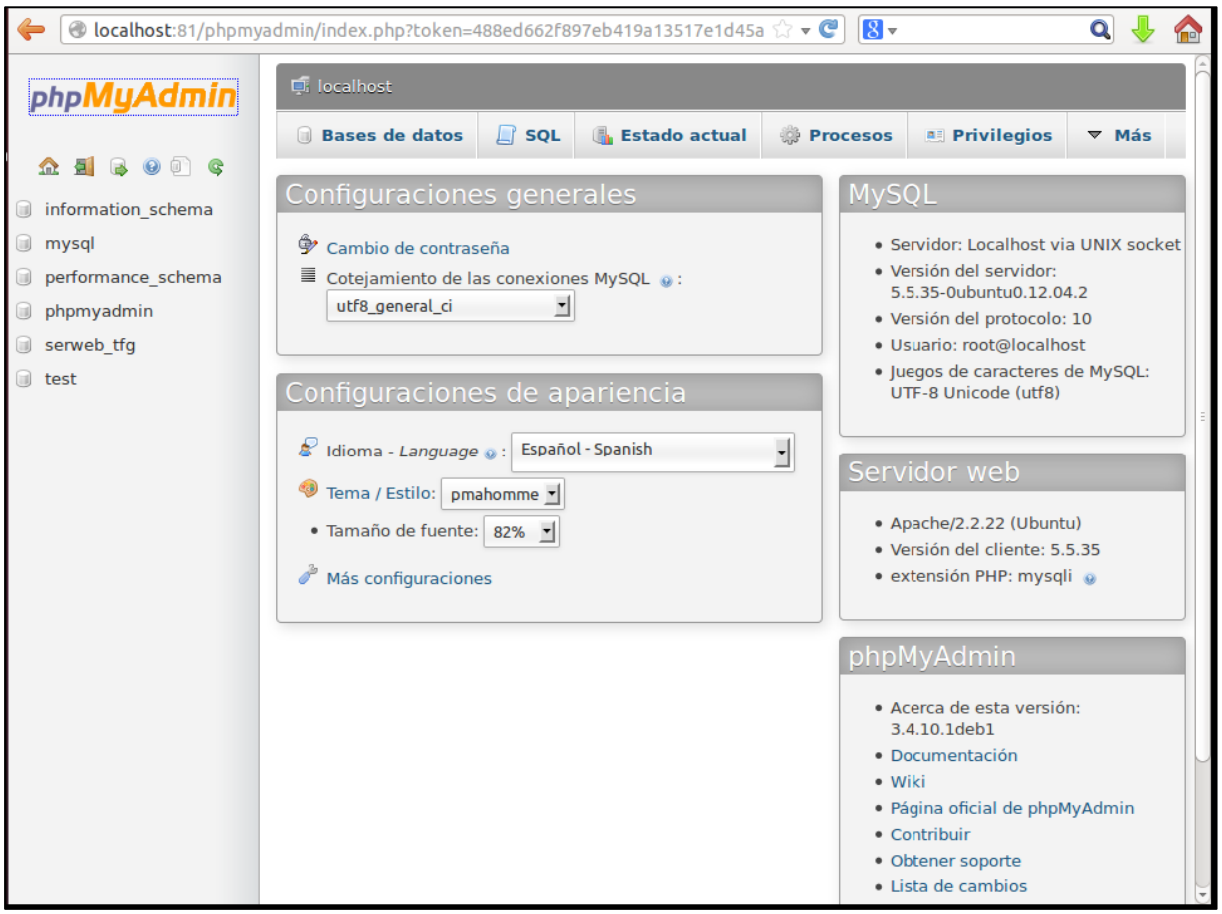
Tras la instalación ejecutaremos el siguiente comando:

```
sudo ln -s /usr/share/phpmyadmin/ /var/www/phpmyadmin
```

Y reiniciamos Apache2 para que aplique los cambios de puerto.

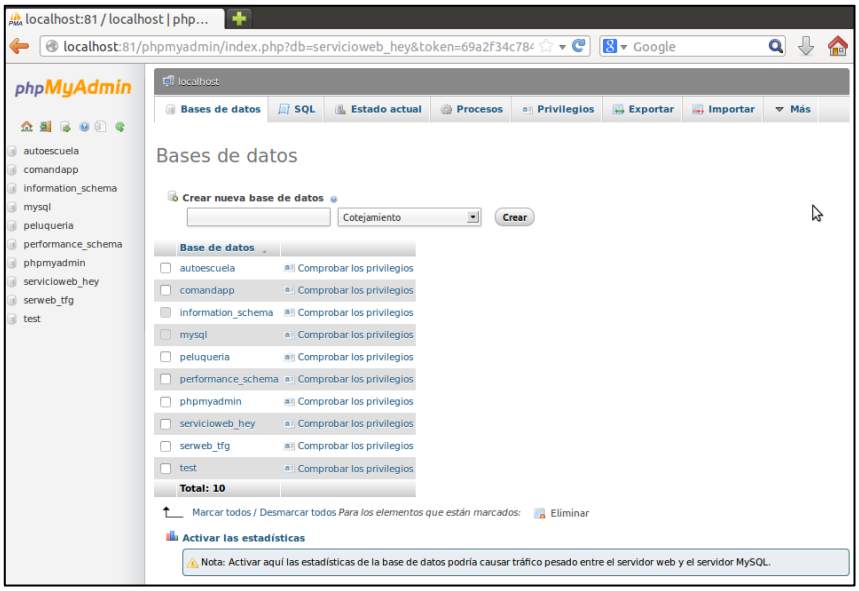
```
sudo /etc/init.d/apache2 restart
```

Para finalizar la parte de la instalación comprobaremos que es accesible por la ruta http://localhost:81/phpmyadmin/ y si es así añadiremos el usuario administrador "*root*".



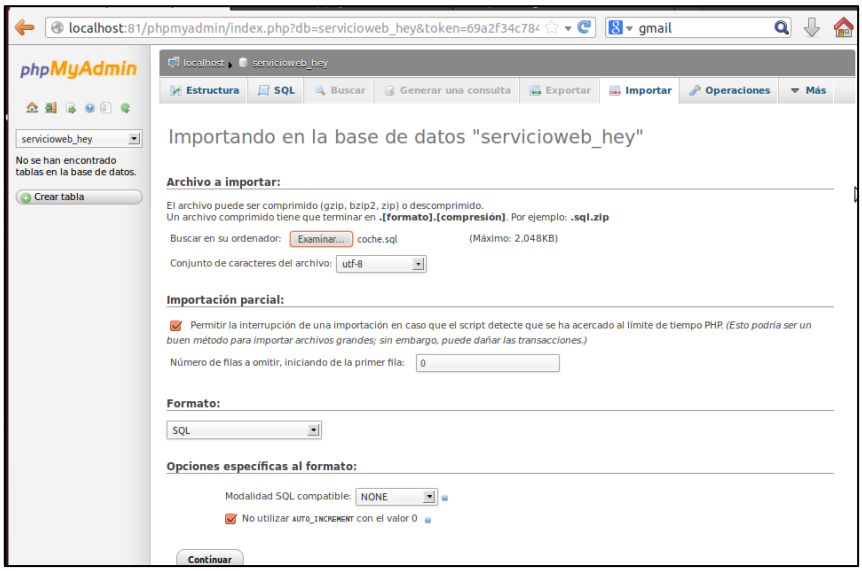
Ahora ya podemos pasar a crear nuestra base de datos a partir de los ficheros .sql proporcionados en este proyecto dentro de la carpeta "*ServicioWeb/src/es/ujaen/H/sql/*".

Para crear la base de datos nos dirigiremos a la pestaña "Base de datos"



Como nombre de nuestra base de datos tenemos que poner exactamente “servicioweb_hey” y elegir la opción “*utf8_general_ci*”.

Lo siguiente que debemos hacer tras la creación de la base de datos es importar los archivos .sql que hemos mencionado anteriormente y que se encuentran en la raíz del proyecto “*ServicioWeb/src/es/ujaen/Hey/sql/*". Para ello nos dirigimos a la pestaña de “*Importar*” y puslamos en “*examinar*”.



Para ir seleccionando los archivos de uno en uno y en el siguiente orden para evitar errores.

1. empleado.sql
2. posiciónGPS.sql
3. coche.sql
4. ruta.sql
5. listaContactos.sql

NOTA: Estos archivos ya incluyen algunos datos de muestra para poder hacer uso de la aplicación o una posible demostración.

Paso 5: Instalación y configuración de Tomcat 7

Apache Tomcat 7 es el servidor donde se va a alojar nuestro servicio web por lo que es de suma importancia. Para la instalación tan solo deberemos de ejecutar el siguiente comando:

```
sudo apt-get install tomcat7
```

Una vez instalado procederemos a su configuración como hemos hecho con el resto de herramientas realizando modificaciones en los siguientes ficheros:

En "*/etc/tomcat7/server.xml*" cambiamos:

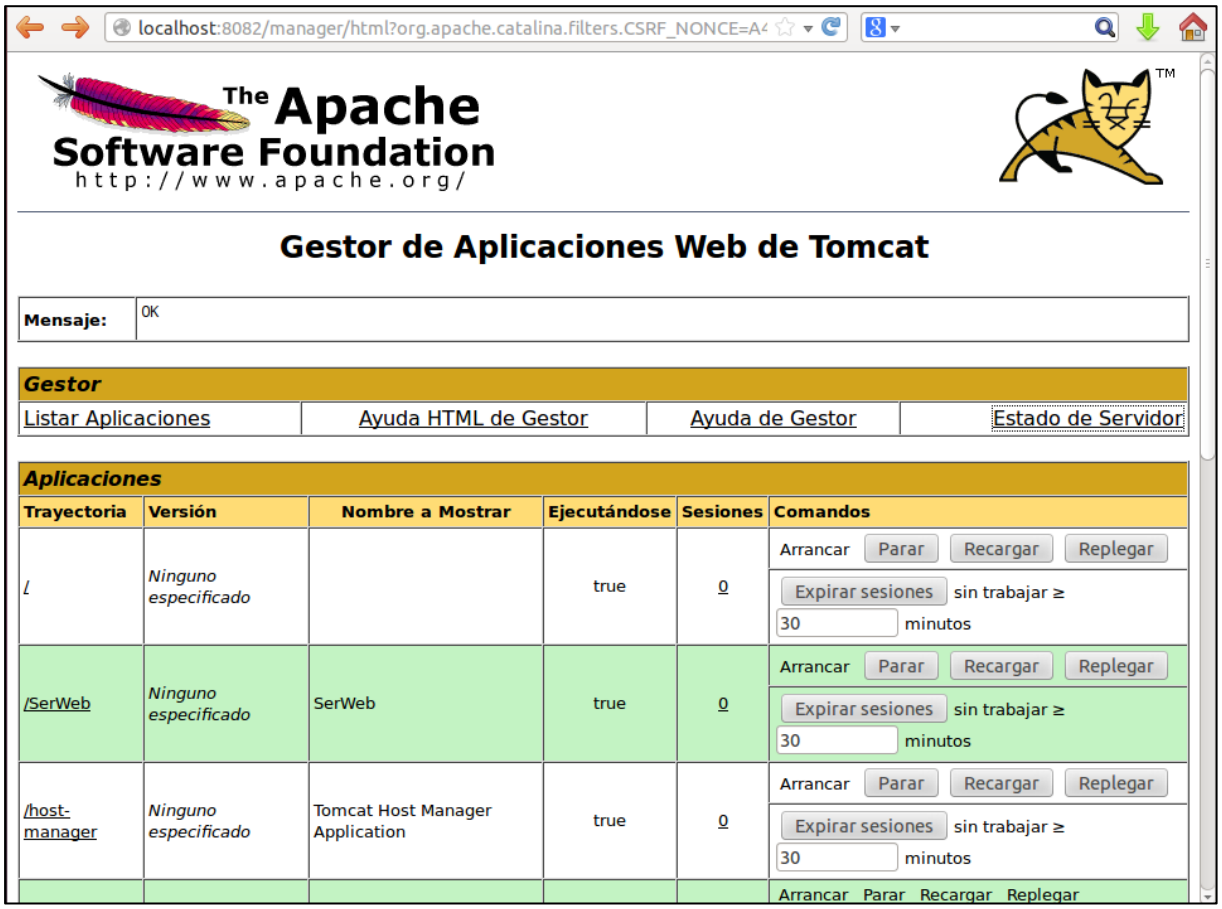
<Connector port="8080" protocol="HTTP/1.1" a **<Connector port="8082" protocol="HTTP/1.1"**

En "*/etc/tomcat7/tomcat-users.xml*" añadiremos al final der archivo las siguientes líneas (debe de añadirse antes de "</tomcat-users>"):

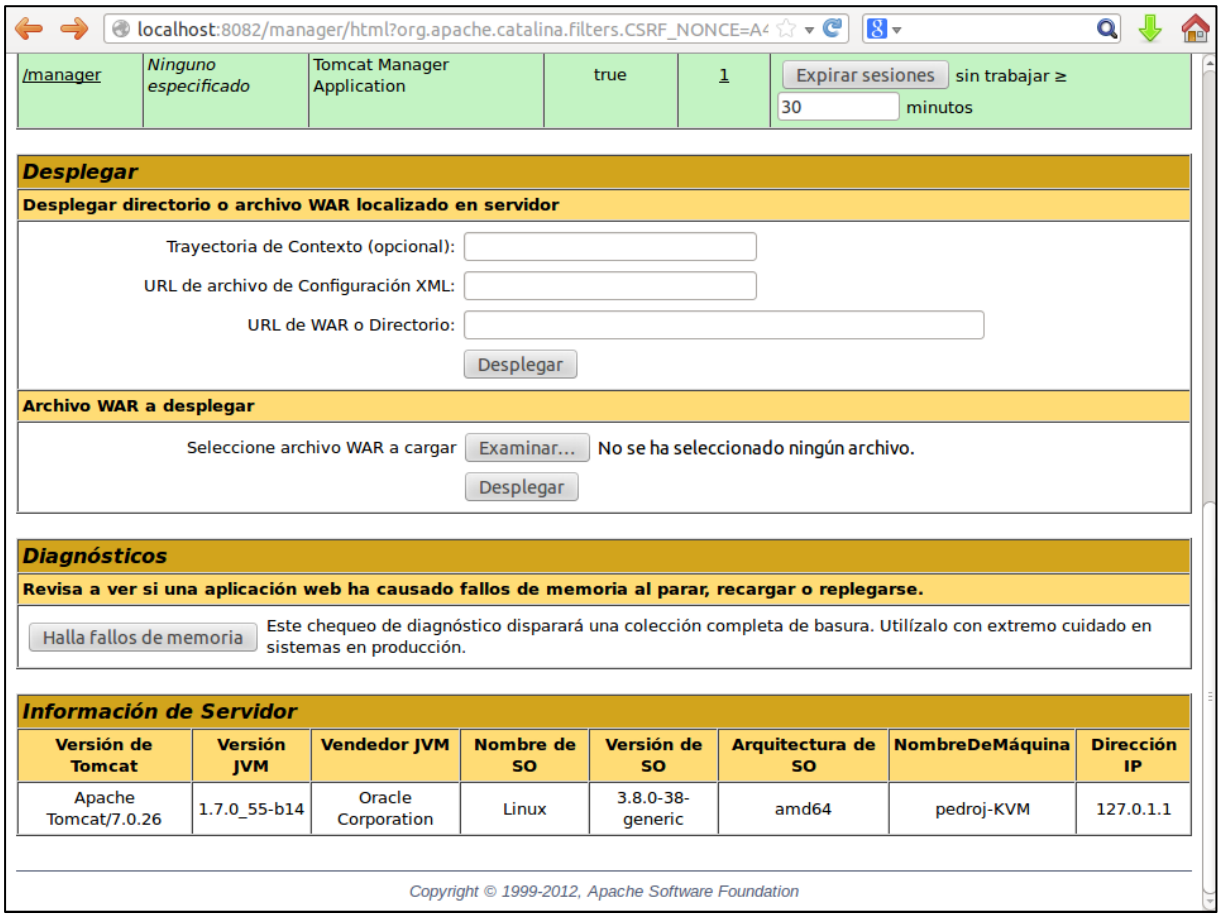
```
<role rolename="manager-gui"/>
<role rolename="admin-gui"/>
<user username="tomcat" password="tomcat"
roles="manager-gui, admin-gui"/>
```

Las dos primeras líneas para establecer el rol del usuario que se declara a continuación que en nuestro caso hemos utilizado tomcat y como contraseña de nuevo pero que podrían ser cualquiera de nuestra elección.

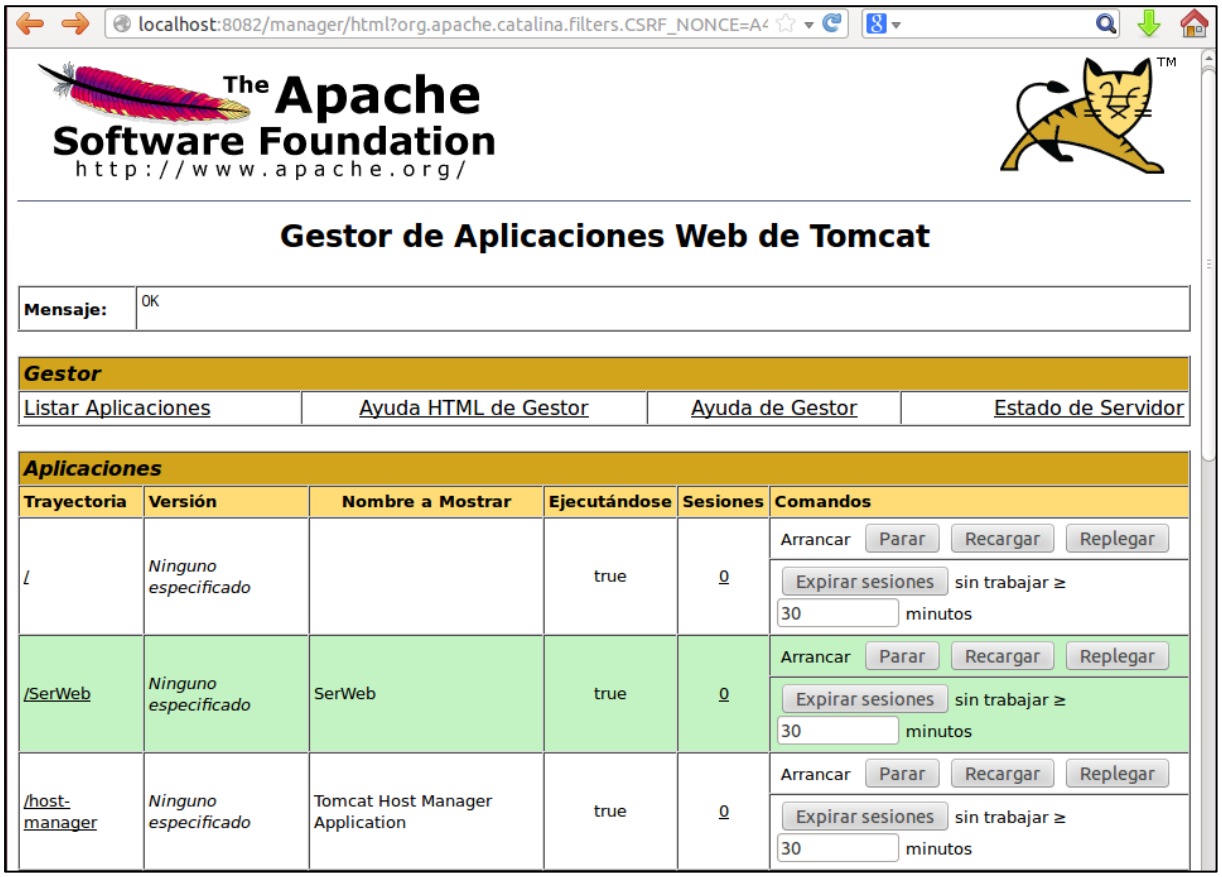
Tras todo esto ya deberíamos de ser capaces de acceder a nuestro panel de administración introduciendo "*localhost/:8082/manageer*" en el navegador he introduciendo posteriormente el usuario y contraseña que acabamos de declarar.



Una vez configurado Tomcat 7 procedemos a insertar nuestro servicio web para dejarlo operativo para ello bajamos en la pantalla que nos acaba de aparecer para visualizar las funcionalidades y pulsamos sobre el botón de “Examinar…” dentro de la sección “Archivo war a desplegar”.



Buscamo nuestro fichero “ServicioWeb.war” incluido en el proyecto y pulsamos en el botón de “Desplegar”. Si el servicio se ha desplegado correctamente nos aparecerá una pantalla con un “OK” y posteriormente la sección de Aplicaciones del servicio web “/ServicioWeb”.



Paso 6: Instalación y configuración de Nginx

Para poder tener acceso desde el exterior del servidor (internet) a estas herramientas que acabamos de instalar tendremos que mapear los puertos internamente y para ello instalaremos Nginx ejecutando el siguiente comando:

```
sudo apt-get install nginx
```

Tras su instalación tendremos que configurarlo apropiadamente siguiendo estos pasos:

1. Editamos el fichero "*/etc/nginx/nginx.conf*" cambiando:

client_max_body_size 2M; a **client_max_body_size 200M;** // La cantidad que necesites.

2. Ahora configuramos las redirecciones modificando el fichero "*/etc/nginx/sites-available/default*" agregando las siguientes líneas dentro de "server {":

```
Listen:80; // Si no está en 80, cambiarlo a 80
Client_max_body_size 50M; // Aumentarlo según se necesite
// dirección para Apache
location /a/ {
    proxy_pass http://localhost:81/;
}
    // dirección para Tomcat
    location /t/ {
            proxy_pass http://localhost:8082/;
}
```

3. Finalmente reiniciamos Nginx para que los cambios se apliquen.

```
sudo /etc/init.d/nginx restart
```

Paso 7: Instalación de MySQL Java Connector

Para finalizar con la instalación de nuestro servicio web necesitamos que la librería del driver para el acceso a la base de datos MySQL esté incorporada a la biblioteca de Tomcat 7 ya que es aquí donde alojamos el servicio.

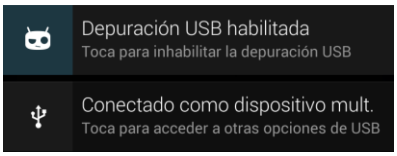
Para ello simplemente cogemos el fichero "*mysql-connector-java-5.1.31-bin.jar*" que encontraremos dentro de nuestro proyecto en la ruta "*ServicioWeb/lib/*" y lo pegamos en nuestro servidor en la ruta "*usr/share/tomcat7/lib*".

Y así finalizamos la instalación y configuración de nuestro servicio web en el servidor para poder hacer uso de él a través de nuestra aplicación Android.

ANEXO B: Manual de instalación del Cliente Android

La instalación de nuestra aplicación en dispositivo Android es bastante sencilla y consta de los siguientes pasos:

1. Conectamos por cable el dispositivo a nuestro ordenador.
2. Pulsamos sobre la notificación que nos aparece en nuestro dispositivo con icono de USB “Conectado como dispositivo multi.”.

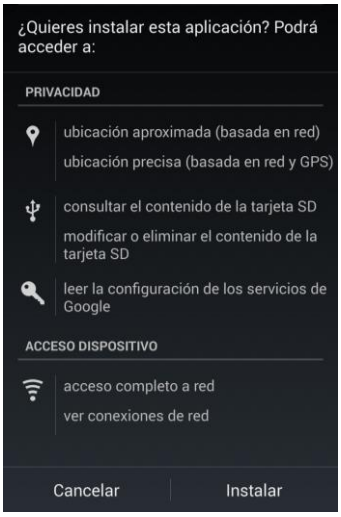


3. Seleccionamos la opción que pone “Mass storage”.

4. Ahora nos dirigimos a nuestro ordenador y veremos que nos aparece el explorador de archivos para poder explorar el contenido de nuestro dispositivo así que pasamos a copiar el fichero “Hey!.apk” que encontraremos en la raiz del proyecto a la carpeta que deseemos dentro de nuestro dispositivo.

   NOTA: Hay que mencionar que nuestro dispositivo debe de tener un navegador de ficheros y en su defecto podemos descargarnos cualquiera gratuito en pocos minutos desde la tienda de aplicaciones de Google.

5. Accedemos a nuestro navegador de archivos y nos dirigimos a la carpeta donde previamente hemos copiado el fichero y ejecutorlo. Nos aparecerá algo así:

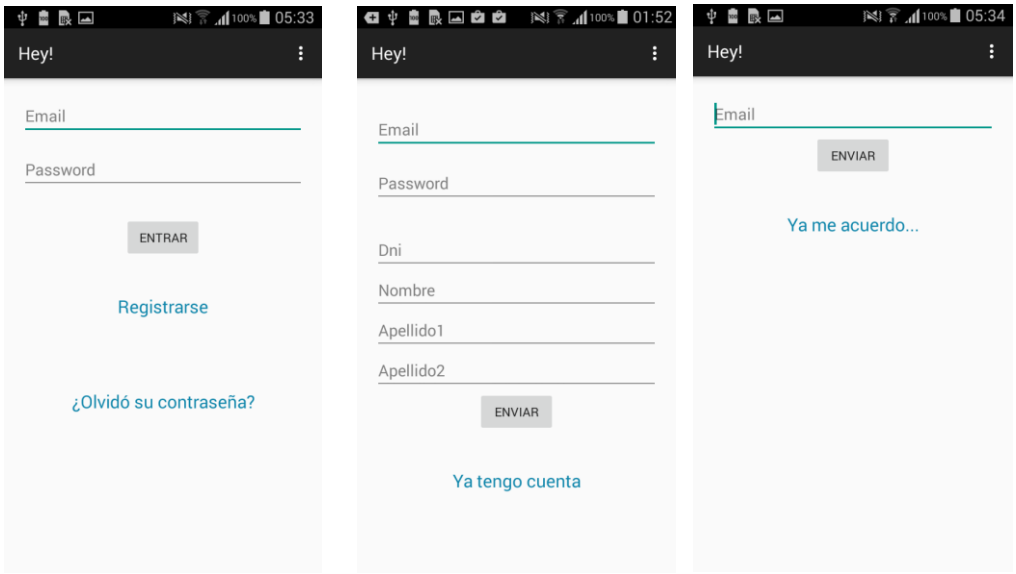


6. Como podemos ver, el dispositivo nos muestra los permisos que debemos conceder a nuestra aplicación para que esta pueda funcionar y si aceptamos conceder esos permisos.
7. Tras iniciarse la intalación nos aparecerá una pantalla indicando si queremos abrirla o finalizar simplemente todo este proceso. En caso de que pulsemos sobre “Abrir” nos abrirá la aplicación y se nos mostrará la pantalla de inicio de la misma.

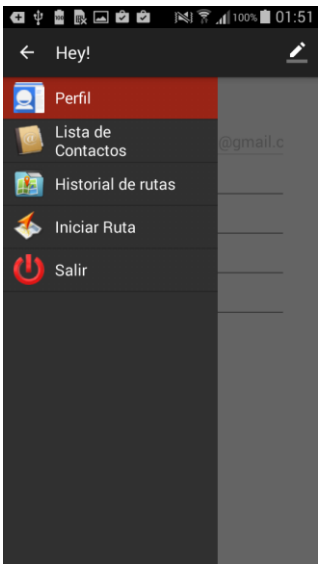
Con esto habremos terminado de instalar nuestra aplicación en cualquier dispositivo Android con una versión igual o superior a 4.0.

ANEXO C: Guía de usuario

Esta es la pantalla de inicio que el dispositivo nos muestra al abrir la plicación.



Para navegar entre las distintas pantallas de la aplicación sólo tenemos que deslizar la pantalla hacia la derecha o pulsar sobre el icono Hey! y así nos aparecerán las distintas pantallas diponibles.



Dentro de cada pantalla tenemos un menú de opciones específico en la parte superior de la pantalla y en el resto aparecerá la información correspondiene a la pantalla elegida.