Desarrollo de aplicaciones para dispositivos con Sistema Operativo Android

Trabajo Final De Carrera

Alejandro Saura Rodríguez

AGRADECIMIENTOS

Con todo mi cariño, dedico este trabajo de fin de carrera

a mi madre,
quien ha sido mi mayor inspiración para luchar en la vida
y a quien echo de menos a cada instante de cada día.
Mi promesa está cumplida. Gracias mama.

A mi familia, por estar ahí en lo bueno y en lo malo,

A mi hermano, sin el cual, esto no hubiese sido posible,

A mi pareja, Carmen, por su apoyo incondicional
y un amor que no se explica con palabras,

Al equipo docente de la Universitat Oberta de Catalunya
por estos años de continuo aprendizaje.

Índice

1. Introducción y punto de partida del TFC.

El presente documento describe el trabajo realizado en el proyecto: "Desarrollo de aplicaciones para dispositivos móviles con Sistema Operativo Android". El objeto del mismo es dar a luz una aplicación basada en el sistema operativo más utilizado mundialmente a día de hoy -según los últimos análisis de cuotas de mercado mostrados en la figura 1- para la obtención del título de Ingeniero Técnico en Informática de Gestión, expedido por la Universitat Oberta de Catalunya.

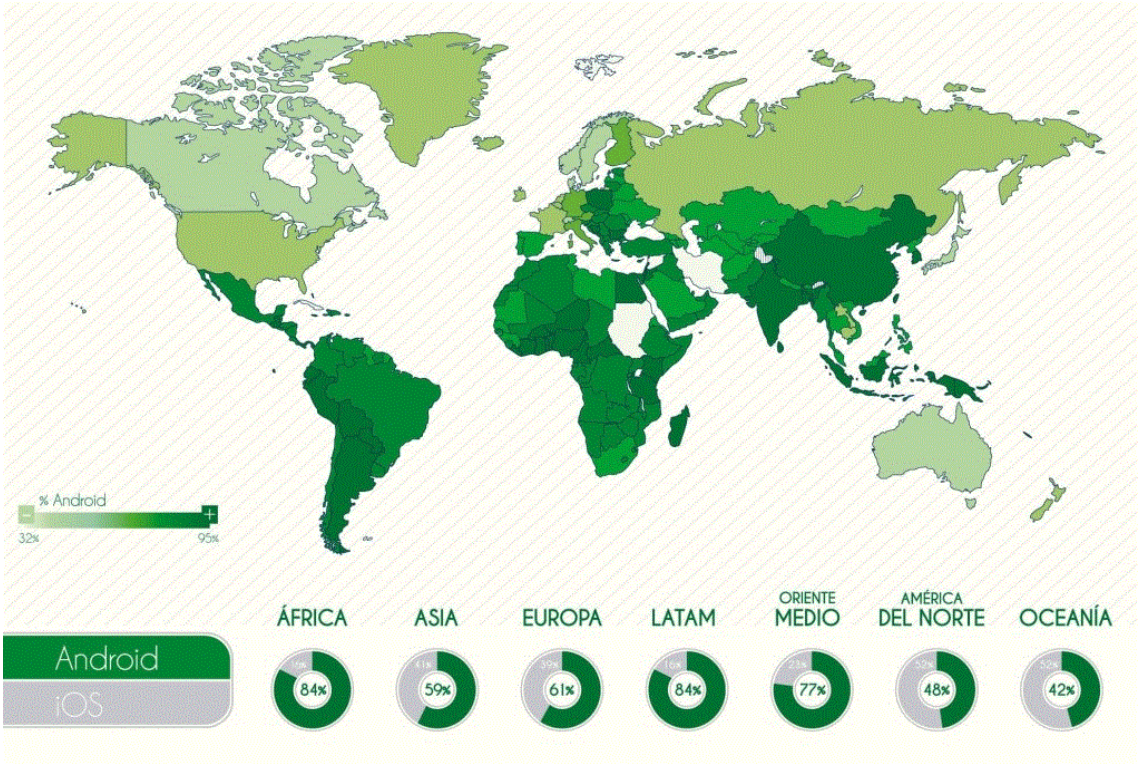


Figura 1. Dominio del Sistema Operativo Android a nivel mundial.

Me gustaría destacar que este Trabajo Final de Carrera no está basado aparentemente en ningún otro proyecto antecesor de la Universitat Oberta de Catalunya, ya que esta tecnología es tan incipiente que apenas se pueden encontrar proyectos semejantes en toda la universidad.

La idea inicial o punto de partida se basa en el trabajo que he estado desarrollando durante 18 meses para una empresa de mi localidad. Por ello, he creído necesario desarrollar una aplicación para dispositivos móviles Android que permita gestionar ordenadamente el trabajo a los empleados de un servicio técnico de reparación de equipos informáticos en la Región de Murcia. He denominado a esta aplicación *SATApp*.

Tomamos como punto de partida la creación de un único rol de usuario de la aplicación, el empleado, aunque a nivel organizativo es conveniente la existencia de un administrador de usuarios que no posee privilegios en la aplicación. Este último es quien recibe la petición de servicio o reparación por parte de un cliente y mediante el uso de la aplicación *SATApp* podrá, en primer lugar, comprobar su localización GPS y, geoposicionar al resto de empleados o técnicos a tiempo real. Además de esta importante funcionalidad, *SATApp* permite interaccionar con cada uno de los usuarios que tengan instalada esta aplicación y haya sido localizado.

Esto se pretende conseguir con la ayuda de Google Maps y a partir de una base de datos de usuarios localizada en un servidor Web (concepto que desarrollaremos en la parte de análisis más adelante).

En líneas generales, mediante *SATApp* se pretende que el propio administrador conozca la ubicación de cada uno de los empleados o técnicos e interaccione con ellos mediante llamada o correo electrónico. A su vez, le permite facilitar la asignación de reparaciones o tareas a aquel que se encuentre libre de reparaciones en ese momento o más cerca del cliente en cuestión (aunque la elección final queda siempre a cargo del administrador).

Se establece que la petición de servicio o reparación por parte del cliente anteriormente mencionada podrá efectuarse vía telefónica, por email o incluso acudiendo personalmente a la empresa para dar parte de la avería como es usual en estos casos.

Cada técnico recibirá sus reparaciones asignadas por *mail* al correo electrónico configurado anteriormente en su terminal móvil e insertado en la base de datos. En este se incluirá:

1. Nombre del cliente.
2. Dirección de la reparación.
3. Teléfono de contacto.
4. Correo electrónico (en el caso de disponer de uno).
5. Breve descripción de la incidencia.

Se toma como referencia el servidor Gmail por su gran divulgación y su extendido uso corporativo. Realmente, la aplicación permitirá usar la aplicación de correo ya instalado que el usuario desee. En este punto, se hace énfasis en que es interesante y conveniente el envío de correos electrónicos desde la propia aplicación debido a su alto nivel de usabilidad.

Además, un técnico dispondrá de tres estados para indicar el proceso en el que se encuentra durante una reparación: aceptado, en proceso y finalizado. Estudiamos los significados de cada uno de estos estados:

- **En tránsito.** Una vez que el técnico ha recibido la petición de reparación por parte del administrador del sistema, este acepta la tarea y procede a visitar al cliente para solucionar la incidencia.
- **En reparación.** El técnico se encuentra en este estado cuando ha iniciado su trabajo de reparación y está ocupado con el mismo.
- **Libre.** Este estado se muestra cuando el empleado ha terminado satisfactoriamente su reparación y queda a la espera de la siguiente.

Una vez ha finalizado el servicio, el técnico reenviará el *mail* recibido en un primer momento con los datos del cliente y añadirá al mismo los detalles de la reparación. Estos datos serán necesarios para la facturación que posteriormente realizará el administrador.

Gracias a esta manera de funcionar, se aspira a ofrecer un servicio más rápido y eficaz a empresas y particulares. Por otro lado, también es deseable para el administrador de esta aplicación conocer la posición en cada instante de sus técnicos y el tiempo que dedican a cada reparación para el buen funcionamiento de la organización.

Inicialmente, y para facilitar la gestión de la aplicación, el ámbito geográfico se ha limitado a la provincia de Murcia aunque la aplicación es funcional en cualquier ubicación del país y solo será necesario un terminal Android con tecnología 4G y sistema GPS.

1.1. Motivación del proyecto.

El punto de partida de este Trabajo de Fin de Carrera tuvo su origen a comienzos del verano de 2013. Por aquella época comenzaba a trabajar como técnico informático mientras acababa de

completar casi la totalidad de los créditos de la titulación y surgieron los primeros planteamientos acerca del tema sobre el que trataría este último trabajo de la carrera.

En un principio, la idea era llevar a cabo un proyecto innovador y útil propuesto por mí mismo y que permitiese aplicar los conocimientos adquiridos a lo largo de los muchos años de formación. Deseaba crear una aplicación que solventase las deficiencias en el proceso diario de trabajo de la empresa y así mejorar la producción y el trato al cliente.

El tema de los dispositivos móviles me agradó bastante desde el inicio y es en este campo sobre el que he decidido centrar mi proyecto. Tras coger impulso gracias a un curso de Desarrollo de Aplicaciones Android impartido por la Universidad de Valencia, incurrí en crear una aplicación para teléfonos móviles basados en este sistema y con conexión GPS, que permitiese acelerar el trabajo diario de la empresa mediante el control y comunicación entre sus empleados gracias a la tecnología 4G.

Más concretamente, espero cubrir necesidades tales como:

- Mantener la organización y comunicación dentro de una empresa orientada a la reparación de sistemas informáticos.
- Unificar todas las funcionalidades en una sola aplicación entre la empresa y el técnico de manera sencilla.
- Facilitar la usabilidad mediante aplicaciones ya conocidas como Google Maps para localizar al empleado y Gmail, si el usuario lo estima oportuno, para el intercambio de correos electrónicos (incluidas de fábrica en la gran mayoría de terminales móviles Android).
- Posibilitar que las funcionalidades pueden ser mejoradas conforme vayan surgiendo necesidades dentro de la empresa.
- Ahorrar en el innecesario uso de herramientas adicionales que incluso puedan generar un coste extra.
- Conocer la situación de los empleados de la organización a tiempo real.
- Ofrecer un diseño y funcionalidad sencillos para que el usuario de la aplicación disponga de conocimientos técnicos suficientes en el uso de la aplicación y no sea necesaria formación adicional.
- Permitir un servicio rápido y eficaz al cliente, lo que mejora la imagen de la empresa.

2. Estado del arte.

Hoy en día, una gran cantidad de empresas del ámbito informático realizan costes muy elevados, tanto en formación como en herramientas, para optimizar su trabajo. En este apartado, pretendo mostrar los recursos utilizados, el conocimiento adquirido y el resultado de la investigación realizada para el desarrollo de la aplicación *SATApp*. Por supuesto, la finalidad siempre ha sido beneficiar al entorno de trabajo de una pyme.

2.1. Metodología y herramientas.

Android es un sistema operativo basado en el núcleo de Linux, más concretamente en Linux 2.6, y enfocado para ser utilizado en dispositivos móviles como *SmartPhones*, *Tablets,* televisores y otros dispositivos. Fue desarrollado por Android Inc. que posteriormente fue comprada por Google convirtiéndose en la primera incursión seria de la compañía en el mercado móvil.

Actualmente, el proyecto Android está encabezado por Google y un conjunto de empresas tecnológicas agrupadas bajo el nombre de *Open Handset Alliance* (OHA). El objetivo principal de esta alianza empresarial (que incluye a fabricantes de dispositivos y operadores, con firmas tan relevantes como Samsung, LG, Telefónica, Intel o Texas Instruments, entre otras muchas) es el desarrollo de estándares abiertos para la telefonía móvil como medida para incentivar su desarrollo y para mejorar la experiencia del usuario. La plataforma Android constituye su primera contribución en este sentido.

Tal y como queda explicado en Wikipedia, "la estructura del sistema operativo Android se compone de aplicaciones que se ejecutan en un *framework* Java de aplicaciones orientadas a objetos sobre el núcleo de las bibliotecas de Java en una Máquina Virtual Dalvik con compilación en tiempo de ejecución. Las bibliotecas escritas en lenguaje C incluyen un administrador de interfaz gráfica, un *framework OpenCore*, una base de datos relacional *SQLite*, una Interfaz de programación de API gráfica OpenGL ES 2.0 3D, un motor de renderizado *WebKit*, un motor gráfico SGL, SSL y una biblioteca estándar de C Bionic" (Android. n.d. En *Wikipedia*. Consultado en Marzo 10, 2015).

La licencia de distribución elegida para Android fue Apache 2.0, lo que lo convierte en software de libre distribución y código abierto. Es más, a los desarrolladores se les proporciona de forma gratuita un SDK y la opción de un *plug-in* para el entorno de desarrollo.

Para el buen desarrollo de *SATApp* sobre Android se hará uso de la última versión de Android Studio, siendo este el IDE oficial para comenzar desarrollar sobre esta plataforma. Dicho desarrollo comenzará desde cero puesto que está basada en una idea concreta y por ello no hay aplicación con las mismas funcionalidades en el mercado. Así mismo, he elegido un proyecto de creación que me motiva y de utilidad en mi sector profesional.

A continuación, detallaré brevemente aspectos relevantes a las herramientas y tecnologías que se usarán durante la vida del presente proyecto aunque no ahondaré demasiado en la historia de las mismas dado que ese no es el objetivo buscado.

Java

Java es un lenguaje orientado a objetos que alcanzó su madurez con la popularización de Internet y que es considerado, en cierta manera, el heredero legítimo del lenguaje C++. Este es el lenguaje en el cual está basado el proyecto *SATApp*.

Java es un lenguaje neutral, portable, robusto, estable, independiente de la plataforma y relativamente sencillo de aprender para programadores con experiencia previa en lenguajes orientados a objetos. Puede utilizarse para realizar aplicaciones en múltiples plataformas hardware y sistemas operativos (Unix, Linux, Windows o HP-UX entre otros sistemas operativos para ordenadores personales y Android, Palm OS o EPOC entre otros sistemas enfocados para dispositivos de telefonía móvil).

Los requisitos de desarrollo para Android exigen la previa instalación del JDK (*Java Development Kit*) en su versión 8.

Siguiendo las recomendaciones de Google, el proyecto usará el lenguaje de desarrollo Java.

Android Studio

Android Studio es el entorno de desarrollo integrado (IDE) oficial de Android y se ha convertido en el recomendado en la actualidad por la mayoría de desarrolladores que anteriormente

usaban Eclipse. Android Studio fue anunciado por Ellie Powers el 16 de mayo de 2013, está disponible para desarrolladores de manera gratuita y ha sido basado en *IntelliJ IDEA* de JetBrains.

Es una realidad que en comparación con Eclipse, Android Studio es más serio, más versátil, más potente y más parecido a un proyecto en java, y es que Android Studio utiliza *Gradle* y trae consigo las siguientes ventajas:

- Facilita la reutilización de código y recursos.
- Facilita la distribución de código.
- Permite la restructuración de código y soluciones más rápidas.
- Facilita configurar, extender y personalizar el proceso.
- Dispone de herramientas específicas, mejor rendimiento, usabilidad y compatibilidad.
- Posee una función para firmar aplicaciones.
- Permite compilar desde la línea de comandos.
- Aporta mejoras en el editor gráfico.
- Tiene soporte para Google Cloud Platform.
- Utiliza el compilador Graddle como herramienta que automatiza la construcción de las aplicaciones y que tiene montadas las tareas para la mayoría de los proyectos por defecto. Dispone de más y mejores plantillas base para crear aplicaciones profesionales.
- Hace increíblemente fácil crear distintas versiones de la aplicación.

En definitiva, Android Studio es una herramienta con un potencial muy alto especialmente de cara a entornos empresariales.

Componentes de una aplicación Android

Todas las aplicaciones en Android pueden descomponerse en cuatro tipos de bloques o componentes principales. Cada aplicación será una combinación de uno o más de estos componentes, que deberán ser declarados de forma explícita en un fichero con formato XML denominado *AndroidManifest.xml*, junto a otros datos asociados como valores globales, clases que implementa, datos que puede manejar, permisos, etc. Este fichero es básico en cualquier aplicación en Android y permite al sistema desplegar y ejecutar correctamente la aplicación.

A continuación, se exponen brevemente los cuatro tipos de componentes en los que puede dividirse una aplicación para Android:

- ***Activity.*** Sin duda, es el componente más habitual de las aplicaciones para Android. Un componente *Activity* refleja una determinada actividad llevada a cabo por una aplicación y lleva asociada una ventana o interfaz de usuario. Es importante entender que no contempla únicamente el aspecto gráfico, sino que éste forma parte del componente *Activity* a través de vistas representadas por clases como *View* y sus derivadas. Este componente se implementa mediante la clase de mismo nombre *Activity*.

  Muy vinculado a este componente se encuentran los *Intents*, una interesante novedad introducida por Android. Un *Intent* consiste básicamente en la voluntad de realizar alguna acción, generalmente asociada a unos datos.

- ***Broadcast Intent Receiver.*** Un componente *Broadcast Intent Receiver* se utiliza para lanzar alguna ejecución dentro de la aplicación actual cuando un determinado evento se produzca (generalmente, abrir un componente Activity). No tiene interfaz de usuario

asociada pero puede utilizar el *API Notification Manager* para avisar al usuario del evento producido a través de la barra de estado del dispositivo móvil.

- ***Service***. Un componente *Service* representa una aplicación ejecutada sin interfaz de usuario, y que generalmente tiene lugar en segundo plano mientras otras aplicaciones (éstas con interfaz) son las que están activas en la pantalla del dispositivo.

- ***Content Provider***. Con este componente, cualquier aplicación en Android puede almacenar datos en un fichero, en una base de datos SQLite o en cualquier otro formato que considere. Además, estos datos pueden ser compartidos entre distintas aplicaciones. Una clase que implemente el componente *Content Provider* contendrá una serie de métodos que permite almacenar, recuperar, actualizar y compartir los datos de una aplicación.

Ciclo de vida de las aplicaciones Android

En Android, cada aplicación se ejecuta en su propio proceso. Esto aporta beneficios en cuestiones básicas como seguridad, gestión de memoria, o la ocupación de la CPU del dispositivo móvil. Android se ocupa de lanzar y parar todos estos procesos, gestionar su ejecución y decidir qué hacer en función de los recursos disponibles y de las órdenes dadas por el usuario.

El usuario desconoce este comportamiento de Android. Simplemente es consciente de que mediante un simple clic pasa de una a otra aplicación y puede volver a cualquiera de ellas en el momento que lo desee. No debe preocuparse sobre cuál es la aplicación que realmente está activa, cuánta memoria está consumiendo, ni si existen o no recursos suficientes para abrir una aplicación adicional. Todo eso son tareas propias del sistema operativo.

Cada uno de los componentes básicos de Android tiene un ciclo de vida bien definido. Esto implica que el desarrollador puede controlar en cada momento en qué estado se encuentra dicho componente, pudiendo así programar las acciones que mejor convengan. El componente *Activity*, es probablemente el más importante y tiene un ciclo de vida como el mostrado en la figura 2.

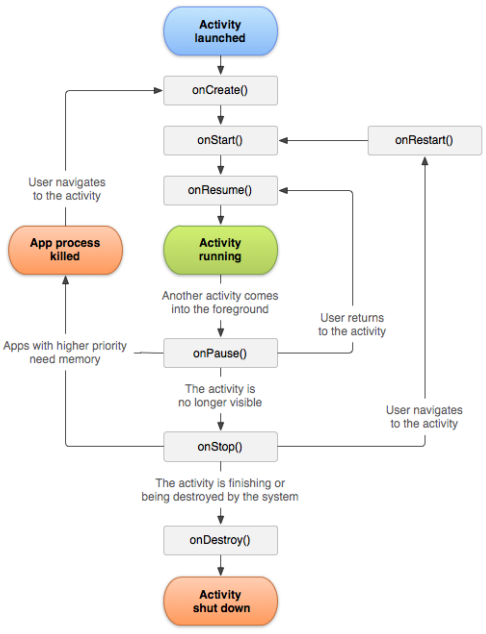


Figura 2. Ciclo de vida de una actividad en Android.

De la figura 2 se pueden obtener las siguientes conclusiones:

- *onCreate()*, *onDestroy()*: abarcan todo el ciclo de vida. Cada uno de estos métodos representan el principio y el fin de la actividad.
- *onStart()*, *onStop()*: representan la parte visible del ciclo de vida. Desde *onStart()* hasta *onStop()*, la actividad será visible para el usuario, aunque es posible que no tenga el foco de acción por existir otras actividades superpuestas con las que el usuario está interactuando. Pueden ser llamados múltiples veces.
- *onResume()*, *onPause()*: delimitan la parte útil del ciclo de vida. Desde *onResume()* hasta *onPause()*, la actividad no sólo es visible, sino que además tiene el foco de la acción y el usuario puede interactuar con ella.

Seguridad en Android

En Android, cada aplicación se ejecuta en su propio proceso. La mayoría de las medidas de seguridad entre el sistema y las aplicaciones deriva de los estándares de Linux 2.6, cuyo *kernel*, recuérdese, constituye el núcleo principal de Android.

Cada proceso en Android constituye lo que se llama un cajón de arena o *sandbox*, que proporciona un entorno seguro de ejecución. Por defecto, ninguna aplicación tiene permiso para realizar ninguna operación o comportamiento que pueda impactar negativamente en la ejecución de otras aplicaciones o del sistema mismo. Acciones como leer o escribir ficheros privados del usuario (contactos, teléfonos, etc.), leer o escribir ficheros de otras aplicaciones, acceso de red, habilitación de algún recurso hardware del dispositivo, etc., no están permitidas. La única forma de poder saltar estas restricciones impuestas por Android, es mediante la declaración explícita de un permiso que autorice a llevar a cabo una determinada acción habitualmente prohibida.

Además, en Android es obligatorio que cada aplicación esté firmada digitalmente mediante un certificado, cuya clave privada sea la del desarrollador de dicha aplicación. No es necesario vincular a una autoridad de certificado, el único cometido del certificado es crear una relación de confianza entre las aplicaciones. Mediante la firma, la aplicación lleva adjunta su autoría.

Para establecer un permiso para una aplicación, es necesario declarar en el manifiesto uno o más elementos *<uses-permission>* donde se especifica el tipo de permiso que se desea habilitar. Un ejemplo de esto mismo queda plasmado en la figura 3:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="satapp.tfc.proyecto.satapp" >

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

    <uses-library android:name="com.google.android.maps" />

    <com.google.android.maps.MapView
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:id="@+id/mapview"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:apiKey="AIzaSyATC09reX7U_ePcPwtWdVcukRfeEOflAN0"
        android:clickable="true" />

    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="com.google.android.providers.gsf.permission.READ_GSERVICES" />
```

Figura 3. Ejemplo de permisos incluidos en el manifiesto de Android.

Emulador de Android

El SDK de Android incluye un completo emulador que permite probar y depurar de manera eficiente las aplicaciones que se van desarrollando. Dicho emulador incluye diferentes *skins* para representar gráficamente un dispositivo móvil real, con pantalla, teclado y botones de función, así como aplicaciones de usuario preinstaladas como el navegador Web.

El emulador es muy completo y a continuación se mencionarán brevemente algunas de las principales características de esta útil herramienta para el desarrollo en Android:

- Puede ser lanzado tanto a través de una ventana de comandos mediante el ejecutable "*emulator.exe*" situado en la carpeta "*\tools*" del directorio de instalación. En este proyecto se hará automáticamente a través del emulador incluido en el SDK aunque existe la posibilidad de descargar emuladores con más funcionales como por ejemplo Genymotion desde el link https://www.genymotion.com/#!/.
- Una vez lanzado el emulador, que puede tardar unos minutos en cargarse, puede interactuarse con éste a través de su interfaz gráfica pulsando en la pantalla del dispositivo, sobre sus botones de función como si de un terminal corriente se tratase.

Figura 4. Emulador de Nexus S incluido en SDK de Android Studio.

Servidor Web

Las conexiones realizadas desde la aplicación *SATApp* hacia el servidor Web siguen un estándar internacional llamado *HiperText Transfer Protocol* o HTTP. Este protocolo consiste en reglas sencillas de transferencia de recursos o archivos entre equipos interconectados a una red.

La comunicación se establece a través de una petición de envío, la cual contiene los datos del cliente, como en este caso, su nombre, correo electrónico, nombre de usuario, contraseña de acceso, teléfono, latitud, longitud y fecha.

Una petición puede tener múltiples objetivos dependiendo del método que se elija. Los tipos de peticiones más comunes son el Retorno de datos y la Publicación de datos. Técnicamente se les conoce como los métodos GET y POST.

El ejemplo más popular del método POST se refleja en el envío de información desde un formulario hacia la base de datos del servidor y este es el usado por la aplicación *SATApp*.

El servidor al que se conecta *SATApp* realiza dos funciones principales: actualizar la información de localización del usuario y enviar la información de localización del usuario o técnico solicitado. Mediante la actualización, un técnico envía sus datos (latitud, longitud,

nombre, correo electrónico y número de teléfono) y estos son almacenados en una base de datos. En la consulta, el servidor lee los contactos solicitados y devuelve la información de todos aquellos que estén presentes en esta misma base de datos.

En la composición del servidor funcionan tres componentes básicos:

- Una base de datos MySQL, donde se almacena toda la información de localización que envían los usuarios de la aplicación.
- Un Servidor Web, que atiende cada petición recibida, la procesa y envía la respuesta correspondiente.

Base de datos MySQL

MySQL es un sistema de gestión de bases de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones hoy en día. MySQL AB –desde enero de 2008 una subsidiaria de Sun Microsystems y ésta, a su vez, de Oracle Corporation desde abril de 2009– desarrolla MySQL como software libre en un esquema de licenciamiento dual.

Por un lado se ofrece bajo la GNU GPL para cualquier uso compatible con esta licencia, pero para aquellas empresas que quieran incorporarlo en productos privativos deben comprar a la empresa una licencia específica que les permita este uso. Está desarrollado en su mayor parte en ANSI C.

MySQL es usado por muchos sitios web grandes y populares, como Wikipedia, Google, Facebook, Twitter, Flickr y YouTube.

Para la base de datos de *SATApp* se utilizará el gestor MySQL, usado bajo la licencia gratuita. En él se implementará una sencilla base de datos llamada *satapp* que cuenta con una única tabla en primera instancia denominada *users*. En ella se almacena para cada usuario, su nombre de usuario, contraseña, el número de teléfono que lo identifica, su nombre, un correo electrónico, la fecha, el estado de reparación, su latitud y longitud. Toda la información concreta de este sistema será estudiada en profundidad y quedará detallada en el área de diseño más adelante.

JSON

JSON, acrónimo de JavaScript Object Notation, es un formato ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML.

La simplicidad de JSON ha dado lugar a la generalización de su uso, especialmente como alternativa a XML en AJAX. Una de las supuestas ventajas de JSON sobre XML como formato de intercambio de datos en este contexto es que es mucho más sencillo escribir un analizador sintáctico (parser) de JSON.

Si bien es frecuente ver JSON posicionado contra XML, también es frecuente el uso de JSON y XML en la misma aplicación. Como es el caso, una aplicación de cliente que integra datos de Google Maps con datos de control de *login* de usuario en PHP hacen necesario soportar ambos formatos.

Google Maps API Key V2

En diciembre de 2012, Google presentaba la segunda versión de su API de Google Maps para Android. Esta nueva versión presentaba muchas novedades interesantes, de las que cabe destacar las siguientes:

- Integración con los Servicios de Google Play (*Google Play Services*) y la Consola de APIs.
- Utilización a través de un nuevo tipo específico de *fragment* (*MapFragment*), una mejora muy esperada por muchos.
- Utilización de mapas vectoriales, lo que repercute en una mayor velocidad de carga y una mayor eficiencia en cuanto a uso de ancho de banda.
- Mejoras en el sistema de caché, lo que reducirá en gran medida las famosas áreas en blanco que tardan en cargar.
- Los mapas son ahora 3D, es decir, podremos mover nuestro punto de vista de forma que lo veamos en perspectiva.

Al margen de las novedades generales, como desarrolladores nos vamos a encontrar con una serie de características importantes respecto a la API anterior a la hora de desarrollar una aplicación.

La principal de estas características es el componente utilizado para la inclusión de mapas en una aplicación. Con la anterior versión de la API, para incluir un mapa en la aplicación había que utilizar un control de tipo *MapView*, que además requería que su actividad contenedora fuera del tipo *MapActivity*. Gracias a la nueva API, nos olvidamos de estos dos componentes para centrarnos en uno solo, un nuevo tipo específico de *fragment* llamado *MapFragment*. Esto permitirá entre otras cosas añadir uno o varios mapas a cualquier actividad, sea del tipo que sea, y contando por supuesto con todas las ventajas del uso de *fragments*.

Además de esta novedad, la integración de la API con los *Google Play Services* y la Consola de APIs de Google, harán que los preparativos del entorno, las librerías utilizadas, y el proceso de obtención de la API Key de Google Maps sean distintos y más sencillos que con la versión anterior.

Investigación previa de aplicaciones similares

Tras una búsqueda exhaustiva, se deja constancia de la existencia de aplicaciones similares con el fin de poder diferenciar funcionalidades respecto a este proyecto:

a) **Trello.**

Trello es un organizador de proyectos o tareas gratuito realmente versátil que he tenido la suerte que utilizar un largo tiempo aunque no está enfocado a labores técnicas.

Puntos fuertes:
- Permite el control de un proceso y que todos los participantes conozcan la tarea del otro.
- Soporta imágenes, videos, listas y comentarios, lo que lo hace muy útil para una buena comunicación con grupos de trabajo.

Puntos débiles:
- Su uso hace necesaria una metodología de trabajo compleja.
- Aún no está traducida al español.

- No permite la geoposición de los miembros del grupo de trabajo.

Información facilitada de: https://trello.com/.

b) ClaveiSAT.

Esta App con demo gratuita es menos conocida pero intuitiva y rápida a la hora de registrar trabajo de campo para un profesional de la informática.

Puntos fuertes:
- Dispone de agenda de trabajo y gestor de tareas.
- Permite el uso de gestión de artículos y material.
- Es posible grabar el kilometraje.
- Es multiplataforma.
- Permite la geolocalización GPS.

Puntos débiles:
- Los requisitos mínimos para usar tanto en un terminal como en un equipo son demasiado altos.
- Existen muchas opiniones en contra de su instalación y funcionamiento debido a errores.

Información facilitada de: http://www.clavei.es/.

c) SAT Móvil.

Es una aplicación de pago por uso muy profesional para la gestión de la movilidad de los técnicos.

Puntos fuertes:
- Permite gestionar los avisos emitidos desde una central.
- Gestión de contratos de mantenimiento.
- Gestión de garantías.
- Posibilidad de alertas programadas.
- Dispone de manuales de uso.

Puntos débiles:
- Permite geoposicionar a los técnicos pero no en tiempo real.
- Poco conocida y difícil de encontrar en Play Store. Hay una app con el mismo nombre enfocada a gestión tributaria.
- Licencias de pago por uso con suscripción necesaria.

Información facilitada de: http://www.softwaregestionsat.com/SATMovil.aspx.

d) Asana.

Asana es una aplicación gratuita para grupos de menos de quince miembros y diseñada para la colaboración y comunicación de un equipo de trabajo.

Puntos fuertes:
- Permite la conexión con otras herramientas útiles.
- Opción de conversación con el resto de usuarios.

Puntos débiles:
- Es lenta y poco funcional.
- Solo está en inglés.
- Poco intuitiva.

- Hay opciones mejores y más completas.

Información facilitada de: https://asana.com.

e) **GD Systems – Soporte Técnico.**

Esta novedosa App ha sido diseñada por la misma empresa para atender las incidencias técnicas de sus clientes.

Puntos fuertes:
- Clara y concisa.
- Permite recepción de imagen de averia.
- Incluye manuales de teléfonos con VoIP.
- Es totalmente gratuita.

Puntos débiles:
- Tan solo tiene 10 descargas en Play Store, lo que no crea confianza.
- Hay opciones más completas.

Información facilitada de: http://gdsystems.es/.

Resumen de ventajas proporcionadas por *SATApp* frente a sus competidores

Para obtener una visión más completa de las particularidades que ofrece SATApp frente al anterior listado de aplicaciones, se muestran las ventajas características sobre la competencia:

- Localización GPS propia y del resto de empleados a tiempo real. Esta funcionalidad optimiza el trabajo de la empresa y proporciona mayores resultados. Ninguna de las aplicaciones proporciona este servicio de manera gratuita. Tan solo SAT Móvil permite geoposicionar a sus empleados (no en tiempo real) y con un coste de licencia de pago por suscripción elevado).
- Usabilidad. La interfaz de *SATApp*, la rapidez de respuesta, su color característico y su fácil localización de funcionalidades le da ventaja respecto al resto de aplicaciones que, por lo general, son más lentas, con interfaces más complejas y poco intuitivas.
- Estado de reparación de un usuario o empleado. Esta peculiaridad ofrece una visión global de las actividades que realiza cada usuario y permite comprobar la fecha de última actualización para estudiar el tiempo medio empleado en cada reparación. Ninguna aplicación estudiada dispone de este servicio.

3. Objetivos.

El objetivo principal de este proyecto es crear desde cero la aplicación SATApp descrita en el punto anterior y que la misma sea funcional. Para lo consecución de este objetivo es imprescindible fijar unos objetivos más concretos que abarquen las actividades que se pretenden realizar y, además, permitan conocer el grado de desarrollo y cumplimiento alcanzado. Estos objetivos podrían ser los siguientes:

- **Conocer las principales características que ofrece el sistema Android.** Es necesario indagar toda la información posible sobre él, a fin de conocer cuál es su arquitectura, sus componentes básicos y su comportamiento al ejecutar las aplicaciones
- **Estudiar el entorno de desarrollo.** El SDK completo está disponible para cualquier desarrollador que desee descargarlo. Este incluye numerosas ayudas para comenzar a crear aplicaciones en Android, desde las API completas con todas las clases y

paquetes, hasta herramientas de programación y un completo emulador para poder realizar pruebas.

- **Investigar acerca de la tecnología a emplear.** Será necesario utilizar un servidor y una base de datos. Los detalles se reflejarán en escritos posteriores junto con su implementación.
- **Gestionar eficientemente el proyecto en sus diferentes fases.** Desde la especificación hasta el test de calidad, todos los pasos deben ser estudiados para no incurrir en errores humanos o de implementación.

A nivel personal el objetivo es, mediante los conocimientos adquiridos previamente en mis estudios en la UOC, ser capaz de gestionar correctamente el proyecto, mejorar mis conocimientos en el desarrollo de aplicaciones para plataforma Android y finalmente superar la asignatura.

Prerrequisitos

Para el desarrollo de la aplicación va a ser posible utilizar un potente y moderno entorno de desarrollo. Al igual que Android, todas las herramientas están basadas en *software* libre. Aunque existen varias alternativas para desarrollar aplicaciones en esta plataforma, en este texto se supondrá que se trabaja con la herramienta de desarrollo Android SDK y el IDE aconsejado por la mayoría de desarrolladores Android Studio. Los detalles de instalación para su buen funcionamiento quedan enumerados a continuación:

1. Instalación de la Máquina Virtual Java (disponible en http://java.com/es /download/).
2. Descarga del entorno de desarrollo Android Studio para el sistema operativo Windows (descargable en https://developer.android.com/sdk/index.html).
3. Instalación de Android Studio. Durante la instalación, es necesario indicar tanto la ruta donde se instala el Android Studio como el SDK.
4. Actualización de Android Studio. Tras finalizar el asistente de instalación, aparece la pantalla de bienvenida donde es conveniente comprobar si existe alguna actualización. De ser así, el IDE se reiniciará y volverá a aparecer la pantalla de inicio.
5. Instalación/actualización de componentes del SDK de Android. En este paso, se debe actualizar algunos componentes del SDK de Android e instalar otros adicionales que son absolutamente necesarios para el desarrollo de la aplicación. Los componentes principales que, como mínimo, deben instalarse/actualizarse son los siguientes:

    - Android SDK Tools.
    - Android SDK Platform-tools.
    - Android SDK Build-tools (por ahora la versión más reciente).
    - Una o más versiones de la plataforma Android.
    - Android Support Repository (extras).
    - Google Play Services (extras).
    - Google Repository (extras).
    - Google USB Driver (extras).

    Instalar al menos una de las versiones de la plataforma es uno de los conceptos más importantes ya que contiene los componentes y librerías necesarias para desarrollar la aplicación sobre cada una de las versiones concretas de Android. Así, al probar la aplicación sobre Android 2.2 y 4.4, por ejemplo, es absolutamente necesario descargar sus dos plataformas correspondientes. Se aconseja instalar al menos dos plataformas: la correspondiente a la última versión disponible de Android y la mínima versión de Android que se desea que soporte la aplicación a desarrollar. Esto permite probar las aplicaciones sobre ambas versiones para asegurar correcto funcionamiento. En este

caso, se ha decidido centrar la aplicación *SATApp* en las versiones 4.1 (API 16) y 5.1 de Android tal y como se explica a continuación. Asimismo, se han instalado los componentes Google Play Services y Google USB Driver para el buen funcionamiento de Google Maps y realizar emulaciones en un terminal físico, respectivamente.

Para realizar la instalación/actualización, es preciso acceder al menú *Configure* > *SDK Manager* de la pantalla de bienvenida, lo que permite la aparición del SDK Manager de Android. Con esta herramienta ya se puede instalar, desinstalar, o actualizar todos los componentes disponibles como parte del SDK de Android.

Plataforma y dispositivos a los que se dirige

El contenido de esta memoria está enfocado a una aplicación desarrollada exclusivamente para dispositivos Android.

Más concretamente, la instalación será posible sobre cualquier terminal móvil con Android 4.1 (API 16) o superior. Esta decisión ha sido tomada teniendo en cuenta el numeroso porcentaje de usuarios aun existentes de dicha versión mostrado en la figura 5 y actualizado a fecha de enero de 2015.

Version	Codename	API	Distribution
2.2	Froyo	8	0.4%
2.3.3 - 2.3.7	Gingerbread	10	7.8%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	6.7%
4.1.x	Jelly Bean	16	19.2%
4.2.x		17	20.3%
4.3		18	6.5%
4.4	KitKat	19	39.1%

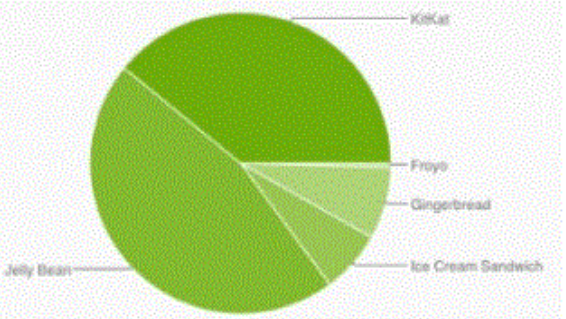


Figura 5. Porcentaje de uso de las versiones de Android en Enero de 2015.

Requisitos mínimos:

- Android 4.1 o superior.
- Instalación y configuración de Google Maps en el terminal móvil.
- Smartphone Android con resolución a partir de 480 x 800 y procesador doble núcleo.
- Conexión a Internet.
- Servicio GPS.

Hay que destacar que, a diferencia de aplicaciones como Trello y Asana de carácter organizativo general, *SATApp* está enfocado a la comunicación, gestión y movilidad de un equipo de trabajo técnico informático.

Este trabajo conlleva una serie de requisitos funcionales explícitos y otra serie de requisitos no funcionales implícitos que se van a presentar a continuación:

Requerimientos funcionales

Los requerimientos del trabajo de final de carrera elegido constan de varios módulos principales con una serie de funcionalidades requeridas. En la tabla resumen de los requerimientos se han

añadido los datos de la funcionalidad correspondiente que se va a implementar en la aplicación final.

REQUERIMIENTOS FUNCIONALES DE SATApp	
Módulos principales	**Funcionalidad requerida**
Mapa basado en API de Google Maps.	Localizar técnico mediante datos de contacto en el dispositivo móvil.
Gestión de GPS a partir de la API de Google.	Facilitar la ubicación de los puntos y su interpretación por parte del usuario.
Gestión de mails enviados desde la aplicación	Envío y recepción de *mails.*
Gestión de incidencias/tareas.	Facilitar el estado e interacción de cada técnico.

Riesgos y requerimientos no funcionales.

La aplicación a desarrollar también dispone de los siguientes puntos débiles, algunos de ellos posiblemente subsanables en posteriores actualizaciones:

- Una mala conexión a la red ralentiza la aplicación. Este inconveniente puede quedar resuelto en posteriores mejoras mediante un sistema de uso *off line*.
- El error humano puede complicar el buen funcionamiento de la aplicación.
- La probable necesidad de un rol de usuario administrador con privilegios sobre el resto de usuarios de la aplicación.
- El técnico debe asegurar la carga de la batería en su teléfono móvil.
- Es aconsejable que la empresa instale y configure la aplicación y los servicios asociados a este en cada terminal.
- Teniendo en cuenta los riesgos personales en el desarrollo de *SATApp*, es necesario considerar la falta de experiencia en este campo y en la realización del proyecto en sí.
- Hay que considerar indeseables problemas en la planificación que pueden conllevar retrasos aunque de ningún modo repercutirá sobre la funcionalidad del proyecto.

3.1. Contenido de la memoria.

A continuación se explica brevemente el contenido de cada capítulo y la bibliografía, que podrán ser modificados en adelante conforme lo requieran las especificaciones:

En el capítulo 1, *Introducción y punto de partida del TFC*, se expone la motivación del presente proyecto y una breve introducción al funcionamiento de la aplicación.

El capítulo 2, *Estado del Arte*, ofrece una descripción general de algunos de los aspectos técnicos relacionados con este proyecto, como aplicaciones móviles relacionadas, el sistema operativo Android, el lenguaje Java y el entorno de desarrollo Eclipse.

En el capítulo 3, *La plataforma Android*, se aporta una visión general del entorno de desarrollo Android y las características que lo definen.

En el capítulo 4, *Objetivos*, se detalla aquello que se desea conseguir, a quien va dirigida la aplicación y su funcionalidad, así como una visión general de los contenidos de la memoria.

En el capítulo 5, titulado *Planificación del trabajo*, se presenta la metodología a emplear y el diagrama de Gantt que se seguirá a lo largo de todo el TFC.

En el capítulo 6, *SATApp, gestor de incidencias técnicas*, se introducirá el análisis y diseño de la aplicación junto con su desarrollo e implementación, sus casos de uso, la arquitectura de cada una de sus partes y las pruebas que definen el correcto funcionamiento de la aplicación.

En el capítulo 7, *Conclusiones finales acerca del proyecto*, se hará un repaso global al proyecto incluyendo una breve valoración del mismo y una descripción de los puntos susceptibles de mejora en versiones posteriores.

Por último, se incluye una bibliografía donde se puede encontrar la referencia a diversos puntos de esta memoria y varios Anexos para datos relevantes a la instalación y uso de la aplicación.

A continuación se detallan de forma resumida los objetivos y entregables de cada fase:

PEC1. Plan de trabajo.

Objetivos:
- Definir el proyecto y realizar su planificación inicial.
- Entrega de PEC1.

El plan de trabajo servirá como guía para el resto del desarrollo del TFC.

PEC2. Análisis, diseño y prototipo.

Objetivos:
- Realizar el análisis de tareas, diseño y prototipo acorde a la metodología DCU.
- Entregar documentación que incluye: perfiles de usuario, contexto de uso, análisis de tareas, escenarios de uso, diagramas de flujo de interacción y un prototipo de alto nivel.

Todas estas tareas sirven como entrada de datos para la siguiente fase.

PEC3. Implementación.

Objetivos:
- Implementar la solución del proyecto y aportar la documentación complementaria
- Entrega de código fuente, instalables y documentación complementaria.

Es la fase final del proyecto en la que se obtiene el producto final. Puede requerir de la iteración con fases anteriores en caso de detectar problemas en el producto.

Entrega final.

Objetivos:
- Finalización del proyecto y documentación.
- Entregar memoria y video de presentación del TFC.

En esta fase se finalizará la fase anterior en caso de no haberlo hecho y se presentará al público objetivo nuestro producto.

Debate.

Objetivos:
- Debatir sobre los trabajos entregados a lo largo del curso y la entrega del proyecto final.

4. Planificación del trabajo.

La metodología a emplear para el desarrollo del TFC se basa en el diseño centrado en el usuario (DCU) estudiado en asignaturas anteriores.

La idea es usar el modelo DCU clásico en las fases de investigación, concepto y diseño dejando la metodología ágil exclusivamente para la fase de codificación.

La justificación de la elección de esta metodología se basa en que la experiencia de usuario y la usabilidad son imprescindibles para el éxito cualquier tipo de producto o software.

4.1. Diagrama de Gantt.

En el siguiente diagrama de Gantt se muestra de forma resumida la planificación del proyecto, sus tareas y las fechas clave.

Diagrama de Gantt

	Nombre de tarea	Duración	Comienzo	Fin	Predecesora
1	TFC Desarrollo Aplicaciones Android - SATApp	94 días?	mié 25/02/15	mar 23/06/15	
2	PEC1 - Plan de trabajo	13 días?	mié 25/02/15	mié 11/03/15	
3	Estudio de App a desarrollar	3 días	mié 25/02/15	vie 27/02/15	
4	Presentación de ideas al consultor	3 días	sáb 28/02/15	mar 03/03/15	3
5	Mejora y propuesta	3 días	mié 04/03/15	vie 06/03/15	4
6	Desarrollo de PEC1	3 días?	sáb 07/03/15	mar 10/03/15	5
7	Entrega PEC1	1 día?	mié 11/03/15	mié 11/03/15	6
8	PEC2 - Análisis, diseño y prototipo	22 días?	jue 12/03/15	mié 08/04/15	2
9	Presentación de ideas al consultor	3 días	jue 12/03/15	dom 15/03/15	7
10	Análisis y diseño	11 días?	lun 16/03/15	dom 29/03/15	9
11	Desarrollo de prototipo	7 días?	lun 30/03/15	mar 07/04/15	10
12	Entrega PEC2	1 día?	mié 08/04/15	mié 08/04/15	11
13	PEC3 - Implementación	32 días?	jue 09/04/15	mié 20/05/15	8
14	Presentación de ideas al consultor	3 días	jue 09/04/15	dom 12/04/15	12
15	Implementación de la App	25 días?	lun 13/04/15	vie 15/05/15	14
16	Desarrollo de la documentación	3 días?	sáb 16/05/15	mar 19/05/15	15
17	Entrega PEC3	1 día?	mié 20/05/15	mié 20/05/15	16
18	TFC - Entrega final	25 días?	jue 21/05/15	dom 21/06/15	13
19	Mejoras y desarrollo final de la memoria	19 días?	jue 21/05/15	lun 15/06/15	17
20	Desarrollo de video de presentación	4 días?	mar 16/06/15	vie 19/06/15	19
21	Entrega TFC	2 días	sáb 20/06/15	dom 21/06/15	20

Figura 6a. Diagrama de Gantt del TFC - Desarrollo de Aplicaciones Android.

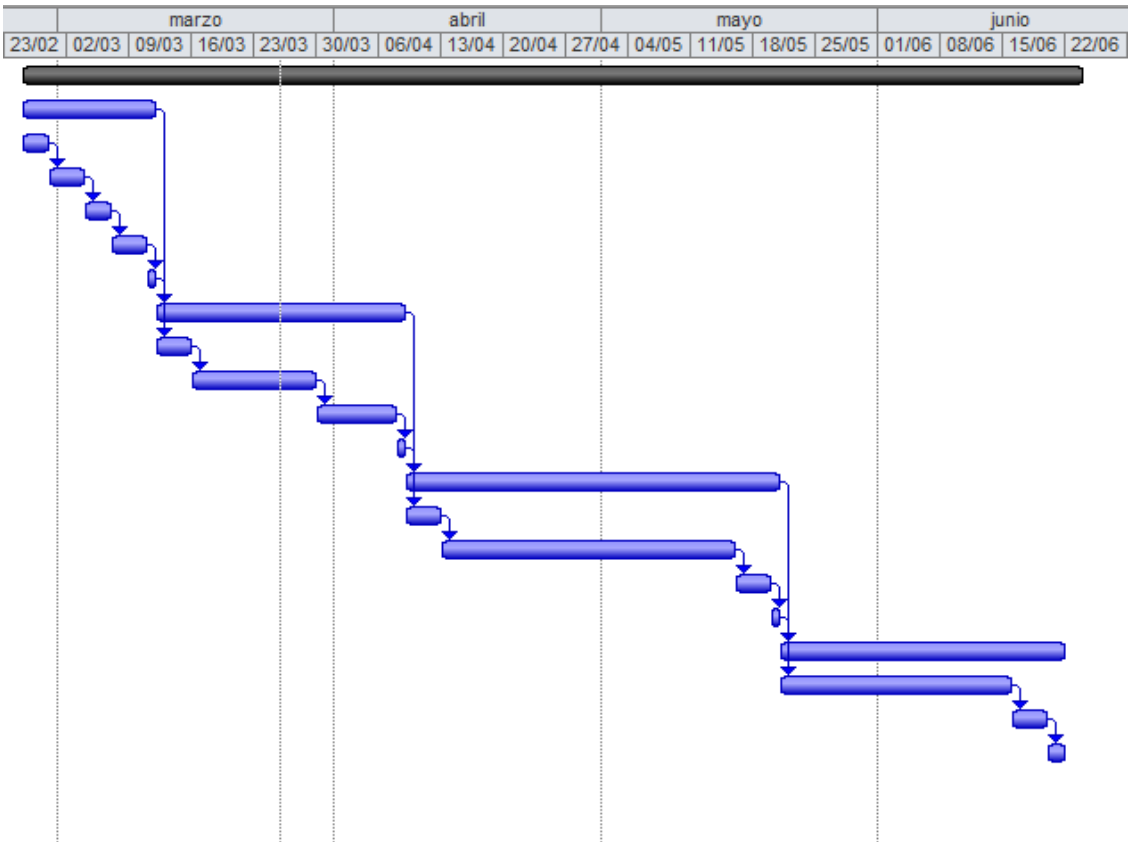


Figura 6b. Diagrama de Gantt del TFC – Desarrollo de Aplicaciones Android.

5. *SATApp*: gestor de incidencias técnicas.

En este capítulo se explica, paso a paso, el desarrollo completo de *SATApp*, una aplicación cuyo objetivo es localizar y mostrar en un mapa a los técnicos mediante sus números de contacto almacenados en el dispositivo, añadiendo además otra serie de funcionalidades.

Mediante este desarrollo se busca ilustrar de una forma más práctica las características principales que ofrece Android y pretende además servir como ejemplo para la creación de otras aplicaciones.

Algunos de los aspectos más interesantes de Android y que se explican con esta aplicación son los siguientes:

- Uso de componentes Activity.
- Uso de componentes Service.
- Solicitudes a través de Intents.
- Utilización de los servicios de Google Maps.
- Control del GPS.
- Comunicaciones por HTTP.
- Uso de la base de datos MySQL.
- Composición del archivo AndroidManifest.xml.
- Acceso a información sobre el propio dispositivo.
- Uso de un Servidor Web gratuito.
- Composición de interfaces de usuario, tanto con código como con XML.
- Declaración y uso de recursos externos.
- Composición gráfica de elementos en pantalla.
- Gestión de opciones de menú.
- Envío de correo electrónico.

5.1. Análisis y DCU de la aplicación.

En los siguientes apartados nos centraremos en el análisis y diseño de la aplicación *SATApp.*

El diseño centrado en el usuario (DCU) es, como su nombre indica, una aproximación al diseño de productos y aplicaciones que sitúa al usuario en el centro de todo el proceso. Así, podemos entender el DCU como una filosofía cuya premisa es que, para garantizar el éxito de un producto, hay que tener en cuenta al usuario en todas las fases del diseño. Además, también podemos entender el DCU como una metodología de desarrollo, es decir, una forma de planificar los proyectos y un conjunto de métodos que se pueden utilizar en cada una de las principales fases.

5.1.1. Introducción a *SATApp*.

El objetivo fundamental de *SATApp* es la geolocalización de los técnicos informáticos a tiempo real y la interacción o comunicación entre los mismos y aquel usuario que dispone del rol de administrador de la aplicación dentro de la empresa.

En primer lugar, la aplicación debe mostrar al administrador la ubicación GPS propia y, tras esto, la de los técnicos mediante la selección de los usuarios o técnicos almacenados en la base de datos. Para ello, se utilizan cinco servicios fundamentalmente:

- La conocida aplicación Google Maps, para mostrar las localizaciones de los usuarios.
- La señal GPS del propio dispositivo móvil, para conocer la propia localización.
- Una base de datos MySQL que contiene la información relevante a los usuarios de la aplicación.
- El intercambio de información referente a la posición de cada usuario mediante un servidor HTTP.
- Una conexión a Internet, con el fin de poder intercambiar información de localización con los técnicos a tiempo real.

Como hemos introducido, el administrador visualiza la situación geográfica de los técnicos mediante sus nombres de usuario gracias a los mapas y servicios de Google Maps, que son ofrecidos por Android a través de una serie de API's específicas. El uso de esta potente herramienta permite tener acceso a miles de imágenes y mapas, convirtiéndose actualmente en uno de los elementos de Android más usado y que más juego está dando en todo tipo de aplicaciones.

Así mismo, la aplicación utiliza el sistema GPS presente en el mismo dispositivo móvil para conocer la ubicación propia, pudiendo de este modo tanto mostrársela al técnico en el mapa como al resto de técnicos que lo deseen. El servicio GPS, que aparece integrado en la totalidad de *Smartphones* de última generación, se ha extendido con rapidez y es usado incluso en redes sociales y aplicaciones dedicadas a previsiones meteorológicas. Todo parece indicar que en breve su incorporación al desarrollo de aplicaciones de movilidad será tan frecuente como lo es hoy el uso la cámara digital o el reproductor de música en los terminales móviles.

Hay que destacar que *SATApp* no requiere la presencia del GPS para poder funcionar, pero su ausencia significa que el usuario no podrá conocer su propia ubicación real y, por tanto, no podrá comunicársela a los técnicos. Recordamos que esta es la funcionalidad principal y la más atractiva de cara a la localización de un empleado por la empresa.

Por otro lado, mediante el establecimiento de una conexión a Internet, la aplicación realiza intercambios de información sobre localizaciones entre los usuarios o técnicos. Este intercambio no se realiza nodo a nodo, es decir, entre ellos, sino que existe un servidor central donde cada uno de los usuarios consulta las localizaciones de los empleados, además de actualizar su propia localización. Este intercambio se realiza utilizando el estándar XML y sobre el protocolo HTTP, aunque los detalles de funcionamiento de esta comunicación y del servidor se detallará en apartados posteriores.

Esta funcionalidad permite que, al igual que pasa con el sistema GPS, no sea imprescindible disponer de una conexión a Internet constante y fiable. De este modo, *SATApp* puede mostrar las ubicaciones de los contactos en todo momento y siempre intenta que esta sea lo más reciente posible. En caso de no poder conectarse, la aplicación mostrará siempre la última ubicación conocida.

Además de mostrar ubicaciones geográficas, *SATApp* permite también realizar otra serie de acciones asociadas a cada uno de los contactos como, por ejemplo, realizar una llamada telefónica, visualizar el estado de la reparación o enviar correos electrónicos.

A continuación, se estudiarán los aspectos relevantes al análisis y diseño de la aplicación *SATApp* aplicando la metodología del diseño centrado en el usuario con el fin de obtener una usabilidad y accesibilidad óptimas en el producto final. De este modo, el plan de trabajo se dividirá en cuatro partes principales: Investigación, Diseño conceptual, Prototipado y Evaluación.

Este trabajo se caracteriza por tener un desarrollo integrado, es decir, los resultados de cada fase del proyecto han servido como datos de entrada para la fase siguiente. Al mismo tiempo, el proyecto de DCU tiene etapas y fases que se retroalimentan y esto es clave a la hora de planificar y llevar a cabo el diseño de un sistema.

5.1.2. Investigación y análisis de usuarios.

El primer paso en la fase de investigación del diseño de la aplicación *SATApp* es tratar de conocer a los usuarios del sistema, el contexto en el que lo usan, y las tareas que realizan. Esto permitirá garantizar que todos los factores que afectan a la utilización del sistema se hayan identificado antes de que comiencen las labores de diseño, y proporcionará un punto de partida para diseñar posteriormente las pruebas de usabilidad.

Para realizar esta fase y conocer los perfiles de usuarios y su contexto de uso utilizaremos los siguientes métodos:

- **Observación e investigación contextual.** Los resultados se basarán en la observación de los usuarios en su entorno habitual. Es una técnica realmente útil para conocer de manera objetiva qué hacen los usuarios, y en qué condiciones lo hacen. La información que se obtiene puede servir para determinar sus comportamientos y evaluar su usabilidad.
- **Dinámica de grupo.** En ella se realizará una entrevista con cada grupo de cuatro empleados que forman una pyme. La moderación de estos grupos es clave para recoger información de calidad y conseguir la participación de todos los asistentes. Estas entrevistas son moderadas mediante un guion para conducir la conversación en el orden que interesa y hacia los temas que se quieren investigar. Suele ser un proceso largo y puesto que puede aportar mucho a los participantes, se esperan resultados claros y en profundidad.
- **Análisis competitivo.** En este punto, será necesario analizar aplicaciones similares y/o que compiten con el sistema interactivo que se está diseñando. Los objetivos de analizar dichos servicios son múltiples: conocer las expectativas de los usuarios, entender las tendencias del mercado, aprender de los errores pero también de lo que funciona, conocer las funcionalidades básicas o comunes, estudiar las interfaces, etc.

Hay que destacar que los usuarios han sido elegidos siguiendo una serie de requisitos específicos y en base a su nivel representativo en la tarea que nos acontece. Algunas de las características más importantes se detallan a continuación:

- Los usuarios provienen del sector de la informática y las telecomunicaciones. Más concretamente, se ha creído conveniente realizar el método de investigación a los empleados de una serie de pymes para comprobar realmente lo que *SATApp* puede aportar en una empresa.
- Como punto demográfico de aplicación del proyecto se elige la Región de Murcia.

- Los usuarios tienen edades comprendidas entre los 25 y los 40 años.
- Las pymes mencionadas junto con sus empleados han sido seleccionados en relación a sus intereses, motivaciones y experiencia con el uso de la tecnología móvil.

Asimismo, se parte de la premisa que no existe una jerarquía de usuarios que cubre amplias gamas del sector en cuestión, sino que todos son técnicos reparadores con los mismos privilegios. Con esta decisión se espera obtener información concreta sobre comportamientos pasados, actitudes e intuiciones y se cuenta con que la información recogida puede estar afectada por diferentes sesgos e influencias.

Al mismo tiempo, se ha confirmado que todos los entrevistados usan un dispositivo Android.

Por otro lado, se entiende que la entrevista individual también hubiese sido una opción útil y cualitativa, a diferencia del método elegido. Mediante esta, se parte de una muestra pequeña de usuarios para obtener los datos y se usan guiones abiertos y poco estructurados. Teniendo en cuenta que se dispone de un perfil alto en cuanto al nivel de conocimientos informáticos y se desea una respuesta grupal y consensuada respecto al uso de *SATApp* en cada pyme, se estima más conveniente el método en grupo elegido.

Llegados a este punto, se presenta a cuatro pymes distintas que disponen de cuatro usuarios cada una. Estos formarán parte en este proceso de usabilidad del prototipo de la aplicación con el fin de obtener los datos buscados.

Pyme número 1 – Digitalia, S.L.
Situación: Las Torres de Cotillas (Murcia)

Presentación de usuario:
Pedro tiene 32 años y trabaja a jornada completa como Comercial Técnico Informático. En este empleo, realiza labores de venta, reparación, mantenimiento y configuración de equipos a empresas de toda la comunidad.

Habilidades y conocimientos relacionados con el presente proyecto:
- Como técnico superior en Administración de Sistemas Operativos en Red, posee conocimientos amplios de informática y telecomunicaciones.
- Conoce la tecnología Android y sus principales características, aunque no realiza funciones de programación y esos conocimientos se escapan a su área de interés.

Aptitud y motivación:
- Está abierto a estudiar el funcionamiento del prototipo de la aplicación y le motiva comprobar si le sería útil en sus funciones diarias.

Relación con la tecnología en cuestión:
- Dispone de un terminal Android, conoce Google Maps y hace uso diario de todo tipo de aplicaciones.

Presentación de usuario:
Carmen tiene 27 años, es Ingeniero Técnico Informático de Sistemas y realiza labores de mantenimiento y configuración de equipos a empresas de toda la comunidad, así como diversos proyectos enfocados al desarrollo de software.

Habilidades y conocimientos relacionados con el presente proyecto:
- Posee conocimientos amplios acerca del desarrollo de aplicaciones Android y ha realizado proyectos en este campo con anterioridad.

Aptitud y motivación:
- Se siente atraída por la funcionalidad real de la aplicación y sus futuras mejoras.

Relación con la tecnología en cuestión:
- Conoce todas las tecnologías relacionadas con el proyecto, así como sus técnicas de implementación y desarrollo.

Presentación de usuario:
Carlos tiene 29 años, trabaja a media jornada y es Analista Programador. En su día a día diseña soluciones relacionadas con el IoT y la movilidad en empresas que tienen el servicio de mantenimiento contratado.

Habilidades y conocimientos relacionados con el presente proyecto: - Es experto programador y tiene conocimientos a nivel profesional en cuanto al desarrollo de aplicaciones Android. Aun usa el entorno Eclipse en sus proyectos.
Aptitud y motivación: - Le resulta interesante el proyecto puesto que puede implantarse en su empresa. Tiene pensadas algunas mejoras al respecto.
Relación con la tecnología en cuestión: - Trabaja a diario con este tipo de tecnología y es experto en programación Java.

	**Presentación de usuario:** **Alejandro tiene 29 años y es Administrador de Sistemas Windows y Diseñador Web. Realiza tareas de *helpdesk* y resolución de incidencias a empresas mediante virtualización de servidores.**
Habilidades y conocimientos relacionados con el presente proyecto: - No tiene nociones de desarrollo de aplicaciones Android. Por otro lado, conoce ampliamente el mundo del desarrollo de aplicaciones Web. - Su perfil es puramente técnico.	
Aptitud y motivación: - Se muestra positivo ante la idea de proyecto puesto que puede llevarla a su terreno.	
Relación con la tecnología en cuestión: - No trabaja con el mundo Android.	

Pyme número 2 – PCBox, S.L.
Situación: Murcia (Murcia).

	**Presentación de usuario:** **Luís tiene 32 años y es el Ingeniero Jefe de la empresa. En su puesto, realiza labores de reparación, mantenimiento y configuración de equipos a empresas, aunque generalmente, supervisa el trabajo del resto de los empleados.**
Habilidades y conocimientos relacionados con el presente proyecto: - Es Ingeniero Informático con experiencia en los campos de la programación. - Tiene amplios conocimientos técnicos aunque últimamente está más centrado en delegar y ampliar su área de influencia empresarial. - Está acostumbrado a organizar grupos de trabajo.	
Aptitud y motivación: - Está interesado en formar parte del equipo que testee la aplicación y comprobar si puede sacarle partido.	
Relación con la tecnología en cuestión: - Dispone de un terminal Android, conoce Google Maps y hace uso diario de todo tipo de aplicaciones. Hasta ahora, ha usado Trello para realizar funciones similares en la empresa.	

	**Presentación de usuario:** **Juan de Dios tiene 34 años, trabaja a media jornada y es Analista Programador. En su día a día diseña software a empresas en varias plataformas y se encarga del mantenimiento de los servidores.**
Habilidades y conocimientos relacionados con el presente proyecto: - Es experto programador en varios lenguajes y tiene conocimientos básicos de desarrollo de aplicaciones Android con Eclipse.	
Aptitud y motivación: - Es algo escéptico respecto al uso de la aplicación puesto que su perfil es puramente programador y no cree que se ajuste a sus necesidades.	
Relación con la tecnología en cuestión: - Tiene pocos conocimientos de Android. Su fuerte es la programación orientada a objetos.	

Presentación de usuario:
Alfonso tiene 25 años y es Técnico reparador. Realiza tareas de montaje, reparación y resolución de incidencias en equipos para particulares y empresas con contrato de mantenimiento.

Habilidades y conocimientos relacionados con el presente proyecto:
- No tiene nociones de desarrollo de aplicaciones Android y su fuerte no es la programación.
- Su perfil es de técnico en *hardware*.

Aptitud y motivación:
- Se muestra positivo ante la idea de proyecto puesto que puede ayudarle a realizar su trabajo de manera más profesional.

Relación con la tecnología en cuestión:
- No trabaja con el mundo Android y no es programador.
- Dispone de un teléfono con tecnología Android en la empresa pero no le saca partido.

Presentación de usuario:
Antonio tiene 26 años y es Administrador de Sistemas e instalador. Realiza tareas de *helpdesk*, instalación, configuración y resolución de incidencias a empresas y particulares.

Habilidades y conocimientos relacionados con el presente proyecto:
- No tiene nociones de desarrollo de aplicaciones Android y su fuerte no es la programación.
- Su perfil es técnico.

Aptitud y motivación:
- Se muestra positivo e interesado por probar la aplicación si le facilita el trabajo diario.

Relación con la tecnología en cuestión:
- No trabaja con el mundo Android y no es programador.

Pyme número 3 – General Informática, S.L.
Situación: Molina de Segura (Murcia).

Presentación de usuario:
Manuel tiene 31 años y trabaja como Técnico Informático. En este empleo, realiza labores de reparación, mantenimiento y configuración de equipos a particulares y empresas.

Habilidades y conocimientos relacionados con el presente proyecto:
- Posee amplios conocimientos técnicos de informática y telecomunicaciones.
- No realiza funciones de programador.

Aptitud y motivación:
- Desea comprobar lo que la aplicación puede ofrecerle.

Relación con la tecnología en cuestión:
- Dispone de un terminal Android que usa para comunicarse con el resto de empleados vía llamada telefónica.

Presentación de usuario:
Juan tiene 28 años, trabaja a media jornada y es Técnico Informático. Realiza labores de reparación y configuración de equipos en la oficina técnica.

Habilidades y conocimientos relacionados con el presente proyecto:
- Amplios conocimientos técnicos.
- Es usuario habitual de terminales Android y conoce sus detalles de configuración.

Aptitud y motivación:
- Le resulta interesante conocer los beneficios que el proyecto puede aportar a su trabajo.

Relación con la tecnología en cuestión:
- No tiene conocimientos a nivel programación en Android.

	**Presentación de usuario:** Damián tiene 25 años y es Técnico Administrador de Redes. Se encarga de la instalación de equipos en red en las empresas que contratan el mantenimiento y ofrecer soporte remoto cuando es necesario.
Habilidades y conocimientos relacionados con el presente proyecto: - No tiene nociones de desarrollo de aplicaciones Android.	
Aptitud y motivación: - Cree que la aplicación aportará rapidez de movimientos y mayor profesionalidad.	
Relación con la tecnología en cuestión: - Desconoce la tecnología. - Su perfil es puramente técnico.	

	**Presentación de usuario:** Álvaro tiene 28 años y trabaja a jornada completa como Técnico Informático. En este empleo, realiza labores de reparación, mantenimiento y configuración de equipos.
Habilidades y conocimientos relacionados con el presente proyecto: - Como técnico superior en Administración de Sistemas Informáticos, posee amplios conocimientos de informática y telecomunicaciones. - Conoce la tecnología Android y sus características de desarrollo.	
Aptitud y motivación: - Tiene interés en comprobar el funcionamiento del prototipo de la aplicación, aportar mejoras y utilizarlo en la empresa.	
Relación con la tecnología en cuestión: - Sabe programar en Android usando el IDE Eclipse.	

Pyme número 4 – InforMarket, S.L.
Situación: Murcia (Murcia).

	**Presentación de usuario:** Francisco tiene 33 años y es Programador Senior y Jefe de Proyectos. A diario, realiza labores de programación en distintos lenguajes, análisis de bases de datos y desarrollo de software.
Habilidades y conocimientos relacionados con el presente proyecto: - Tiene conocimientos a nivel profesional de lenguajes de programación como Java, C# o VB.	
Aptitud y motivación: - No conoce demasiado el campo y es reacio al uso de la aplicación. Piensa que sería más útil si fuese creada en un entorno cliente/servidor.	
Relación con la tecnología en cuestión: - No ha tenido contacto con la solución Android Studio pero es un experto programador acostumbrado a trabajar con Eclipse.	

	**Presentación de usuario:** Maribel tiene 28 años, es Diseñadora Web y Programadora a tiempo parcial. Realiza labores consultoría profesional, programación en varios lenguajes para pequeños aplicativos y diseño Web profesional a todos los niveles.
Habilidades y conocimientos relacionados con el presente proyecto: - Posee amplios conocimientos en programación orientada a objetos. - Es experta en los entresijos del diseño Web.	
Aptitud y motivación: - Se siente atraída por la funcionalidad real de la aplicación aunque espera una ampliación que le permita sacarle partido.	
Relación con la tecnología en cuestión: - No tiene relación con Android aunque le resulta familiar el código usado.	

	**Presentación de usuario:** Paco tiene 36 años y es Ingeniero Técnico Informático de Gestión. Realiza tareas de reparación y mantenimientos de equipos junto con análisis y diseño de bases de datos profesionales.
Habilidades y conocimientos relacionados con el presente proyecto: - Tiene un perfil de analista de datos. - Posee experiencia como administrador de sistemas y soluciones en *hardware*.	
Aptitud y motivación: - Se muestra positivo ante la idea de proyecto puesto que puede llevarla a su terreno.	
Relación con la tecnología en cuestión: - No tiene nociones de desarrollo de aplicaciones Android ni está familiarizado con las posibilidades que esto conlleva.	

	**Presentación de usuario:** Carlos tiene 27 años y es Técnico Informático. Realiza tareas de *helpdesk* y resolución de incidencias a particulares y empresas.
Habilidades y conocimientos relacionados con el presente proyecto: - Tiene un perfil claramente técnico.	
Aptitud y motivación: - Se muestra positivo ante la idea de proyecto puesto que puede facilitarle el trabajo.	
Relación con la tecnología en cuestión: - No trabaja con el mundo Android pero es curioso y dispone de información amplia a nivel usuario.	

A continuación, se presenta el planteamiento, desarrollo y resultado obtenido mediante los métodos de observación e investigación contextual, dinámica de grupo y análisis competitivo.

Para el proceso de observación e investigación se ha examinado al usuario haciendo uso de la aplicación *SATApp* en su entorno diario. Más tarde, se ha desarrollado una dinámica de grupo para cada pyme teniendo en cuenta la aptitud y resultados obtenidos por el primer método. Por último, se ha realizado un análisis comparativo tomando como referencia el conjunto de aplicaciones similares estudiado anteriormente en el Estado del Arte de este proyecto.

Usuarios:	Empleados de Pyme número 1 – Digitalia, S.L.
Escenario:	Pedro toma el rol de administrativo desde la oficina mientras que Carmen, Alejandro y Carlos asumen los roles de técnicos. Recordamos que la usabilidad y los privilegios de ambos roles son idénticos.
Estudio:	Observación, investigación contextual, dinámica de grupo y análisis en profundidad de los miembros de Digitalia.
Contexto:	1. *Login* y registro de usuario. 2. Uso de Google Maps para localizar contactos. 3. Interacción con usuarios.
Observaciones:	Usabilidad: alta. Visibilidad: media. Rapidez: media. Funcionalidad: media. Eficacia: media. Utilidad: alta. Satisfacción: alta.
Justificación:	La aplicación se ha usado en un entorno ideal y ha ofrecido buenos resultados. En general, se echan de menos más funcionalidades y opciones.
Dinámica de grupo:	Pregunta: Valorad la usabilidad de la aplicación. Respuesta: En general, es atractiva y fácil de usar. Le damos un 7. Pregunta: ¿Os parece que su manejo es intuitivo? Respuesta: Sí, todo está a simple vista y aporta sencillez. Pregunta: Valorad la utilidad en el mercado de la aplicación. Respuesta: Ha resultado útil en las labores que promete. Eso sí, el envío de

	mensajes no es útil ni necesario. Pregunta: ¿Permite el sistema una buena comunicación entre usuarios? Respuesta: Siempre que tenga acceso a la red y cobertura. Pregunta: Valorad la visibilidad de la aplicación. Respuesta: La interfaz es atractiva y sencilla pero existe poca diferencia de contrastes y no se diferencian del todo las opciones.
Análisis competitivo:	Aplicación: Trello. Conclusiones: La interfaz es sencilla y amigable. No está enfocada a dar el mismo servicio pero sus funcionalidades son parecidas. Aplicación: ClaveiSat. Conclusiones: Es más completa y profesional, aunque también más lenta. Aplicación: Asana. Conclusiones: Está en inglés y nos parece poco intuitiva. Aplicación: GD Systems. Conclusiones:
Resultados:	Satisfactorios. Se dan a conocer aspectos reales acerca de la funcionalidad de la aplicación y gusta su simplicidad. Se anotan los defectos respecto al diseño para su posterior mejora.

Usuarios:	**Empleados de pyme número 2 - PCBox, S.L.**
Escenario:	Luís toma el rol de administrativo desde la oficina mientras que Juan de Dios, Alfonso y Antonio asumen los roles de técnicos como es habitual. Recordamos que la usabilidad y los privilegios de ambos roles son idénticos.
Estudio:	Observación, investigación contextual, dinámica de grupo y análisis en profundidad de los miembros de PCBox.
Contexto:	1. *Login* y registro de usuario. 2. Uso de Google Maps para localizar contactos. 3. Interacción con usuarios.
Observaciones:	Usabilidad: alta. Visibilidad: media. Rapidez: media. Funcionalidad: alta. Eficacia: media. Utilidad: baja. Satisfacción: media.
Justificación:	Aporta unas soluciones necesarias a la empresa. Se esperan actualizaciones de mejora en adelante.
Dinámica de grupo:	Pregunta: Valorad la utilidad en el mercado de la aplicación. Respuesta: Conocemos nuestra ubicación y podemos comunicarnos, pero necesitamos más herramientas. Pregunta: ¿Permite el sistema una buena comunicación entre usuarios? Respuesta: Sí, mediante mensajes y correo electrónico. Es más útil que usar *Whatsapp* por un lado y la aplicación por otro. Pregunta: ¿Permite el sistema un fácil manejo de los mapas? Respuesta: En unos terminales es más rápido que en otros pero en definitiva funciona con normalidad. Pregunta: ¿Echáis en falta alguna funcionalidad en la aplicación? Respuesta: Resultaría interesante incluir una agenda, herramientas de contabilidad y facturación, ayuda a usuarios y un calendario. Pregunta: Valorad la visibilidad de la aplicación. Respuesta: Muy simple pero poco llamativa. Quizá los contrastes y los colores deberían ser más diferenciados.
Análisis competitivo:	Aplicación: Trello. Conclusiones: No está enfocada a la resolución de incidencias técnicas como *SATApp* pero la interfaz y el diseño son ideales. Aplicación: ClaveiSat. Conclusiones: Es muy completa y contiene funcionalidades interesantes. Aplicación: Asana. Conclusiones: No traducida al español y de paga. Aplicación: GD Systems.

	Conclusiones: Muy sencilla y amigable.
Resultados:	Razonablemente buenos. Se nota la falta de herramientas aunque las funcionalidades actuales resultan interesantes para la buena organización de la empresa. Se deben realizar cambios en el diseño.

Usuarios:	**Empleados de pyme número 3 – General Informática, S.L.**
Escenario:	Manuel toma el rol de administrativo desde la oficina mientras que Juan, Damián y Álvaro asumen los roles de técnicos. Recordamos que la usabilidad y los privilegios de ambos roles son idénticos.
Estudio:	Observación, investigación contextual, dinámica de grupo y análisis en profundidad de los miembros de General Informática.
Contexto:	1. *Login* y registro de usuario. 2. Uso de Google Maps para localizar contactos. 3. Interacción con usuarios.
Observaciones:	Usabilidad: alta. Visibilidad: media. Rapidez: alta. Funcionalidad: alta. Eficacia: media. Utilidad: media. Satisfacción: alta.
Justificación:	Gusta la sencillez y resulta útil para el control de los empleados y sus tareas.
Entrevista:	Pregunta: Valorad la utilidad en el mercado de la aplicación. Respuesta: Nos parece bastante funcional, innovadora y útil en nuestro mercado. Pregunta: Explicad los inconvenientes en cuanto a la usabilidad de la aplicación en tu entorno. Respuesta: No estamos habituados a usar aplicaciones para comunicarnos entre nosotros ni a controlar nuestra situación real por GPS. Pregunta: ¿Cómo mejoraríais la aplicación *SATApp*? Respuesta: Incluiría servicio *off-line*, gestión económica y agenda de clientes. Pregunta: Valorad la interfaz de la aplicación. Respuesta: Es sencilla. Gustan los colores aunque no destacan. Pregunta: Valorad la rapidez de la aplicación. Respuesta: Nos ha parecido realmente fluida.
Análisis competitivo:	Aplicación: Trello. Conclusiones: Es la alternativo en nuestro entorno empresarial. Aplicación: ClaveiSat. Conclusiones: Dispone de más opciones como el kilometraje y la agenda, pero da errores y no se adapta a nuestras necesidades. Aplicación: Asana. Conclusiones: Muy completa pero de paga. Es interesante la integración de herramientas. Aplicación: GD Systems. Conclusiones: Demasiado básica.
Resultados:	Satisfactorios. Buscan la sencillez y el control de la plantilla.

Usuarios:	**Empleados de pyme número 4 – InforMarket, S.L.**
Escenario:	Francisco toma el rol de administrativo desde la oficina mientras que Maribel, Paco y Carlos asumen los roles de técnicos. Recordamos que la usabilidad y los privilegios de ambos roles son idénticos.
Estudio:	Observación, investigación contextual, dinámica de grupo y análisis en profundidad de los miembros de InforMarket.
Contexto:	1. *Login* y registro de usuario. 2. Uso de Google Maps para localizar contactos. 3. Interacción con usuarios.
Observaciones:	Usabilidad: alta. Visibilidad: media. Rapidez: alta.

	Funcionalidad: media. Eficacia: media. Utilidad: media. Satisfacción: media.
Justificación:	Cubre las necesidades que están buscando e interesa.
Entrevista:	Pregunta: Valorad la usabilidad de la aplicación Respuesta: Genial. Se ajusta muy bien a nuestras necesidades. Pregunta: ¿Creéis que su manejo es intuitivo para todos los usuarios? Respuesta: Si, las opciones están a la vista y el diseño es amigable. Pregunta: ¿Permite el sistema una buena comunicación entre usuarios? Respuesta: Si, tiene varios métodos de comunicación a tiempo real. Pregunta: ¿Qué funcionalidades echáis en falta en la aplicación? Respuesta: Nada excepcional a simple vista. Quizá en adelante. Pregunta: Valorad la funcionalidad de la aplicación Respuesta: Es simple a la vez que funcional y novedosa. El envío de mensajes conlleva un coste y no gusta.
Análisis competitivo:	Aplicación: Trello. Conclusiones: No se adapta a nuestras necesidades y no dispone de geolocalización. Aplicación: ClaveiSat. Conclusiones: No hemos logrado que funcione de manera correcta. Aplicación: Asana. Conclusiones: Es poco intuitiva y bastante lenta. Aplicación: GD Systems. Conclusiones: Le faltan muchas opciones aunque facilita manual.
Resultados:	Razonablemente buenos. Cubre las necesidades propuestas.

Conclusiones finales

Aunque el conjunto de usuarios estudiados no es muy extenso, todos han aportado detalles que se tienen en cuenta y coinciden en la importancia de los siguientes aspectos:

- Importancia de mejoras e implementación de funcionalidades más profesionales.
- La interfaz es un punto fuerte y gusta.
- Es razonablemente útil en el entorno estudiado de una pyme.
- Se suprime la opción valorada acerca del envío de mensajes de texto por no causar buena impresión para los usuarios estudiados y entender dicha valoración como nada positiva.
- La idea resulta innovadora aunque siempre es mejorable.

Gracias a los datos de la entrevista anterior, se han descubierto deficiencias y funcionalidades en la aplicación *SATApp* y podemos apreciar que:

- Los usuarios potenciales del sistema serán aquellos con perfil técnico cuyas funciones estén sujetas a la interacción entre los miembros implicados y la organización del trabajo diario como técnico reparador.
- Entre muchos otros aspectos, el objetivo principal de la aplicación *SATApp* será mejorar el sistema de atención al cliente usando una aplicación que aporte rapidez, comunicación y gestión interna.
- El diseño de la aplicación ha de ser elegante, rápido, amigable y sencillo al mismo tiempo. Asimismo, no debe estar sobrecargado de información para evitar que se produzcan retardos, brechas de ejecución o brechas en la evaluación.
- La interfaz debe ser suficientemente intuitiva para no tener que usar la opción de ayuda. Se ha decidido no incluir esta funcionalidad debido a que el usuario dispone de conocimientos suficientes en relación a su perfil y debería ser capaz de navegar por la aplicación sin incidencias.
- Las funcionalidades de facturación, contabilidad, agenda y calendario no son necesarias en este proyecto. La falta de tiempo y de conocimientos en este punto impide el desarrollo de una aplicación profesional y completa aunque estos detalles,

como muchos otros, quedan pendiente para futuras mejoras. De cualquier manera, las funcionalidades comentadas no entraron nunca dentro del fin buscado en este proyecto.

5.1.2.1. Perfiles de usuario.

Una vez realizados los métodos anteriores, podemos concluir que la segmentación de usuarios según su perfil informático no sería efectiva puesto que se ha comprobado como satisface las necesidades del sector técnico.

Aunque para el buen uso de la aplicación no es necesaria una experiencia mínima entorno al mundo de la tecnología Android, sí que es primordial su funcionamiento en un entorno de trabajo.

Demográficamente, la aplicación es usable en cualquier lugar aunque por motivos de sencillez se centrará en dar servicio a la ciudad de Murcia (con posibilidad de ampliación en el futuro). A su vez, el sexo y lingüística del usuario tampoco es un rasgo de importancia ni destacable en este proyecto pero su edad se limita a rangos comprendidos entre los 25-40 años por motivos profesionales.

Teniendo en cuenta los argumentos anteriores, los usuarios van a ser segmentados de acuerdo al interés que presenten en el uso de la aplicación, asumiendo que la han descargado e instalado en los dispositivos móviles de su puesto de trabajo y entienden la necesidad empresarial que están cubriendo.

Estos son los perfiles de usuario a considerar para el uso de la aplicación *SATApp*:

- **Usuario administrador:** hace uso de la aplicación desde la central y controla la geolocalización del resto de usuarios en la empresa, se comunica con estos mismos miembros y visualiza los cambios que se van dando en sus estados. Este usuario puede realizar funciones de técnico si lo estima oportuno.
- **Usuario técnico:** hace mayor uso de la aplicación. Añade todo tipo de información relevante a las reparaciones, busca información GPS necesaria, añade su estado de reparación, realiza llamadas, envía correos electrónicos y gestiona los recursos proporcionados por la aplicación al máximo.

Estimando los datos obtenidos de los usuarios anteriores, se puede concluir que la aplicación *SATApp* está dirigida a todos aquellos usuarios con un perfil informático característico que, desde diferentes dispositivos Android de media o alta gama, quieran llevar un control de las reparaciones o visitas a clientes mediante geolocalización, comunicación y control de servicios en una sola aplicación. Por supuesto, las funcionalidades que ofrece *SATApp* son flexibles respecto a modificaciones y pueden ser aumentadas o incluso mejoradas.

Para un necesario estudio de los perfiles de usuario, tomaremos un usuario característico de cada grupo de pymes:

Usuario:	Pedro.
Perfil de usuario:	Técnico Informático Jefe.
Descripción del perfil:	Dueño del negocio con conocimientos técnicos muy elevados e interés por realizar mejoras importantes respecto a la eficiencia de los servicios que ofrecen y el nivel de profesionalidad de sus empleados.
Rol en la aplicación:	Administrador.
Necesidades que cubre:	Organización de empleados, comunicación y gestión de incidencias.
Tareas que realiza:	Administrativas, de control, localización de técnicos, asignación de reparaciones según su localización y soporte remoto.

Usuario:	Alfonso.
Perfil de usuario:	Técnico reparador en PCBox.
Descripción del perfil:	Perfil técnico con grandes conocimientos en hardware y nuevas tecnologías.
Rol en la aplicación:	Técnico.
Necesidades que cubre:	Rapidez de respuesta, comunicación y organización de tareas.
Tareas que realiza:	Localización de usuarios, apoyo a técnicos, interacción con el resto de técnicos e intercambio de datos con administrador.

Usuario:	Damián.
Perfil de usuario:	Técnico Administrador de Redes.
Descripción del perfil:	Solución de incidencias en redes de computadoras, instalación y configuración de terminales y mantenimiento en general.
Rol en la aplicación:	Técnico.
Necesidades que cubre:	Apoyo al resto de técnicos en tiempo real, comunicación y organización de tareas.
Tareas que realiza:	Interacción con usuarios, intercambio de información con su equipo de trabajo y localización.

Usuario:	Maribel.
Perfil de usuario:	Diseñadora Web y programadora.
Descripción del perfil:	Diseño Web profesional, asesoramiento a empresas, desarrollo de *software* y resolución de incidencias *on line*.
Rol en la aplicación:	Técnico.
Necesidades que cubre:	Apoyo a tiempo real, rapidez de respuesta a incidencias y comunicación y organización con el resto de componentes del equipo.
Tareas que realiza:	Localización de técnicos y comunicación con el resto de usuarios de la empresa.

5.1.2.2. Contextos de uso.

Puesto que *SATApp* es una aplicación enfocada a empresas, se debe concretar que a nivel técnico, es imprescindible disponer de un dispositivo Smartphone con sistema operativo Android, GPS y conexión a Internet móvil para el óptimo funcionamiento de la aplicación. Por otro lado, a nivel informático, no existe una limitación concreta más allá de los conocimientos mínimos necesarios para usar la tecnología del dispositivo y disponer del mismo.

Teniendo en cuenta los datos anteriores, un posible contexto de uso de la aplicación sería aquel en el que un usuario con perfil tecnológico desea implantar la aplicación en su entorno de trabajo porque necesita un mayor control de los procesos realizados por sus empleados que transcurren en la empresa a diario, una comunicación a tiempo real sin costes y una localización instantánea que permita una respuesta rápida desde la central. Hay que destacar, como se ha comentado anteriormente, que *SATApp* no genera los mismos resultados en todos los entornos tecnológicos empresariales y por tanto se minimizará su radio de actuación a pymes, donde su rendimiento es mayor y sus resultados, útiles.

A continuación, se detallan los contextos de uso más corrientes para la aplicación *SATApp* en relación al grupo de usuarios estudiados en el punto anterior:

Usuario:	Pedro.
Tipo de uso:	Registro de usuario.
Contexto:	No dispone de conexión a Internet.
Conclusión:	No se puede realizar la conexión HTTP y no dispone de conexión *off line*.

Usuario:	Alfonso.
Tipo de uso:	Localización GPS de usuario.
Contexto:	El servicio GPS está desconectado.
Conclusión:	El sistema ofrece una ventana de error para que el usuario active el servicio en su terminal.

Usuario:	Damián.
Tipo de uso:	Inicio de sesión.
Contexto:	No dispone de cobertura móvil.
Conclusión:	El sistema no puede comprobar la autenticación de usuario sin una conexión a Internet y deberá intentarlo más tarde.

Usuario:	Maribel.
Tipo de uso:	Envío de correo electrónico a usuario.
Contexto:	El técnico que recibe el correo no ha abierto la aplicación, por tanto, la localización GPS ofrecida por el sistema es errónea.
Conclusión:	El correo electrónico es independiente de la aplicación y es recibido siempre que el usuario disponga de conexión a Internet.

5.1.2.3. Análisis de tareas

En este apartado se va a profundizar algo más en los casos de uso, las necesidades y las tareas que los usuarios realizarán cuando utilicen la aplicación *SATApp.* Con esto, se busca comprender mejor las funciones que realiza cada perfil dentro de la aplicación.

Como recordatorio de los conceptos aprendidos en Ingeniería del Software, se aclara que un caso de uso representa un uso típico que se le da al sistema. La técnica de los casos de uso permite capturar y definir los requisitos que debe cumplir una aplicación, y describe las típicas interacciones que hay entre un usuario y esta. Dicha técnica es utilizada con frecuencia por los ingenieros de software para, entre otras cosas, mostrar al cliente de forma clara y sencilla qué tipo de acciones podrá realizar su futuro sistema.

A continuación se muestra un diagrama con los casos de uso asociados a *SATApp*, utilizando el estándar UML 2.0.

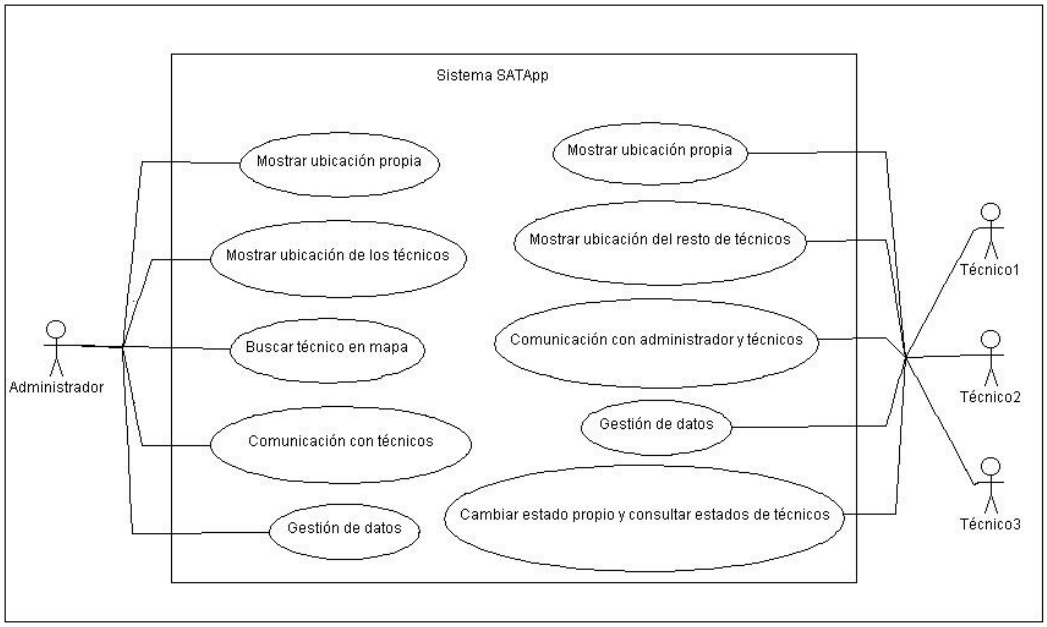


Figura 7. Diagrama de Casos de Uso de la aplicación *SATApp*.

A continuación se ofrece una breve descripción del cometido de cada uno de los casos de uso mostrados en el diagrama anterior:

- **Mostrar ubicación propia**: muestra en el mapa la ubicación actual del usuario que inicia sesión.
- **Listar usuario**: lista todos aquellos usuarios contenidos en la base de datos mediante un desplegable en el menú principal.
- **Mostrar ubicación de un usuario**: muestra en el mapa la ubicación del usuario seleccionado, siempre y cuando se disponga de información de localización.
- **Desplazar mapa**: desplazar el mapa en el sentido pulsado por el usuario.
- **Hacer *zoom***: acerca o alejar el nivel de *zoom* del mapa.
- **Seleccionar interacción con usuario**: elegir un usuario, ubicándolo en el mapa y haciéndole objeto de las diferentes acciones.
- **Llamar a contacto**: realizar una llamada telefónica al contacto actual.
- **Enviar correo electrónico a contacto**: enviar un correo electrónico al contacto actual.
- **Cambiar estado**: cambiar estado propio como técnico y consultar los estados del resto.
- **Ver información de perfil de contacto**: visualizar la fecha y hora de la última vez que el contacto actualizó su estado.

Para que se den los casos de uso indicados, primero *SATApp* solicita al servidor la localización de sus usuarios y utiliza el nombre de usuario de cada uno como clave identificativa. Esto es así para que, en el hipotético caso de que un usuario cambie de número de teléfono o de terminal, pueda seguir usando sus datos mediante el mismo nombre de usuario.

Así mismo, la aplicación muestra el nombre de cada contacto junto con su ubicación en el mapa, y utiliza tanto el teléfono como la dirección de correo electrónico para realizar una llamada o enviar un correo si lo cree oportuno. Por tanto, toda esta información se obtiene de los contactos almacenados con anterioridad en el dispositivo móvil.

Siguiendo el mismo criterio, el siguiente diagrama expresa los casos de uso asociados al servidor al que se conecta la aplicación para gestionar la información de localización:

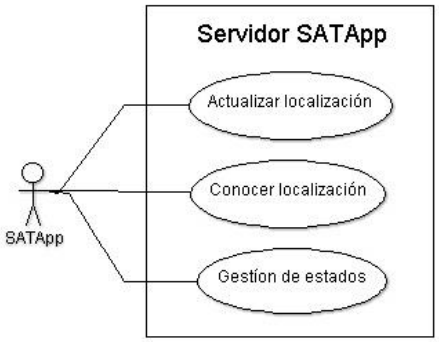


Figura 8. Diagrama de casos de uso de SATApp.

En la figura anterior, el único actor presente es la propia aplicación *SATApp* que establece la conexión con el servidor para el intercambio de información. Los casos de uso considerados son los siguientes:

- **Actualizar localización.** Actualizar los datos de localización del usuario.
- **Conocer localización.** Obtener los datos de localización del usuario solicitado.

Es importante recalcar que los casos de uso presentados simplemente expresan el punto de vista del usuario sobre cómo debe funcionar la aplicación y qué se puede realizar a través de ella. Son una forma de facilitar su comprensión y, por tanto, no tiene por qué existir ninguna correspondencia entre los casos de uso y las clases finalmente implementadas, más allá de que las clases en su conjunto, como sistema completo, realizan aquello que los casos de uso expresan.

Análisis del perfil administrador

El usuario que utiliza la aplicación como administrador desde la central necesitará realizar las siguientes tareas:

- **Mostrar ubicación GPS propia.** Podrá consultar su ubicación GPS mediante Google Maps pulsando sobre la opción dedicada a tal efecto.
- **Mostrar ubicación GPS de los técnicos.** Podrá consultar la ubicación GPS de cada técnico de la misma manera que en el punto anterior.
- **Comunicación con un técnico.** Podrá enviar un correo electrónico o incluso realizar una llamada a un técnico en concreto para asignar un servicio o proporcionar cualquier información de importancia. Esta opción será posible pulsando sobre el desplegable que contiene los nombres de usuario de los técnicos y tras esto, sobre la opción de interacción deseada que se encuentra en forma de botón en la pantalla principal.
- **Gestión de datos.** Podrá gestionar cualquier dato correspondiente a los correos electrónicos intercambiados con los técnicos. Asimismo, podrá comprobar los estados de los técnicos en tiempo real.

Análisis del perfil técnico

- **Mostrar ubicación GPS propia.** Podrá consultar su ubicación GPS mediante Google Maps pulsando sobre la opción dedicada a tal efecto.
- **Mostrar ubicación GPS del resto de los técnicos.** Podrá consultar la ubicación GPS de cada técnico de la misma manera que en el punto anterior.
- **Comunicación con administrador y técnicos.** Podrá realizar una llamada o un correo electrónico a otro técnico o al mismo administrador desde la información de ubicación en el mapa de Google Maps. Ello será posible mediante un icono situado de manera intuitiva y una vez seleccionado el técnico en el desplegable.
- **Gestión de estados de reparación.** Cada técnico podrá cambiar su estado para informar del proceso en el que se encuentra en ese momento. A su vez, podrá comprobar los estados del resto de los técnicos y la fecha en la que se hizo el último cambio de estado.

- **Envío de informe de reparación.** Tras realizar una reparación, el técnico enviará un correo electrónico al administrador con los detalles de dicha reparación con el fin de llevar un control de los clientes y realizar cobros o gestiones administrativas.

5.1.3. Diseño conceptual.

El siguiente diagrama muestra las relaciones presentes entre las clases existentes y la comunicación que existen entre ellas. Con ello, se pretende esclarecer un poco más el funcionamiento de la aplicación *SATApp* aunque también se presentará un modelo de clases más completo en el apéndice del proyecto.

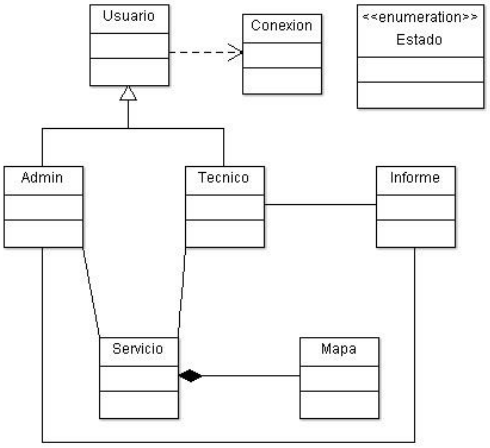


Figura 9. Diagrama de clases simplificado de la aplicación *SATApp*.

5.1.3.1. Escenarios de uso.

A continuación, se detallan una serie de posibles escenarios de uso de la aplicación a desarrollar. Recordamos que la versión beta estudiada en este punto puede sufrir cambios, tanto en su interfaz como en sus funcionalidades, por motivos de usabilidad, visibilidad o eficacia.

Escenario 1

Usuario:	Pedro.
Contexto:	Registro de usuario para realizar labores de administrador de la aplicación.
Objetivo:	Comprobar la visibilidad, rapidez y usabilidad de la aplicación
Tareas:	Introducir datos personales para utilizar la aplicación por primera vez.
Necesidad de Información:	Inicio de sesión.
Funcionalidades:	Insertar información en base de datos e iniciar sesión.
Desarrollo de tareas:	Añadir información de registro.

Escenario 2

Usuario:	Alfonso.
Contexto:	Búsqueda de un técnico en el mapa.
Objetivo:	Localizar a un técnico usando el mapa para ofrecer apoyo según distancia entre ambos o compartir información.

Tareas:	Buscar el contacto del técnico en cuestión.
Necesidad de Información:	Localización del técnico.
Funcionalidades:	Localización GPS, cobertura móvil y gestión de contactos en la aplicación.
Desarrollo de tareas:	Desde el menú principal y ya dentro de Google Maps, el usuario busca al contacto y su localización aparecerá en el mapa.

Escenario 3

Usuario:	Damián.
Contexto:	Interacción con un usuario (llamada).
Objetivo:	Intercambiar información con otro técnico o interactuar con el administrador.
Tareas:	Pinchar sobre el contacto localizado en el mapa y pulsar sobre la opción de llamar.
Necesidad de Información:	Intercambiar información con otro usuario a nivel organizativo.
Funcionalidades:	Localizar al técnico en el mapa y realizar una llamada al mismo.
Desarrollo de tareas:	Buscar al usuario en la lista de contactos, localizarlo en el mapa y pinchando sobre él, pulsar sobre la opción de llamada.

Escenario 4

Usuario:	Maribel.
Contexto:	Envío y recepción de mensajes o correos electrónicos.
Objetivo:	Entablar una comunicación con el resto de usuarios de manera instantánea y aportar información escrita.
Tareas:	Enviar y recibir mensajes y correos electrónicos e intercambiar información o realizar gestiones administrativas.
Necesidad de Información:	Intercambio de información y asignación de servicios.
Funcionalidades:	Comunicación entre usuarios.
Desarrollo de tareas:	Localizar a un usuario en el mapa y enviar un mensaje o un correo electrónico mediante la opción que aparece al pulsar sobre el contacto.

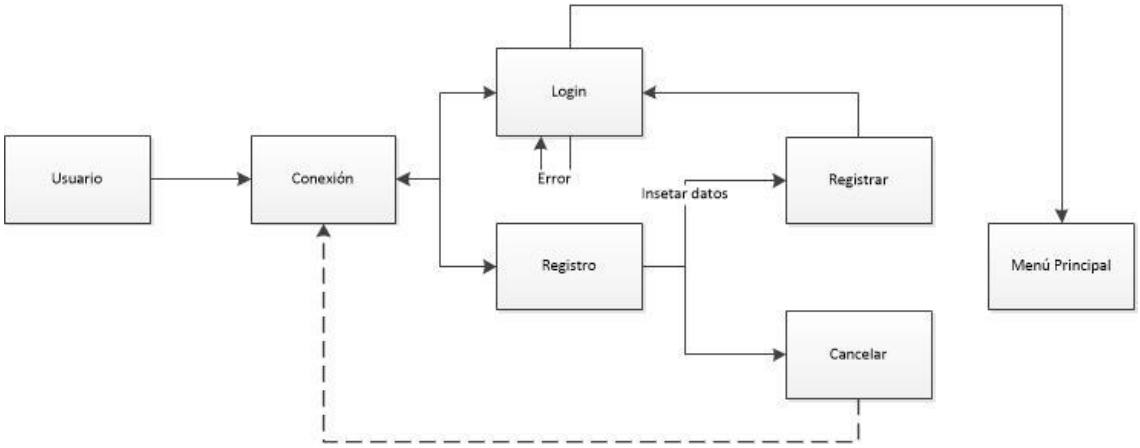
Escenario 5

Usuario:	Carmen.
Contexto:	Consulta del estado del resto de usuarios y cambio del estado propio.
Objetivo:	Facilitar el estado de cada usuario a tiempo real.
Tareas:	Visualizar el estado de otro usuario e incluir el propio.
Necesidad de Información:	Conocimiento del estado de los empleados y la actividad que realizan a tiempo real.
Funcionalidades:	Conocer el estado de todos los miembros de la empresa.
Desarrollo de tareas:	Localizar a un usuario en el mapa y mediante el desplegable de opciones, comprobar su estado y facilitar el estado propio.

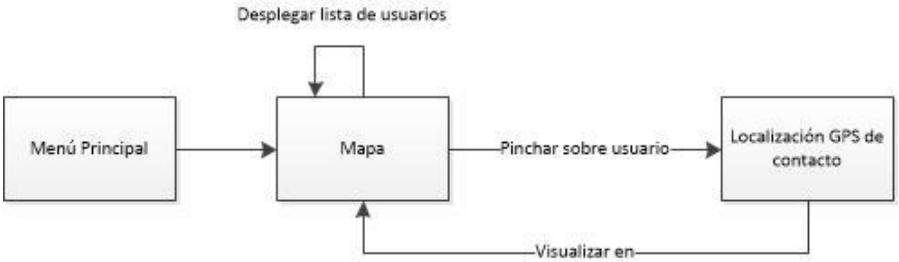
5.1.3.2. Flujos de interacción.

A continuación se presentan los flujos de interacción de las tareas principales que realiza la aplicación *SATApp*.

Flujo de interacción 1 – Registro de usuario

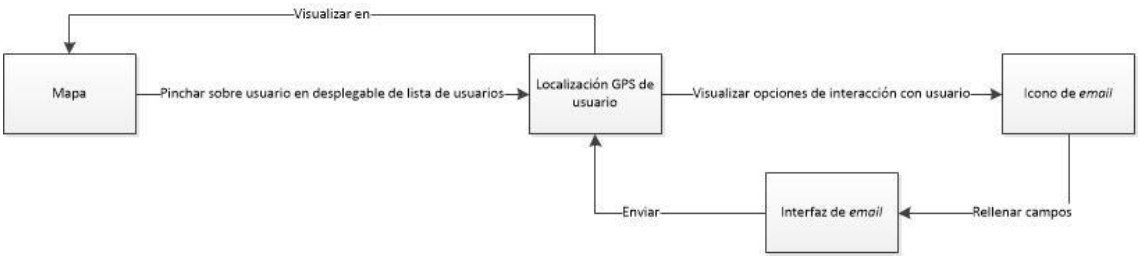


Flujo de interacción 2 – Buscar a usuario en el mapa

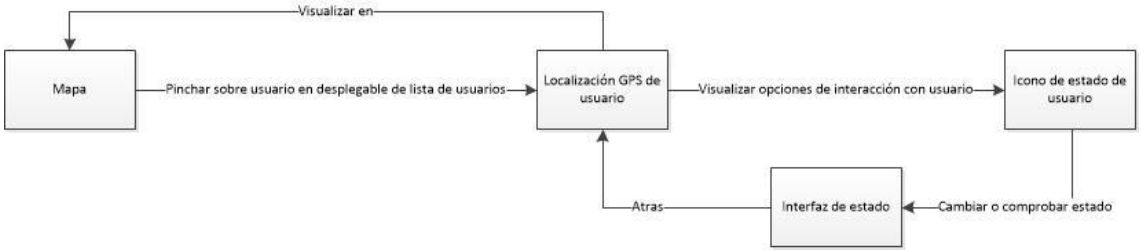


Flujo de interacción 3 – Llamar a un usuario

Flujo de interacción 4 – Interacción con un usuario mediante correo electrónico



Flujo de interacción 5 – Establecer y comprobar estado de usuario



5.1.4. Prototipado.

Prototipo horizontal de alta fidelidad

Con el apoyo obtenido mediante el análisis de usuarios, los contextos de uso y el diseño conceptual, se presenta a continuación el siguiente prototipo, donde se aprecia de forma horizontal el funcionamiento de la aplicación en sus diferentes pantallas.

Pantalla inicio de sesión

Este es el diseño de la pantalla de inicio de sesión. En este, el usuario introduce su *login* y contraseña para acceder a la aplicación, o bien pulsa sobre el botón Registro si aún no está registrado.

Se ha seleccionado un fondo de un azul característico y amigable. A su vez, se ha diseñado una imagen llamativa e intuitiva como *splash* de la aplicación. Los botones tienen un color algo más oscuro para destacar sobre el fondo.

El texto ha sido desarrollado mediante el tipo de letra *MS Sans Serif* y el color blanco otorga claridad y visibilidad sobre el fondo azul. La opción de registro está separada para destacar ante un primer uso.

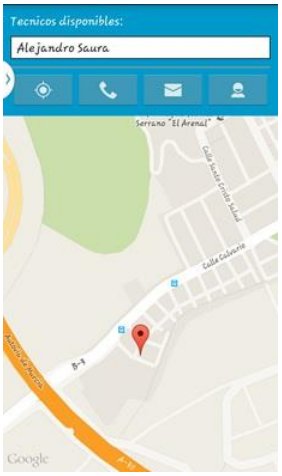
Pantalla registro de usuario

En este punto, el usuario introduce sus datos para quedar registrado en el sistema.

Permanece el color característico, el diseño y el formato del texto en todo momento.

Para realizar el registro adecuadamente, el usuario solo necesita un *login*, contraseña, nombre, teléfono y correo electrónico. Tras añadir estos datos, el usuario tan solo debe pulsar sobre el botón Registrar y pasa de nuevo al menú de inicio de sesión, desde donde puede acceder al menú principal de la aplicación *SATApp*.

Pantalla menú principal



Como se ha mencionado, tras registrarse, el usuario ya puede acceder mediante autentificación. Este accede al menú principal donde el diseño sigue los mismos patrones característicos.

La aplicación abre con el servicio Google Maps de fondo y en la zona superior de la pantalla se aprecia un desplegable de los técnicos registrados y disponibles en *SATApp* y cuatro botones que aportan las funcionalidades sobre ellos:

- Actualizar la posición.
- Realizar una llamada al técnico seleccionado.
- Enviar un correo electrónico a dicho técnico.
- Comprobar y cambiar el estado de reparación.

Pantalla de llamada a usuario

En este punto, se ha seleccionado un usuario del desplegable creado para tal efecto, su ubicación aparece señalada en el mapa y pulsamos sobre el botón de llamada situado en la zona superior de la pantalla.

Como podemos apreciar en la imagen adjunta a esta funcionalidad, la aplicación abre la opción de llamada a ese usuario.

Al finalizar la llamada, el usuario puede volver a la aplicación sin problemas.

Envío de *emails*

Para llegar a esta funcionalidad, se ha seleccionado un usuario del desplegable donde se sitúan los usuarios registrados, su ubicación aparece señalada en el mapa y pulsamos sobre el botón de envío de *emails* situado en la zona superior de la pantalla.

Como podemos apreciar en la imagen adjunta a esta funcionalidad, la aplicación abre la opción de envío de email al técnico.

En ella aparece el nombre del técnico y su correo electrónico de manera predeterminada.

Tan solo es necesario incorporar un asunto y el mensaje que se desea transmitir (nombre del cliente, número de teléfono, avería o servicio, dirección y observaciones de ser necesarias).

Estado de usuario

Por último, en la misma posición del menú principal, el usuario dispone de un botón para gestionar el estado de reparación.

Cada usuario puede cambiar su propio estado mediante esta funcionalidad y visualizar al mismo tiempo el estado del resto de miembros que usan la aplicación.

Además, se aprecia la fecha de última modificación como detalle característico y útil para el administrador.

El diseño permanece inalterable y es propio de la aplicación *SATApp*.

5.1.5. Evaluación.

Como se ha comentado anteriormente y ya es bien sabido, el proceso de DCU es iterativo y redundante, y como consecuencia de esto, se ha de evaluar y reevaluar los diseños corrigiéndolos y perfeccionándolos.

A continuación, se procederá a definir las preguntas que ayudarán a conocer las opiniones de los usuarios respecto al diseño centrado en el usuario de la aplicación *SATApp*.

5.1.5.1. Recopilación de preguntas a usuarios.

A los usuarios de *SATApp* se les ha realizado el siguiente test con el fin de determinar la evaluación de la aplicación:

- ¿Qué dificultades ha encontrado para completar las tareas?
- ¿De qué elementos carece el prototipo actual?
- ¿Qué defectos ha detectado en el uso del prototipo?
- ¿La iconografía es comprensible?
- ¿Este prototipo cumple con sus expectativas?
- ¿Qué es lo más destacable de la aplicación? ¿Y lo menos?
- ¿Cree que son de utilidad todas las tareas?
- ¿Cómo mejoraría la aplicación *SATApp*?
- ¿Cómo valoraría la visibilidad de la aplicación *SATApp*? ¿Y su usabilidad?
- ¿Cree que la velocidad de acceso y navegación es aceptable?
- ¿Utilizaría esta solución en el futuro (por favor explique por qué o por qué no)?

5.1.5.2. Tareas a realizar por los usuarios.

A los usuarios de la aplicación SATApp se les requerirá realizar la siguiente lista de tareas:

- Realizar registro de usuario e iniciar sesión como tal.
- Realizar una búsqueda de un usuario mediante su número de contacto.
- Geolocalizar a un usuario.
- Comunicarse con un usuario mediante mensaje, llamada o correo electrónico.
- Cambiar el estado propio de reparación.

5.1.5.3. Preguntas referentes a las tareas.

A los usuarios de SATApp se les realizarán las siguientes preguntas respecto a las tareas realizadas:

- ¿Ha encontrado dificultades o trabas para completar las tareas con éxito?
- ¿De qué elementos cree que carece el prototipo actual?
- ¿Ha encontrado algún fallo o defecto mientras usaba el prototipo?
- ¿Cree que podría mejorarse la herramienta?
- ¿Cree que la aplicación es intuitiva?
- ¿Cumple con sus expectativas?

Respecto a cada una de las tareas nombradas en el punto anterior, se les realizarán las siguientes preguntas en concreto:

- ¿Ha realizado la tarea con éxito?
- ¿Le ha resultado sencillo e intuitivo?
- ¿Cree que es de utilidad y la usará normalmente?
- ¿Incluiría algún aspecto de importancia a esta tarea?

5.1.5.4. Aspectos susceptibles de mejoras

En general, la aplicación *SATApp* dispone de unas funcionalidades útiles y atractivas. Asimismo, se asume que la interfaz, usabilidad y visualización desarrolladas hasta ahora satisfacen en cierta medida las pretensiones de futuros usuarios.

Aun así, se establecen una serie de aspectos que pueden ser mejorados con tiempo y los conocimientos necesarios. Se nombran algunos de ellos a continuación:

- Agenda con citas, eventos y planificación de reparaciones.
- Sistema de notificaciones.
- Base de datos de clientes.
- Gestión de contabilidad y facturación.

Teniendo en cuenta la retroalimentación permitida por el DCU, se abordarán estas funcionalidades más adelante y se valorarán por el bien de una aplicación más profesional. Por tanto, *SATApp* es susceptible de mejoras y se tienen en cuenta para ser estudiadas en el momento de su implementación o en adelante.

5.2. Desarrollo e implementación.

Una vez expuestos los objetivos y características principales de la aplicación, así como las decisiones tomadas en cuanto a su diseño inicial, se procede a desgranar en este apartado los aspectos relacionados con su implementación.

En las siguientes líneas se busca no sólo mostrar el código más relevante de la aplicación *SATApp*, sino ofrecer también una explicación general sobre cómo funcionan las aplicaciones para Android, de forma que a su vez se consiga mostrar las peculiaridades de este nuevo sistema. Así mismo, se espera que los detalles de implementación mencionados en este proyecto puedan ayudar a otros desarrolladores a ampliar las capacidades y límites de *SATApp*, o incluso inspirar la creación de aplicaciones con servicios similares.

Se advierte al lector de que los fragmentos de código fuente mostrados a partir de ahora no son una copia literal del código de *SATApp*, sino que en algunos casos se reducen o modifican por motivos de limitación de espacio, pero sobre todo por simplificar y facilitar su comprensión.

5.2.1. Inicio de sesión.

La aplicación SATApp incluye una pantalla *splash* que se muestra durante unos pocos segundos al inicio. Con esto se pretende ofrecer un estilo propio y dar sensación de carga de la aplicación.

Tras esto, el sistema invita a introducir usuario y contraseña para iniciar sesión o realizar el registro de nuevo usuario como hemos podido comprobar en la pantalla de inicio de sesión correspondiente al prototipo de alta definición. La clase encargada de ello se denomina *Login*.java.

Este inicio de sesión ha sido desarrollado usando un servidor Web gratuito en el que se realizan peticiones de datos por POST hacia unos documentos en lenguaje PHP que conectan con una base de datos MySQL.

A continuación, se facilita el código correspondiente a la respuesta ofrecida por *SATApp* cuando realiza el inicio de sesión.

```
public void onPost(JSONObject respuesta) throws Exception {
    boolean error;
    // json success tag
    error = respuesta == null || respuesta.has(JSONClient.TAG_ERROR);
    if (error) {
        Log.d("Login Failure!", "Error al iniciar sesion");
        pDialog.dismiss();
        Toast.makeText(Login.this, "Error al iniciar sesion", Toast.LENGTH_LONG).show();
    } else {
        Log.d("Login Successful!", respuesta.toString());
        // save user data
        SharedPreferences sp = PreferenceManager.getDefaultSharedPreferences(Login.this);
        Editor edit = sp.edit();
        edit.putString("username", username);
        edit.commit();
        Intent map = new Intent().setClass(Login.this, MapsActivity.class);
        startActivity(map);
        pDialog.dismiss();
        finish();
    }
}
```

Figura10. Obtención de respuesta en inicio de sesión.

Para usuarios no registrados en la base de datos, estos deberán pulsar sobre la opción de registro e introducir su nombre de usuario, contraseña, nombre, correo electrónico y número de teléfono asociado. El número es necesario puesto que existe un error con las SIM's europeas que no permite el acceso al número de teléfono de un terminal y, por tanto, este paso será imprescindible para el buen funcionamiento de la aplicación. Para aprovechar este inconveniente, tomaremos en cuenta que de este modo, el administrador puede crear un nuevo usuario desde su propio terminal y ello se convierte en una funcionalidad añadida que la empresa puede aprovechar.

5.2.2. Acceso a la base de datos de usuarios.

Para cada usuario, la aplicación necesita conocer su nombre, número de teléfono, su dirección de correo electrónico, sus datos de localización, es decir, latitud y longitud, y la fecha en la que fue actualizada su ubicación. Toda esta información se obtiene desde el servidor y la base de datos anteriormente creada.

Esta es accesible mediante la clase *JSONClient* que se utiliza para realizar una petición POST a partir de unos datos JSON. El intercambio de información se puede apreciar en el código de la siguiente figura y mediante el mismo se obtiene acceso a los datos de los usuarios.

```
protected void onPostExecute(JSONObject result) {
    if(listener != null){
        try{
            //al terminar la tarea, llama al listener con el resultado.
            listener.onPost(result);
        }catch(Exception e){
            e.printStackTrace();
            Log.e("Utilidades", "Error al tratar la respuesta del servicio.");
        }
    }
};

protected void onCancelled() { onCancelled(null); };

protected void onCancelled(JSONObject result) {
    if(listener != null){
        try{
            //al terminar la tarea, llama al listener con el resultado.
            listener.onPost(result);
        }catch(Exception e){
            e.printStackTrace();
            Log.e("Utilidades", "Error al tratar la respuesta del servicio.");
        }
    }
};
```

Figura 11. Conexión y desconexión en torno a las respuestas del Servido Web.

5.2.3. Acceso a Google Maps.

La utilización del popular servicio Google Maps es una de las posibilidades más atractivas de Android. Hoy en día, un gran número de las aplicaciones presentadas al concurso de desarrolladores propuesto por Google utilizan estas bibliotecas con fines muy distintos.

El paquete que incluye todas las clases relacionadas con la carga y manejo de mapas directamente desde Google Maps es *com.google.android.gms.maps*. Dentro de este paquete se encuentra la clase Google Maps y a partir de ella se ha creado una instancia denominada *mMap* sobre la cual trabajaremos.

Un paso previo a la utilización de este API para mostrar y manejar mapas es el registro en Google Maps y su aceptación de los términos y condiciones de uso. Este registro se realiza a través de una clave denominada en Android como *API key*.

Una vez se ha procedido al registro, la utilización de mapas en nuestra aplicación no requiere más que unas cuantas llamadas a las clases pertinentes. Todo ello se puede comprobar en los dos siguientes apartados.

<u>Método de obtención de una API Key</u>

Tal y como se ha mencionado, para utilizar los servicios de Google Maps en una aplicación Android es imprescindible realizar unos pasos previos que implican el registro del desarrollador y la aceptación de las condiciones de uso. De esta forma, Google asegura que se hará en todo momento un uso adecuado y apropiado de los servicios y datos que nos va proporcionar desde entonces con Google Maps.

Toda aplicación en Android está acompañada de un certificado que asegura su autoría y la vincula con su desarrollador. Si se utiliza el *plug-in* de Android para Eclipse, que no es el caso, todas las aplicaciones están firmadas por un certificado al que se puede calificar de prueba y que permiten a los desarrolladores poder crear y probar sus aplicaciones sin más esperas. Este certificado se obtiene a través del fichero de claves de prueba "*debug.keystore*", presente por defecto en el SDK de Android.

Recuérdese, sin embargo, que si un desarrollador desea distribuir su aplicación de forma pública o hacerla accesible desde servicios de descarga, como por ejemplo Android Market, es necesario que se cree su propio fichero e individual de claves "*.keystore*" con el que crear un certificado y firmar su aplicación.

El registro para tener acceso a Google Maps se hace a través de dicho certificado. En concreto el proceso es el siguiente, utilizando el fichero "*debug.store*". En caso de disponer un fichero de claves propio, el proceso sería el mismo cambiando los parámetros necesarios.

- En primer lugar, se ha de obtener el certificado SHA-1 para *debug* que se necesita para obtener la API Key. Este certificado puede generarse, por ejemplo, con la herramienta *keytool* presente en el SDK de Java.
  La llamada correspondiente a su obtención se muestra en la figura 15.

```
Microsoft Windows [Versión 6.3.9600]
(c) 2013 Microsoft Corporation. Todos los derechos reservados.

C:\Windows\system32>keytool -list -v -keystore "C:\Users\Alejandro\.android\debu
g.keystore" -alias androiddebugkey -storepass android -keypass android
Nombre de Alias: androiddebugkey
Fecha de Creación: 08-may-2015
Tipo de Entrada: PrivateKeyEntry
Longitud de la Cadena de Certificado: 1
Certificado[1]:
Propietario: CN=Android Debug, O=Android, C=US
Emisor: CN=Android Debug, O=Android, C=US
Número de serie: 49be4d01
Válido desde: Fri May 08 16:11:33 CEST 2015 hasta: Sun Apr 30 16:11:33 CEST 2045

Huellas digitales del Certificado:
         MD5: 24:73:72:5F:4E:56:6E:F8:33:40:BF:8C:5F:4D:76:6E
         SHA1: 86:CD:53:94:3F:9E:13:48:72:68:E9:E5:16:BD:E4:5E:CC:BF:31:28
         SHA256: 90:72:44:20:DC:35:19:07:4B:DD:A6:12:CB:5C:1E:78:F0:49:7D:12:73:
97:2E:94:5A:86:5C:9E:CD:17:C8:DB
         Nombre del Algoritmo de Firma: SHA256withRSA
         Versión: 3

Extensiones:

#1: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: C2 CC EB 3A 06 01 D0 F6   A6 48 31 E4 68 82 62 2A  ...:.....H1.h.b*
0010: 37 A1 36 0B                                        7.6.
]
]
```

Figura 12a. Ejemplo de obtención de una API Key proporcionada por Google.

- Ya con este *SHA1 Fingerprint Certificate*, se procede a solicitar en el Google API Console la respectiva API Key. La dirección en la que se encuentra esta opción es la siguiente: https://code.google.com/apis/console/. Es necesario tener creada una cuenta de Google personal o corporativa para acceder a este servicio.
- Una vez se ha accedido a la *Google API Console*, si es la primera vez, pide la creación de un nuevo proyecto al que nombraremos "API Project".
- Con el nuevo proyecto creado, se aprecia un listado de servicios a los cuales se tiene acceso mediante la opción *APIs y Autenticación > APIs*. Ahí es donde aparece *Google Maps Android API v2*, que debe habilitarse pulsando sobre *Habilitar API*.
- Solo es necesario aceptar los términos y condiciones de uso de este servicio y pulsar sobre la opción *Ver informes en la consola de la API*.
- Por último, haciendo *click* en la sección *API Access*, se crea una *API Key* para Android, empleando el boton *Create new Android Key*. Para crear una nueva Android Key es necesario el *SHA1 Certificate Fingerprint* obtenido en la figura anterior, ya que es un requerimiento de seguridad de Google para controlar posibles abusos de la plataforma.
- Con el *SHA1 Certificate Fingerprint* y el nombre del paquete java usado en el proyecto (en este caso la aplicación *SATApp* está creada bajo el paquete satapp.tfc.proyecto.satapp), basta con insertar estos datos tal y como se muestra en la siguiente figura.
- Al hacer *click* sobre el botón *Create*, Google genera un *API Key* de 40 caracteres alfanuméricos para poder hacer uso de su servicio de Mapas.

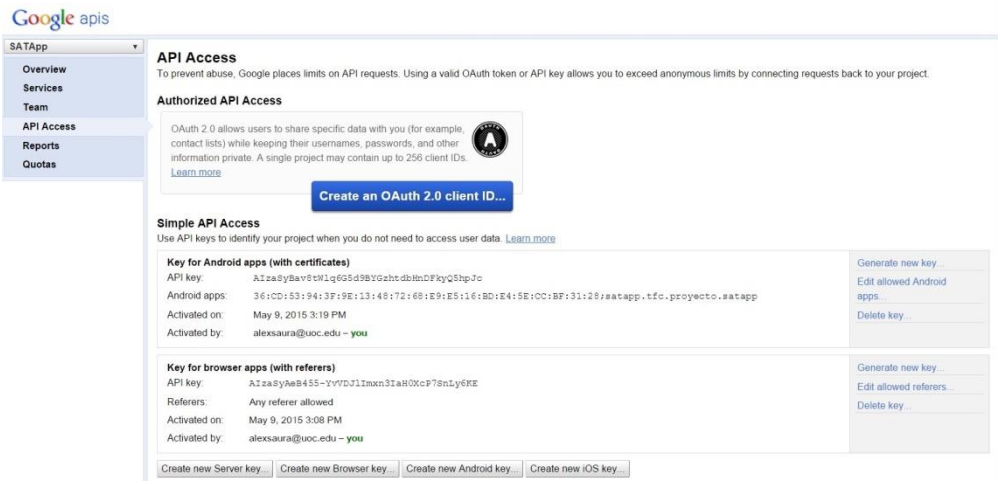


Figura 12b. Ejemplo de obtención de una API Key proporcionada por Google.

Mostrar un mapa en el menú principal

Ahora que se dispone de una *API key* con la que poder acceder a los mapas y servicios de Google Maps y mediante la llamada *Intent* contenida en la clase *Login*.java, no resta más que hacer las llamadas pertinentes a las clases del paquete *com.google.android.gms.maps*.

La clase *MapsActivity* es la clase principal y la que coordina la parte más importante, el acceso a los mapas de Google. Esta clase deriva de la clase *Activity*, como ya se advirtió, pero lo hace de forma indirecta. A quien extiende en realidad es a la clase *FragmentActivity*, que incluye las funciones estándares de *Activity* más aquellas otras relacionadas con la visión de mapas en la API de Google.

Para poder visualizar un mapa obtenido desde Google Maps, es necesario realizar la llamada al siguiente método:

```
private void setUpMapIfNeeded() {
    // Do a null check to confirm that we have not already instantiated the map.
    if (mMap == null) {
        // Try to obtain the map from the SupportMapFragment.
        mMap = ((SupportMapFragment) getSupportFragmentManager().findFragmentById(R.id.map)).getMap();
        // Check if we were successful in obtaining the map.
        if (mMap != null) {
            mMap.setMapType(GoogleMap.MAP_TYPE_NORMAL);
        }
    }
}
```

Figura 13. Método que muestra el mapa dentro de la clase *MapActivity*.

5.2.4. Control del mapa.

Android cuenta con un paquete muy completo para la composición de elementos de interfaz, también llamado *widgets*, como botones, imágenes, cajas de texto, etc. Este paquete, denominado *android.widget* contiene clases como las siguientes:

- Button: representa un botón que puede ser pulsado por el usuario.
- EditText: clase que muestra una caja de texto donde el usuario puede escribir.
- FrameLayout: clase que representa un área de la pantalla donde colgar otros elementos visuales.
- ImageButton: representa un botón al que se le puede asociar una imagen.
- ProgressBar: muestra una barra de progreso.
- TextView: clase que ofrece un área de texto no editable; por ejemplo, títulos para campos de formularios.
- ZoomControls: ofrece elementos de control de zoom.

En *SATApp*, la interfaz mostrada no incluye solamente el mapa con los usuarios dibujados en él sino que también se incluyen ciertos widgets a través de los cuales el usuario pueda realizar algunas de las acciones contempladas en la aplicación.

Una de estas acciones es el uso de un *spinner* que muestra una lista desplegable para seleccionar un único elemento y es equivalente al típico *select* de *html* o los *ComboBox* de otros entornos de desarrollo. Este sirve para desplegar los usuarios disponibles en la aplicación y obtener su localización.

```
private Spinner spinnerUsuario;
private Intent locationServiceIntent;
private static Marker usuarioSeleccionadoMarker;
private static final int mapZoom = 16;

public static MapsActivity currentActivity;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.maps_activity);
    currentActivity = this;
    setUpMapIfNeeded();

    localizar = (ImageButton)findViewById(R.id.accion_localizar);
    localizar.setOnClickListener(this);
    llamar = (ImageButton)findViewById(R.id.accion_llamar);
    llamar.setOnClickListener(this);
    email = (ImageButton)findViewById(R.id.accion_email);
    email.setOnClickListener(this);
    estado = (ImageButton)findViewById(R.id.accion_estado);
    estado.setOnClickListener(this);
    spinnerUsuario = (Spinner)findViewById(R.id.listadoUsuarios);
    spinnerUsuario.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {

        @Override
        public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
            try {
                usuarioSeleccionado = datosUsuarios.getJSONObject(position);
                obtenerPosicion(usuarioSeleccionado.getString("username"));
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
```

Figura 14. Uso de un widget del tipo *spinner* en aplicación *SATApp*.

No he creído conveniente el uso de controles de zoom puesto que el proceso de localización se centra en la pantalla y esta permite libertad de movimientos.

5.2.5. Control de la señal GPS.

SATApp utiliza la señal GPS para conocer la ubicación actual del usuario, pudiendo así tanto dibujarla en el mapa mostrado como comunicársela a los demás usuarios a través de su actualización en el servidor Web.

Conocer la ubicación propia

De manera predeterminada, *SATApp* ofrece la posición del usuario que inicia sesión gracias al método *ObtenerPosicion(final String username)*. Esta queda reflejada mediante un *marker* sobre el mapa que apunta a la latitud y longitud determinadas. Se puede ver como el método *onCreate()* llama a este método en la figura anterior y a continuación se facilita el código correspondiente a la implementación de este proceso en la siguiente figura.

```
private void obtenerPosicion(final String username){
    if(pDialog != null && pDialog.isShowing()){
        pDialog.dismiss();
    }
    try {
        JSONObject datos = new JSONObject();
        datos.put("username", username);

        pDialog = new ProgressDialog(MapsActivity.this);
        pDialog.setMessage("Obtaining " + username + " position...");
        pDialog.setIndeterminate(false);
        pDialog.setCancelable(true);
        pDialog.show();
        Log.d("request!", "starting");
        // getting product details by making HTTP request
        JSONClient.post(datos, JSONClient.serverBaseUrl + "/obtenerPosicion.php", (respuesta) -> {
                boolean error;
                error = respuesta == null || respuesta.has(JSONClient.TAG_ERROR);
                if (error) {
                    Log.d("Get position Failure!", "Error obteniendo la posicion del usuario");
                    pDialog.dismiss();
                    Toast.makeText(MapsActivity.this, "Error obteniendo la posicion del usuario", Toast.LENGTH_LONG).show();
                } else {
                    Log.d("Get position success!", "ok");
                    JSONArray resultadoPosicion = respuesta.getJSONArray("posicion");
                    JSONObject posicion = resultadoPosicion.getJSONObject(0);
                    double latitud = posicion.getDouble("latitud");
                    double longitud = posicion.getDouble("longitud");

                    if (usuarioSeleccionadoMarker != null) {
                        usuarioSeleccionadoMarker.remove();
                    }
                    usuarioSeleccionadoMarker = mMap.addMarker(new MarkerOptions().position(new LatLng(latitud, longitud)).title(username));
                    CameraUpdate cameraUpdate = CameraUpdateFactory.newLatLngZoom(new LatLng(latitud + 90d / Math.pow(2, mapZoom), longitud), mapZoom);
                    mMap.animateCamera(cameraUpdate);
                    pDialog.dismiss();
                }
        });
    }catch (Exception e){
        e.printStackTrace();
    }
}
```

Figura 15. Obtención de localización de usuario en el mapa.

Además de esto, es necesario otorgar a la aplicación el permiso correspondiente para acceder al dispositivo de localización mediante la declaración en el fichero *AndroidManifest.xml* de la siguiente etiqueta:

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

Figura 16. Declaración en el manifiesto del permiso de acceso al GPS.

Actualizar los datos de ubicación

Una vez se ha obtenido la ubicación propia, esta se actualiza con el servidor de la forma que se muestra a continuación.

```
private void startSendPositionUpdates(String numeroUsuarioLogeado){
    if(locationServiceIntent != null){
        Toast.makeText(this, "Error al iniciar el servicio: El servicio de geoposicionamiento ya esta funcionando", Toast.LENGTH_LONG).show();
        return;
    }
    locationServiceIntent = new Intent(this, LocationService.class);
    locationServiceIntent.putExtra("numero", numeroUsuarioLogeado);
    startService(locationServiceIntent);
}

private void stopSendPositionUpdates(){
    if(locationServiceIntent != null) {
        stopService(locationServiceIntent);
        locationServiceIntent = null;
    }else{
        Toast.makeText(this, "Error al detener el servicio: El servicio de geoposicionamiento no esta funcionando", Toast.LENGTH_LONG).show();
    }
}
```

Figura 17. Actualización de los datos de localización con el servidor.

Este es el resultado de la ubicación propia de un usuario logueado en la aplicación *SATApp*:

Geolocalización optima de usuario

La clase *Geolocation* es una utilidad para obtener la localización de los usuarios y guarda la última localización conocida. Esto es muy útil teniendo en cuenta posibles fallos de conexión que no permitan al usuario ofrecer su ubicación a tiempo real. Mediante esta clase, se obtiene la posición del usuario mediante el servicio GPS o la Red y devuelve la mejor de ellas gracias al método *getDevideLocation()* como se muestra a continuación.

```
public Double[] getDeviceLocation(){
    Double[] deviceLocation = null;
    if (!gps_enabled && !network_enabled) {
        Toast.makeText(context, "No coordinate location provider found. Please, activate GPS or location from network.", Toast.LENGTH_LONG).show();
    }else{
        Location location = null;
        if(network_enabled){
            Location networkLocation = locationManager.getLastKnownLocation(LocationManager.NETWORK_PROVIDER);
            if(isBetterLocation(networkLocation, lastNetWorkLocation))
                lastNetWorkLocation = networkLocation;
            location = lastNetWorkLocation;
        }
        if(gps_enabled){
            Location gpsLocation = locationManager.getLastKnownLocation(LocationManager.GPS_PROVIDER);
            if(isBetterLocation(gpsLocation, lastGpsLocation))
                lastGpsLocation = gpsLocation;
            if(isBetterLocation(lastGpsLocation, location))
                location = lastGpsLocation;
        }

        if(location!=null)
            deviceLocation = new Double[]{/*X*/location.getLatitude(), /*Y*/location.getLongitude()};
        else {
            MapsActivity.currentActivity.runOnUiThread(() -> {
                    Toast.makeText(MapsActivity.currentActivity, "No coordinate location found. Please, activate GPS or location from network.", Toast.LENGTH_LONG).show();
            });

            deviceLocation = new Double[]{0d, 0d};
        }
    }
    return deviceLocation;
}
```

Figura 18. Implementación del método getDevideLocation() de la clase Geolocation.

5.2.6. Conexiones HTTP.

SATApp conecta con el servidor de forma periódica, más concretamente, cada dos minutos, para actualizar la localización del usuario y para conocer las localizaciones de sus usuarios. Esta comunicación se realiza a través de Internet con conexiones HTTP y en las peticiones se utiliza el método POST.

Android cuenta con numerosos paquetes y clases dedicados en exclusiva a los diferentes aspectos de las comunicaciones. Para estos flujos de información se utiliza el paquete *org.apache.http*.

A continuación, se crea el cliente HTTP con la clase *HttpClient*. Ésta cuenta con el método *execute()*, que precisa como parámetros la petición a enviar y se encarga de ejecutar la conexión. Para recibir la correspondiente respuesta, se define un objeto *HttpResponse* que recibe el resultado de *execute()*.

Se puede apreciar con más claridad en la siguiente figura.

```
HttpClient httpclient = new DefaultHttpClient();
HttpPost httppost = new HttpPost(JSONClient.serverBaseUrl + "/registrarPosicion.php");
StringEntity postEntity = new StringEntity(datosActuales.toString());
postEntity.setContentType("application/json; charset=utf-8");
httppost.setEntity(postEntity);
//ejecuto peticion enviando datos por POST
HttpResponse response = httpclient.execute(httppost);
```

Figura 19. Conexiones HTTP para el servicio de localización de un usuario.

La misma clase *LocationService* donde nos encontramos permite que la aplicación continúe en *background* con el fin de enviar la mejor localización en intervalos de un minuto. Esto es muy útil en el caso en que el usuario necesita salir de la aplicación para realizar otra operación como recibir una llamada o revisar el correo electrónico y, mientras no cierre por completo esta, seguirá en *background* hasta que decida acabar destruirla.

Recordamos que Para poder realizar una conexión es necesario tener declarado en el manifiesto el permiso que da acceso a Internet *android.permission.INTERNET*.

5.2.7. Interacciones entre usuarios.

Llegados a este punto, el usuario ya puede desplegar utilizar las opciones de interacción entre usuarios que aparecen en el código de la figura 20. Estas son: llamar al usuario que aparece en

el desplegable, enviarle un correo electrónico o visualizar su estado de reparación junto con la fecha de última actualización del mismo. Estas aparecen en la captura de pantalla proporcionada en la figura 23.

A continuación, se proporciona el código correspondiente a la implementación de las llamadas a las distintas opciones de usuario mencionadas.

```
public void onClick(View view) {
    switch(view.getId()){
        case R.id.accion_localizar:
            try {
                obtenerPosicion(usuarioSeleccionado.getString("username"));
            }catch (Exception e){
                e.printStackTrace();
            }
            break;
        case R.id.accion_llamar:
            try {
                Intent telefonoIntent = new Intent(Intent.ACTION_DIAL, Uri.parse("tel:" + usuarioSeleccionado.getString("numero")));
                startActivity(telefonoIntent);
            }catch (Exception e){
                e.printStackTrace();
            }
            break;
        case R.id.accion_email:
            try {
                Intent sendMailIntent = new Intent(MapsActivity.this, SendMailActivity.class);
                sendMailIntent.putExtra("nombre", usuarioSeleccionado.getString("nombre"));
                sendMailIntent.putExtra("email", usuarioSeleccionado.getString("email"));
                startActivity(sendMailIntent);
            }catch (Exception e){
                e.printStackTrace();
            }
            break;
        case R.id.accion_estado:
            try {
                Intent profileIntent = new Intent(MapsActivity.this, ProfileActivity.class);
                profileIntent.putExtra("username", usuarioSeleccionado.getString("username"));
                profileIntent.putExtra("nombre", usuarioSeleccionado.getString("nombre"));
                startActivity(profileIntent);
            }catch (Exception e){
                e.printStackTrace();
            }
            break;
    }
}
```

Figura 20. Implementación de las llamadas a los métodos de interacción de usuarios.

Más concretamente, las opciones de usuario geolocalizado se comportan de la siguiente manera:

- La opción de llamada utiliza un *Intent* y lanza una *Activity telefonoIntent*. En ese momento, el sistema abrirá el gestor de llamadas de Android y marcará el número del usuario recogido anteriormente en la base de datos.
- La opción de envío de correo electrónico funciona de forma similar salvo que ahora, el *Intent* apunta a la clase *SendMailActivity* ,que extiende de *Activity,* y recoge todos los campos relativos al envío de un correo al usuario seleccionado. El nombre y el *email* del mismo ya están definidos mediante esta clase.
- La opción de cambio de estado también utiliza un *Intent* que apunta a la clase *ProfileActivity*. En esta clase se encuentra el siguiente código, correspondiente al cambio de estado de un usuario.

```
cambiarEstado = (Button)findViewById(R.id.cambiarEstado);
cambiarEstado.setOnClickListener((view) → {
        String estado = "";
        switch (grupoEstado.getCheckedRadioButtonId()){
            case R.id.libre:
                estado = "libre";
                break;
            case R.id.transito:
                estado = "transito";
                break;
            case R.id.reparacion:
                estado = "reparacion";
                break;
        }
```

Figura 21. Implementación de los estados de un usuario.

En la sección correspondiente al prototipado de la aplicación se presenta un pantallazo con los resultados de las distintas interacciones con un usuario.

A modo de detalle, se concreta que la llamada es un proceso simple y no necesita de más interacción con la aplicación por parte del usuario.

Por otro lado, en el envío del *email* por parte de un usuario, obtenemos definidos los datos de usuario como su nombre y dirección de correo. Para el buen funcionamiento de esta funcionalidad, se espera que el usuario complemente los campos *Asunto* con el tema en cuestión y en *Mensaje* incluya el nombre del cliente, su número de teléfono, dirección y avería.

Por último, el usuario puede realizar cambios en su estado, donde como vemos en la figura anterior, también dispone de su nombre y una última fecha de localización de manera predefinida. Los cambios de estado permanecen inalterables hasta que el usuario vuelve a realizar otro cambio.

6. Conclusiones finales acerca del proyecto.

Mi experiencia personal en el desarrollo de esta aplicación ha sido positiva aunque me hubiese gustado disponer de más tiempo y recursos para mejorar algunos aspectos, tal vez incluir más idiomas y poder realizar más pruebas.

A continuación, se detallan aquellas variaciones que han surgido respecto a la idea original y me han motivado a obtener una solución profesional y funcional:

- La interfaz ha sufrido cambios importantes tras comprobar errores iniciales tales como: incluir un botón para salir de la aplicación, ofrecer distintas opciones de mapa o aplicar unos colores y diseños que aportaban poca visibilidad.
- En primera instancia, se pensó en diferenciar dos roles de usuario. Más tarde, las funcionalidades finales de la aplicación demostraron que estos no son necesarios puesto que no existen privilegios sobre ninguno de ellos.
- La localización de un usuario se realiza conforme a su nombre de usuario, definido a la hora de realizar el registro. Esto es así y no mediante su número de teléfono puesto que se tiene en cuenta que el usuario puede cambiar de terminal o de número y de esa manera perdería su cuenta.
- Se suprimió la opción de mensaje de texto debido a su coste y poca funcionalidad real ya que el correo electrónico satisface todas las necesidades escritas de comunicación.
- Mediante la opción de envío de correo electrónico, el usuario puede usar la aplicación de correo instalada en su terminal que desee. Dichas opciones de uso aparecerán al pulsar sobre el botón *Enviar*.

A pesar de las muchas virtudes de la aplicación *SATApp*, esta tiene varios puntos susceptibles de mejora. Debido a la falta de tiempo y conocimientos a gran escala de la tecnología de desarrollo Android, no se ha podido sacar el máximo partido a funcionalidades como:

- Agenda de reparaciones, clientes y eventos de empleados.
- Soluciones administrativas como gestión de facturas.
- Calendario personalizado a cada empleado.

En lugar de todo ello, se ha abogado por ofrecer una interacción útil entre los empleados de una empresa y resolver la problemática acerca de la situación de cada empleado en un determinado instante.

Además de esto, en adelante, en la aplicación se podría añadir la opción de introducir manualmente la población en lugar de centrarla en una provincia en concreto.

La realización del presente trabajo final me ha servido para aplicar los conocimientos que he adquirido a lo largo de carrera y llevarlos a la práctica.

Desde un punto de vista objetivo, la aplicación *SATApp* desarrollada puede ser útil y la opinión de aquellos usuarios que han podido observar su comportamiento ha sido positiva. Concretamente, *SATApp* se ha definido en general como una aplicación práctica y fácil de usar, lo cual me crea una enorme satisfacción puesto que ese era mi objetivo.

Este proyecto me ha servido para motivarme a continuar desarrollando nuevas aplicaciones de Android.

7. Bibliografía.

Colaboradores de Wikipedia. Android [en línea]. *Wikipedia, La enciclopedia libre*, 2015 [fecha de consulta: 10 de marzo del 2015]. Disponible en http://es.wikipedia.org/wiki/Android.

Colaboradores de Wikipedia. Diseño centrado en el usuario [en línea]. *Wikipedia, La enciclopedia libre*, 2015 [fecha de consulta: 10 de marzo del 2015]. Disponible en http://es.wikipedia.org/wiki/Dise%C3%B1o_centrado_en_el_usuario.

López, Alfonso (2015, 15 de enero). "*Tendencias O.S. Móviles para el 2015*". Zona de marrones [artículo en línea]. [Fecha de consulta: 10 de marzo de 2015]. <http://www.zonademarrones.com/tendencias-o-s-moviles-para-el-2015/>

Martín, Iván (2015, 8 de enero). "Las cosas no cambian: los fabricantes no siguen el ritmo de Google en las actualizaciones [artículo en línea]. [Fecha de consulta: 10 de marzo de 2015]. <http://androidayuda.com/2015/01/08/cosas-no-cambian-los-fabricantes-no-siguen-el-ritmo-de-google-en-las-actualizaciones/>

Robledo Sacristán, C.; Robledo Fernández, D. (eds.) (2012). En: *Programación en Android* (1ª ed.) ("Aula Mentor"). Ministerio de Educación de España.

Tomás Gironés, J. (eds.) (2012). En: *El gran libro de Android* (2ª ed.) México: Alfaomega Grupo Editor.

Colaboradores de Wikipedia. Diseño centrado en el usuario [en línea]. *Wikipedia, La enciclopedia libre*, 2015 [fecha de consulta: 08 de abril del 2015]. Disponible en http://es.wikipedia.org/wiki/Dise%C3%B1o_centrado_en_el_usuario

Materiales UOC:

- ***Diseño Centrado en el Usuario.*** Muriel Garreta Domingo y Enric Mor Pera.
- ***Redacción* de textos científico-técnicos.** Nita Sáenz Higueras y Rut Vidal Oltra.

Colaboradores de Wikipedia. JSON [en línea]. *Wikipedia, La enciclopedia libre*, 2015 [fecha de consulta: 4 de mayo del 2015]. Disponible en http://es.wikipedia.org/wiki/JSON.

Colaboradores de Wikipedia. MySQL [en línea]. *Wikipedia, La enciclopedia libre*, 2015 [fecha de consulta: 04 de mayo del 2015]. Disponible en http://es.wikipedia.org/wiki/MySQL.

Madriaga, Ramiro. (2015, 23 de marzo). "Bases de datos remotas". Curso Android Studio [artículo en línea].[Fecha de consulta: 07 de mayo de 2015].<http://cursoandroidstudio.blogspot.com.ar/2015/01/base-de-datos-remotas-login.html>

Ovalle, José Pablo. (2013, 06 de mayo). "Conexión a servidor desde Android". Slideshare [artículo en línea].[Fecha de consulta: 07 de mayo de 2015].< http://es.slideshare.net/JosePabloOvalle/conexion-a-servidor-desde-android>

Colaboradores de desarrollo. Android [en línea]. Stackoverflow, 2015 [fecha de consulta: 15 de mayo del 2015]. Disponible en http://stackoverflow.com/.

Colaboradores de desarrollo. Crear un *splash* en Android [en línea].[fecha de consulta: 5 de mayo del 2015]. Disponible en https://amatellanes.wordpress.com/2013/08/27/android-crear-un-splash-screen-en-android/.

Apéndice A. Casos de uso.

En este punto, se considerarán únicamente los casos de uso para los cuales la aplicación *SATApp* está siendo desarrollada, es decir, el administrador es el actor principal y coordinador de usuarios mientras que el técnico realiza las funciones propias que facilitan su trabajo y saca el máximo partido a la aplicación.

Identificador	CU-001
Nombre:	Registro de nuevo usuario
Prioridad:	Alta
Descripción:	El nuevo usuario de la aplicación debe pulsar sobre la opción de registro para introducir sus datos personales y poder acceder a la aplicación.
Actores:	Administrador y Técnico.
Precondiciones:	- El usuario debe disponer de un terminal Android con GPS e Internet.
Iniciado por:	Administrador o Técnico.
Flujo:	1. Iniciar aplicación *SATApp*. 2. Pinchar sobre la opción de *Registro*. 3. Introducir los datos requeridos. 4. Pulsar sobre el botón *Registrar*.
Postcondiciones:	- Si el proceso es correcto, vuelve a la pantalla de *Login*, donde el usuario puede introducir ahora sus datos para acceder.
Notas	N/A.

Identificador	CU-002
Nombre:	Localización GPS propia y de un usuario.
Prioridad:	Alta.
Descripción:	Por lo general, el usuario administrador (aunque un técnico puede realizar esta misma función) pulsa en el desplegable sobre aquel nombre de usuario (inclusive el suyo) al que desea localizar. Tras esto, el usuario contempla su posición en el mapa.
Actores:	Administrador, Técnico y Servicio Google Maps.
Precondiciones:	- Ambos usuarios deben tener conexión GPS y Wi-fi o Internet móvil. - Ambos deben disponer de suficiente cobertura. - El usuario localizado, por su parte, debe haber realizado al menos una conexión al servidor para establecer su ubicación.
Iniciado por:	Administrador o Técnico.
Flujo:	1. Posicionarse en el menú principal de la aplicación. 2. Buscar en el desplegable por nombre de usuario y pinchar sobre este. 3. Comprobar ubicación GPS de dicho técnico en el mapa.
Postcondiciones:	- Si el proceso se ha realizado correctamente, se obtiene la geolocalización del usuario en concreto que se está buscando.
Notas	El fin de esta búsqueda es poder realizar alguna de las funcionalidades de comunicación con los usuarios, es decir, llamar, mandar correos electrónicos o visualizar su estado de reparación.

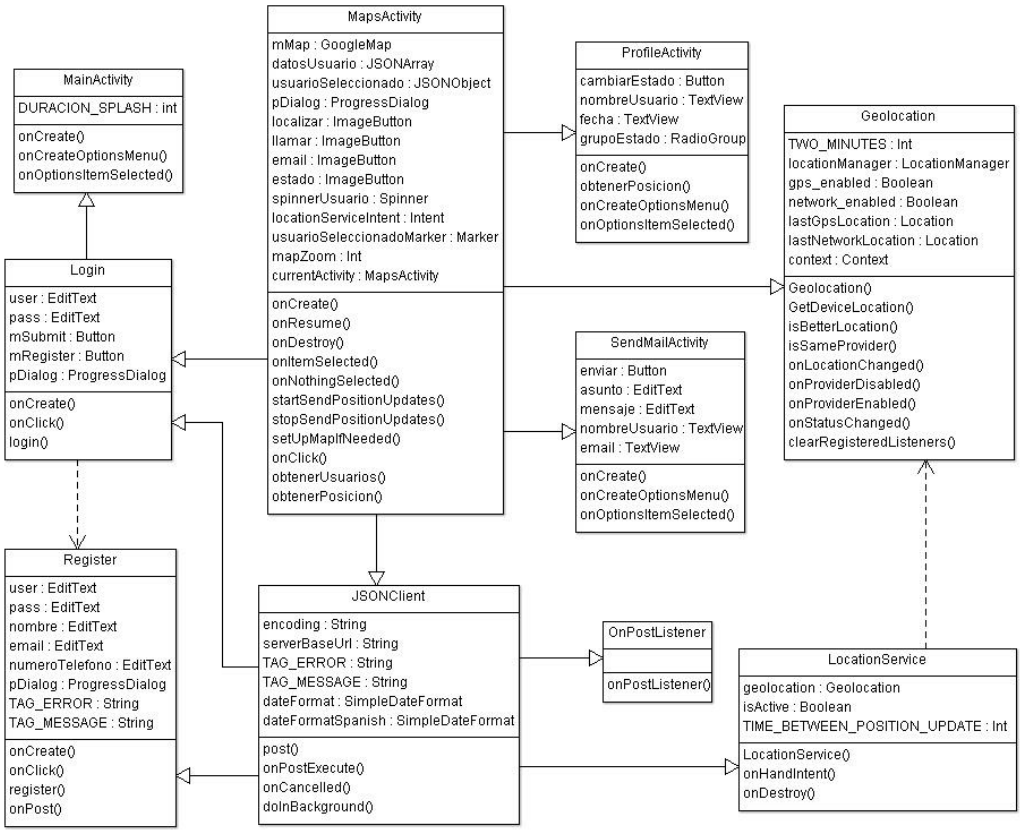
Identificador	CU-003
Nombre:	Llamar a un usuario.
Prioridad:	Media.
Descripción:	Ejecutar la opción de llamada a un usuario una vez que se ha ubicado su posición en el mapa.
Actores:	Administrador, Técnico, Servicio de Google Maps y Servicio de llamadas de Android.
Precondiciones:	- Ambos usuarios deben tener conexión GPS y Wi-fi o Internet móvil. - Ambos deben disponer de suficiente cobertura y ninguna restricción de

	llamadas. - El usuario localizado, por su parte, debe haber realizado al menos una conexión al servidor para establecer su ubicación.
Iniciado por:	Administrador o Técnico
Flujo:	1. Pinchar sobre la opción llamar una vez que el usuario se encuentra localizado. 2. Colgar la llamada cuando haya terminado su función. 3. Volver al menú donde el técnico sigue localizado por GPS.
Postcondiciones:	- Se ha conseguido realizar una llamada con el usuario deseado.
Notas	N/A.

Identificador	**CU-004**
Nombre:	Enviar correo electrónico.
Prioridad:	Alta.
Descripción:	Ejecutar la opción de envío de correo electrónico dentro del menú principal una vez que se ha ubicado la posición del usuario en el mapa.
Actores:	Administrador, Técnico, Servicio de Google Maps y Servicio de correo electrónico de Android.
Precondiciones:	- El usuario debe haber sido localizado anteriormente y, por tanto, este debe haber realizado al menos una conexión al servidor para establecer su ubicación. - Al menos una aplicación de correo electrónico debe haber sido instalada con antelación en cada terminal. - Ambos terminales deben disponer de conexión a Internet y cobertura suficiente.
Iniciado por:	Administrador o Técnico.
Flujo:	1. Pinchar sobre la opción de correo electrónico una vez que se ha localizado al usuario deseado. 2. Comprobar que el nombre de usuario es correcto. 3. Completar los campos *Asunto* y *Mensaje*. 4. Pulsar sobre *Enviar*.
Postcondiciones:	- Si el proceso se ha realizado correctamente, se ha enviado un correo electrónico a un usuario de la aplicación *SATApp* con los detalles de una reparación.
Notas	N/A.

Identificador	**CU-005**
Nombre:	Visualizar el estado de un usuario.
Prioridad:	Alta.
Descripción:	Comprobación del estado de un usuario ejecutando la opción en el menú principal y una vez que se ha ubicado la posición del usuario en el mapa.
Actores:	Administrador, Técnico, Servidor y Servicio de Google Maps.
Precondiciones:	- El usuario debe haber sido localizado anteriormente para usar esta opción. Por tanto, este debe haber realizado al menos una conexión al servidor para establecer su ubicación. - Ambos terminales, el del técnico y el del administrador, deben disponer de cobertura suficiente e Internet móvil o Wi-fi.
Iniciado por:	Administrador o Técnico.
Flujo:	1. Pinchar sobre el icono de cambio de estado en el menú principal. 2. Comprobar estado anterior y última fecha de actualización de localización. 3. Elegir el estado adecuado y pulsar sobre *Cambiar estado*.
Postcondiciones:	- Si el proceso se ha realizado correctamente, se ha cambiado el estado de reparación y comprobado la fecha de la última localización.
Notas	N/A.

Apéndice B. Modelo de clases de la aplicación *SATApp*.



Las clases mostradas en el diagrama anterior, junto con su naturaleza y función principal dentro de la aplicación *SATApp* se resumen brevemente a continuación:

- ***MapsActivity***: es la clase principal y la que coordina la mayor parte de la ejecución de la aplicación. Mediante ella, que extiende la clase *FragmentActivity*, se muestra el mapa, la interfaz de usuario y la gestión de obtención de usuario y su localización.
- ***MainActivity***: es la clase con la que inicia la aplicación. Se encarga de definir un *splash* y manda al usuario a la interfaz de inicio de sesión. Extiende de la clase *Activity*.
- ***Login***: es la clase encargada del inicio de sesión de los usuarios. Intercambia información con el servidor Web para confirmar que el usuario existe en la base de datos. Extiende de la clase *Activity*.
- ***Register***: es la clase que permite al usuario registrarse. Confirma la creación de un usuario mediante la conexión al servidor Web donde guarda los datos del mismo en la base de datos. Extiende de la clase *Activity*.
- ***JSONClient***: es la clase encargada de conectar periódicamente con el servidor para actualizar la información de la aplicación mediante peticiones POST.
- ***ProfileActivity***: gestiona los aspectos referentes a los cambios de estado de un usuario. Realiza conexiones con el servidor para actualizar los datos JSON. Extiende de la clase *Activity*.
- ***SendMailActivity***: representa el envío de un correo electrónico a un usuario. Extiende de la clase *Activity*.
- **Geolocation:** es una clase de utilidad para la geolocalización de un usuario. Determina qué tipo de localización es mejor, por Red o por GPS, y obtiene la posición de un usuario.
- **OnPostListener:** es usada por varias clases del proyecto para recibir la respuesta del servidor Web.
- **LocationService:** extiende de la clase *IntentService* y actualiza los datos de localización de un usuario con el servidor Web.

Apéndice C. Diseño de arquitectura.

Una vez que se ha descrito el funcionamiento general de la aplicación *SATApp*, las funcionalidades ofrecidas al usuario, el comportamiento del lado servidor y el intercambio de información mediante XML, a continuación se ofrece un diagrama con la arquitectura general del sistema:

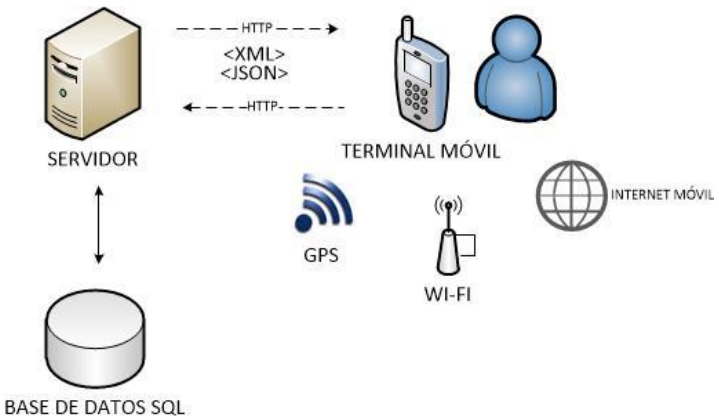


Figura 13. Arquitectura de funcionamiento del sistema *SATApp*.

En la figura anterior, se pueden apreciar los siguientes elementos descriptivos:

- **Un Dispositivo móvil:** terminal que dispone del sistema Android y donde está instalada la aplicación *SATApp*. Mediante el uso del GPS, el dispositivo conoce la ubicación actual del usuario y a través de la conexión Wi-fi, o bien por medio de Internet móvil, realiza intercambios de información con el servidor, es decir, notifica sus datos de localización y *login*, y solicita la ubicación de dicho usuario.
- **Documentos XML y JSON:** el terminal móvil realiza intercambios de documentos XML y datos JSON con el servidor utilizando el protocolo HTTP. El documento de petición comunica al servidor la ubicación actual del usuario o los datos de *login*, y solicita respuesta.
- **Servidor:** permanece siempre a la espera de conexiones por parte del administrador o los técnicos usuarios de *SATApp*.
- **Base de datos MySQL:** la base de datos almacena toda la información de localización e inicio de sesión de los usuarios. Esta es consultada en periodos breves de tiempo. En la siguiente figura, se muestra la estructura de la única tabla creada para la base de datos de la aplicación *SATApp*. No se incluye un diagrama de esta puesto que solo contiene una tabla como ha sido especificado.

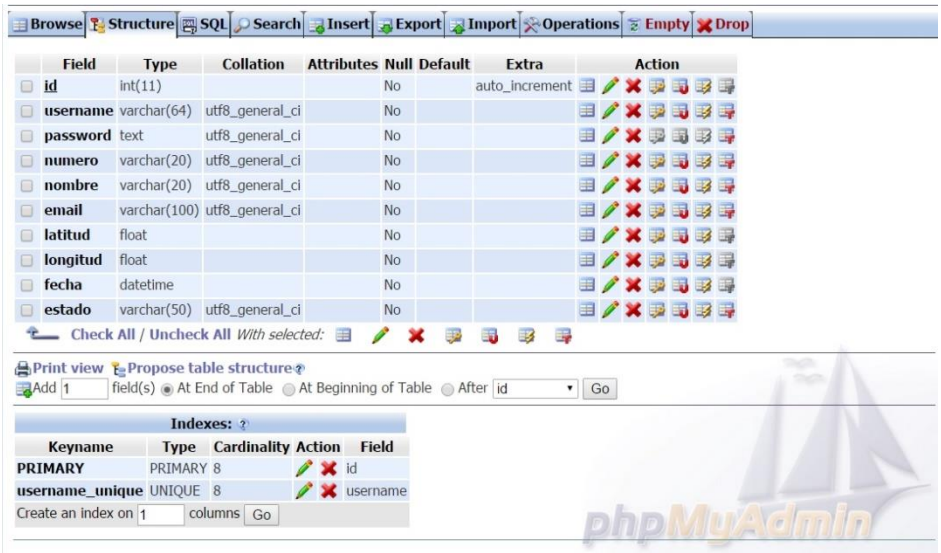


Figura 14. Estructura de la base de datos de la aplicación *SATApp*.

Apéndice D. Manual de instalación de la aplicación *SATApp*.

En este breve capítulo se ofrece un pequeño y simple manual de instalación de la aplicación *SATApp*. Como ya es sabido, *SATApp* requiere de varios componentes diferentes para poder funcionar. En concreto, se hace uso de los siguientes elementos:

- El sistema gestor de bases de datos MySQL, con la base de datos *satapp* funcionando en él. Este no necesita instalación ni configuración alguna puesto que se encuentra alojado en el Servidor Web.
- El servidor Web Apache alojado en *www.000webhost.com* funciona correctamente y pueden comprobarse sus configuraciones accediendo al servicio mediante:
  - Usuario: alejandrosaurarodriguez@gmail.com.
  - Contraseña: kCqNSXTg3MgRu6xI.
- La aplicación *SATApp*, ejecutándose correctamente en un dispositivo móvil o en un emulador que disponga de servicio GPS, a través del entorno de desarrollo Android Studio.
- Internet móvil y una cobertura aceptable.