UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf
Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Aplicación de Android para geo-localización de tareas pendientes. Implementación y gestión de servidor, comunicación e interfaz gráfica

Trabajo Fin de Grado

Grado en Ingeniería Informática

Curso: 2013 - 2014

Autor: Emanuel Alloza Álvarez

Tutor: Joan Josep Fons Cors

Resumen

Cada día que pasa la gente tiene menos tiempo para organizarse, para recordar que tenía que hacer, para buscar el sitio adecuado. ¿Cuántas veces la gente pasa por delante de un sitio el cual tenía algo que comprar o hacer y por no recordarlo le toca volver otro día?

Para ello este proyecto va explicar una aplicación móvil basada en un gestor de tareas, donde al usuario se le permitirá gestionar todas sus tareas pendientes, así como añadir sus puntos de interés favoritos donde poder realizarlas. A parte, la aplicación mostrará en todo momento su posición y las tareas que tenga más próximas, igual que notificaciones de cercanía y tiempo que permitirán al usuario recordar si tenía que realizar alguna tarea. Además, el usuario podrá sincronizar sus tareas contenidas con otros gestores, permitiéndole tener en una única aplicación, todo unificado. Toda la información de la aplicación estará almacenada en un servidor que será el que haga de conexión con el resto de gestores y donde estará almacenado todos los datos del usuario.

Para conseguir el objetivo explicado en el párrafo anterior, se ha realizado una aplicación móvil en la plataforma Android, la cual hace uso de muchas librerías propias del entorno para proporcionar al usuario una aplicación fácil de usar y que se acomode a sus necesidades. Por otro lado, la aplicación ha utilizado las APIs de Google Maps y Google Task para poder realizar el mapa y la sincronización de tareas. Para la creación del servidor y su comunicación con la aplicación y con otros gestores de tareas se ha utilizado PHP y una capa RESTful.

Una vez finalizada la aplicación y el servidor se ha conseguido que el usuario pueda tener agrupadas todas las tareas que tenga creadas en la propia aplicación, a través de un sistema unificado y él cual le notifica en todo momento si existe alguna tarea cercana. Esto ayuda mucho a las personas olvidadizas y desordenadas a mantener sus tareas al día, permitiéndoles vivir con mayor calidad de vida.

Palabras clave: gestor, tareas, aplicación, móvil, sincronización, geolocalización, servidor.

Abstract

Every day that passes people have less time to organize, to remember what they should do, to find the right place. How many times people pass in front of a site which had something to do or buy but do not remember it, and therefore must come again another day?

For this reasons, this project will explain a mobile application based on a task manager where the user will be allowed to manage all your pending tasks and add your favorite points of interest where they can perform them. In addition, the application displays your position all the time and nearest tasks, plus notifications of closeness and countdown that will remind the user to perform some task. Furthermore, users can synchronize their tasks with other managers, allowing their to have in a single application, unified whole. All application information will be stored on a server that will synchronize with other managers and stored all user data.

To achieve the objective described in the previous paragraph, we have made a mobile application on the Android platform, which makes use of many own environment libraries to provide the user an easy to use application that fulfil your needs. On the other hand, the application uses Google Maps API to perform the map and Google Task API for tasks synchronization. To develop the server and its communication interface with the application and other task managers we use PHP and a RESTful layer.

Upon completion the application and the server, the user can have grouped all the tasks created in both applications through a unified system that notifies you all the time if there is a near task. This helps a lot to people forgetful and disorganized to maintain their daily tasks, allowing them to live with greater quality of life.

Keywords: manager, tasks, application, mobile, synchronization, geo-localization, server.

Tabla de contenidos

1. Siglas y abreviaturas

HTML (HyperText Markup Language): es un lenguaje de marcado para la desarrollo de páginas web.

PHP (Hypertext Pre-Processor): es un lenguaje de programación del lado del servidor diseñado para webs de contenidos dinámicos.

HTTP (Hypertext Transfer Protocol): Es el un protocolo sin estado utilizado para la transacción web.

REST (Representational State Transfer): es una técnica de arquitectura software sobre capa HTTP que establece la idea de tratamiento del contenido como recursos con operaciones aplicables.

IDE (Integrated Development Environment): Entorno de desarrollo que ofrece frameworks y herramientas para proyectos sobre distintos lenguajes.

GPS (Global Positioning System*):* sistema de navegación espacial basado en geo-posicionamiento por satélite. Proporciona al usuario información precisa sobre su posición y hora actual.

SQL (Structured Query Language): Lenguaje estructuras de consultas de base de datos relacionales.

MySQL: Sistema gestor relacional de bases de datos, multiusuario, rápido y robusto, que utiliza el lenguaje SQL.

SqLite: es un sistema de gestión de base de datos relacional compacto, comprimido en un único fichero y muy ligero.

XML (eXtensible Markup Language): lenguaje de marcas utilizado para el almacenamiento de datos sin codificación.

SOAP (*Simple Object Access Protocol*): protocolo estándar que define la forma de que dos objetos en diferentes procesos puedan comunicarse con datos XML.

XML-RPC (eXtensible Markup Language – Remote Procedure Call): protocolo de transmisión de mensajes basado en XML.

Java: Lenguaje de programación orientado a objetos y clases, concurrente y ejecutable en cualquier dispositivo que tenga la máquina virtual de java.

API (Application Programming Interface): conjunto de funciones y procedimientos que proporciona una librería para poder ser utilizado por otros software como una capa de abstracción.

2. Tabla de ilustraciones

3. Introducción

Se ha diseñado un gestor de tareas que incorpora una capa de interoperabilidad para poder comunicarse con diferentes plataformas de tareas. Además, tiene acceso a una base de datos con geo-localización de los puntos de interés donde se pueden realizar las tareas.

El uso de este gestor se realiza a través de aplicaciones móviles, en este caso se ha decidido desarrollar una aplicación implementada en Android. La aplicación permite al usuario gestionar sus tareas, tanto de este gestor como de otros (Google Tasks,…), y geo-localizarlas en un mapa.

3.1 Historia y motivación

Actualmente el desarrollo de aplicaciones está en alza con previsión de crecimiento en el futuro. Estos datos se pueden confirmar viendo en la Ilustración las estadísticas que demuestran que los Smartphone son los dispositivos más utilizados hoy en día. Las personas utilizan su Smartphone para realizar cualquier actividad, todo esto es gracias a la gran variedad de apps que pueden contener. Estas apps permiten a las personas entretenerse, comunicarse, gestionarse,.... Y si a esto se le suma que cada día que pasa la gente tiene menos tiempo para sus tareas y vive estresada hace pensar que hacía falta una aplicación que les ayude a gestionar sus tareas cotidianas. Por ello se ha decidido realizar un gestor de tareas con su aplicación móvil para brindar a las personas el poder gestionar sus tareas sin necesidad de perder mucho tiempo. Ya no tendrán que preocuparse en recordar que tarea tenían pendiente y en donde tenían que realizarla, la aplicación se encargara de almacenar toda esa información y suministrarla cuando sea necesario.

3.2 Objetivos

En este apartado se definen los objetivos personales para este proyecto grupal, los cuales consisten en diseñar un servidor gestor de tareas sincronizable congestores conocido (Google Tasks) y la parte de la aplicación móvil consistente en inicio sesión y registro, así como la comunicación vía REST y el diseño de interfaces., Se pretende implementar las siguientes funciones:

- Representar la posición geográfica del usuario.
- Consultar el listado de tareas pendientes.
- Gestionar una base de datos con las tareas pendientes y sus puntos de interés donde se puedan realizar.
- Geo-localizar en un mapa la posición donde se oferta la localización de una tarea, y los puntos de interés donde se pueda realizar.
- Notificar las localizaciones donde se presten las tareas más cercanas al usuario.
- Diseño de una capa de acceso web usando servicios REST para consultar, actualizar y gestionar las tareas pendientes geo-localizadas.
- Implementación de la capa REST de acceso a la funcionalidad del sistema.
- Sincronización con otros gestores de tareas.

En el siguiente apartado se realizara una comparativa entre los objetivos que se han definido y aplicaciones que ya están en el mercado.

3.3 Antecedentes

Este apartado se centra en comparar los objetivos de la aplicación con otras aplicaciones que están en el mercado e intentar suplir sus deficiencias dando mayores funcionalidades al usuario. En este caso se han elegido dos bastante usadas que son explicadas a continuación:

1. ePythia :
    - 1.1 Tareas geolocalizadas
        - 1.1.1 Añadir tareas desde su localización concreta
        - 1.1.2 Asignar en el mapa localización de la tarea
        - 1.1.3 Introducción de tareas manual como de voz
        - 1.1.4 Notificaciones
        - 1.1.5 Definir radio para las notificaciones según prioridad de la tarea
        - 1.1.6 Radar para revisar lugares cercanos para realizar tareas pendientes
    - 1.2 Sincronización en la nube

2 Wunderlist:
    2.1 Planifica cualquier tarea
    2.2 Comparte con cualquiera las tareas
    2.3 Accede desde cualquier lugar
    2.4 Creación de tareas concurrentes
    2.5 Subtareas (Las tareas grandes pueden subdividirse en pequeñas para facilitar su realización)
    2.6 Notas
    2.7 Fecha de vencimiento en las tareas
    2.8 Recordatorios
    2.9 Notificaciones
    2.10 Posibilidad de imprimir
    2.11 Sincronización con la nube
    2.12 Utilizable desde cualquier dispositivo

Estas aplicaciones tienen extras que se quiere incluir en la aplicación (tareas geo-localizadas, notificaciones, fecha de vencimiento de las tareas…), pero aparte de todas estas funcionalidades nuestra aplicación también puede sincronizar tareas con otros servidores, dando al usuario la oportunidad de tener toda sus tareas contenidas en un único lugar.

Seguidamente, se explica cómo se estructura la memoria con sus diferentes apartados.

3.4 Estructura de la memoria

Al principio se han definido las diferentes siglas y abreviaturas, para que el lector tenga referencia de todas ellas según lea el contenido de la memoria. Después se visualizan el índice de la Ilustraciones. Después, está la introducción la cual pone en contexto al lector sobre lo que va a tratar el proyecto, en este punto se encuentran también definidos los objetivos y su estructura.

A continuación se detallan los diferentes apartados que han ayudado a definir el proyecto. Primero de todo se explicará una breve comparativa de las tecnologías conocidas para implementarlo y la explicación de que tipo de tecnologías se han escogido. En segundo lugar se ha definido la especificación de requisitos donde se recoge toda la información a través de la identificación de los requisitos funcionales y no funcionales. Justo después, se habla de la planificación donde está representada todas las funcionalidades que tiene que contener el proyecto, como un listado de prioridades de implementación y un

esquema cronológico con la representación del tiempo estimado que debería costar realizar el proyecto.

Seguidamente esta explicada la implementación del servidor como de los puntos que he realizado de la aplicación, en ellas se describe y visualizan los diferentes métodos que han definido las funcionalidades. Seguidamente, hay una descripción de las pruebas que se he realizado de que problemas han conllevado y su solución.

Y por último, se encuentra las conclusiones que se han llegado después de realizar todo el proyecto, como sus mejoras a largo plazo para dar mayores servicios al usuario y mejores versiones para que los usuarios tengan la aplicación que ellos deseen.

4. Contexto tecnológico

4.1 Comparativa de tecnologías, frameworks e IDEs

Se han evaluado los requisitos de la aplicación y del servidor, y tras realizar un pequeño estudio de las tecnologías disponibles se han tomado las siguientes resoluciones:

La aplicación móvil se desarrollará para la plataforma Android, debido al último estudio de Worldpanel Comtech, el 88.6% de los usuarios Españoles de dispositivos móviles utiliza dicha plataforma, sumado a que incluye una gran compatibilidad con Google Maps y otras herramientas para notificaciones. Esto complementado a la experiencia y soporte del lenguaje forman la mejor opción.

	ESPAÑA	EUROPA*	EEUU	CHINA	AUSTRALIA	JAPÓN
Android	88,6	70,7	57,6	80,0	57,3	41,5
BlackBerry	0,0	1,1	0,7	0,1	1,0	0,0
iOS	7,6	19,2	35,9	17,9	33,1	57,6
Windows	3,0	8,1	5,3	1,0	6,9	0,9
Other	0,8	0,9	0,4	1,0	1,7	0,0

Fuente: Worldpanel Comtech
*Alemania, Reino Unido, Francia, Italia, España

Ilustración 1 - Estadísticas del uso de Android

El servidor se implementará con lenguaje PHP por su notable ligereza, velocidad y rendimiento respecto a otras opciones, por ejemplo una aplicación Java. El lenguaje PHP tiene una robustez significativa suficiente para los propósitos de este servidor, es nativo en la gran mayoría de hostings, y complementado con la integración de soportes tales como MySQL, lo hace una elección acertada, dados los requisitos considerados.

4.2 Tecnologías elegidas

En este apartado se describen las tecnologías que se ha utilizado para implementación de las diferentes funcionalidades.

4.2.1 Android

Android es un sistema operativo orientado a dispositivos móviles, basado en una versión modificada del núcleo Linux. Se trata de un sistema abierto, multitarea, que permite a los desarrolladores acceder a las funcionalidades principales del dispositivo mediante aplicaciones. Cualquier aplicación puede ser reemplazada libremente o ser desarrollada por terceros, a través de herramientas proporcionadas por Google, mediante los lenguajes de programación Java y C.

El código fuente de Android está disponible bajo diversas licencias de software libre y código abierto, Google liberó la mayoría del código de Android bajo la licencia Apache. Todo esto permite que un desarrollador no solo pueda modificar su código sino también mejorarlo. A través de esas mejoras puede publicar el nuevo código y con el ayudar a mejorar el sistema operativo para futuras versiones.

4.2.2REST

Rest es tipo de arquitectura de desarrollo web que utiliza el estándar HTTP. Permite a servicios y aplicaciones ofrecer su contenido en forma de recursos, accesibles por cualquier cliente o dispositivo que utilice HTTP. Es mucho más simple y ligero que SOAP y XML-RPC.

4.2.3 MySQL

MySQL es el gestor de bases de datos de código abierto más popular del mundo. Gracias su velocidad superior, confiabilidad y facilidad de uso le ha permitido convertirse en la opción preferida para Web, Web 2.0, ISV, SaaS, empresas de telecomunicaciones y los administradores de las TI. A parte, elimina los principales problemas con el tiempo de inactividad, con el mantenimiento y la administración permitiendo a las nuevas aplicaciones ser mucho más eficaces.

4.2.4 PHP

PHP es un lenguaje de programación de uso general ejecutado en el servidor. Permite incrustar su código en un documento HTML en lugar de llamar a un archivo externo que procesara los datos. El código es interpretado por un servidor web con un módulo de procesador PHP que genera la página web resultante. Puede ser usado en cualquier servidor web como sistemas operativos y plataformas sin ningún tipo de coste. Hoy en día, PHP ha ido más

allá incluyendo una interfaz de línea de comandos que puede ser usada por aplicaciones gráficas independientes.

4.2.5 Google Tasks API

Google Tasks API proporciona a los desarrolladores un potente conjunto de herramientas para la búsqueda, lectura, actualización del contenido y los datos de Google Tasks. Esta API puede llegar a ser muy útil para los proyectos de integración con aplicaciones en conjunto con otras APIs permitiendo la comunicación entre Google Tasks y aplicaciones externas.

5. Especificación de requisitos

Lo primero que se realiza siempre si se quiere obtener un resultado satisfactorio en la aplicación es tener una buena planificación y especificación en este apartado realizado por los dos integrantes del equipo se define los pasos que se han llevado acabo para tomarlos. Casi todos los pasos han sido planteados en común, pero cada integrante ha redactado sus casos de uso individuales que estarán definidos en la memoria complementaria del otro integrante.

5.1 Toma de requisitos

El primer paso para desarrollar el gestor de tareas y la aplicación móvil es la toma de requisitos funcionales y no funcionales:

- Funcionales: Describen la interacción entre el sistema y su ambiente independientemente de su implementación. El ambiente incluye al usuario y cualquier otro sistema externo que interactúa con el sistema.
- No funcionales: Describen las restricciones y obligaciones que el sistema debe contener.

Complementario a esto, se ha realizado un análisis de las tecnologías existentes para evaluar su funcionalidad y rendimiento, para tomar las decisiones de implementación de este proyecto más adecuadas.

5.2 Requisitos funcionales

En este apartado se describe la interacción entre el sistema y su ambiente (usuario y cualquier sistema externo) independiente de su implementación.

5.2.1 Actores

Los actores son los que representan un conjunto de roles, que son jugados por una persona, dispositivo u otros sistemas que interactúan con la aplicación. En este caso, el sistema contiene cuatro roles diferentes definidos a continuación:

- ***Usuario no registrado:*** Este actor solo tiene acceso a registrarse, para después poder pasar a ser un usuario registrado.

- ***Usuario registrado:*** Es el actor que tiene acceso a todas las funcionalidades de la aplicación. Toda la aplicación está orientada para dar servicios a este usuario con la idea de mantener todas sus tareas pendientes unidas y geo-localizadas para que sepa en cada momento que tiene pendiente cerca de él.

- ***Servicio de autentificación de Google Tasks:*** Permite al usuario conectarse a su lista de tareas contenida en Google Tasks para sincronizarlas con las que tendrá en la aplicación.

- ***Administrador del servidor:*** Actor encargado de gestionar y mantener el servidor.

5.2.2 Persona

El diseño de esta aplicación es utilizable por la mayoría de usuarios habituales de dispositivos móviles, si bien es cierto que un gestor de tareas es una herramienta usable por un amplio rango de usuarios, su uso habitual y cotidiano se enfoca entre personas de entre 16 y 65 años. No se han hecho excesivas distinciones en cuanto a sexo, religión o situación económica, pues se considera que por el formato de aplicación no existen diferencias significativas.

En cuanto a la orientación hacia profesiones de los usuarios, sí que se enfoca la balanza hacia gente que trabaja en empleos de amplia jornada laboral, estresantes y lo más posible pertenecientes al sector servicios, por lo que se ha realizado el diseño visual e interactivo pensando en la rapidez y facilidad de uso.

Pensando en el estilo de vida estresante de los usuarios se ha diseñado un sistema personalizable y no intrusivo de notificaciones para ayudar por motivos

de cercanía y cuenta atrás de plazos para una tarea. Dichas notificaciones se pueden configurar por tipo de alerta, prioridad de tareas, sonido, vibración y color de led de parpadeo.

Para cubrir el mayor rango de usuarios se ha diseñado e implementado un sistema de servidor y aplicación multilenguaje con dos idiomas de serie: castellano e inglés. Con los que se cubre un gran porcentaje de la población, y es ampliable.

El usuario ejemplo o “ Persona“ a utilizar en el escenario siguiente es Paco, un varón de 30 años oficinista de una mediana empresa, casado y con un hijo. A continuación se describe un escenario ejemplo de su interacción con la aplicación.

5.2.3 Escenario

El escenario representa un claro uso del sistema en términos de una serie de interacciones entre la aplicación y el usuario. A continuación se detalla el escenario:

El usuario Paco, es una persona algo olvidadiza. Después de desayunar coge su móvil y hace a la aplicación WhereIsMyTask, donde ya tiene su usuario y contraseña guardados por haber entrado otras veces, por lo que pulsa directamente iniciar sesión.

Una vez dentro de la aplicación, visualizando el mapa, pulsa el botón de agregar tarea y crea una indicando que tiene que comprar patatas con prioridad alta, pues su novia le ha indicado que las quiere para la cena. Como valores para esta tarea establece en el campo dirección unos de sus puntos de interés favoritos (el Mercadona de la esquina) y la categoría (de alimentación). Tras guardar la tarea, vuelve a la página principal y comprueba que hay un marcador rojo en la calle del Mercadona.

Acto seguido pulsa el botón de sincronizar tareas, y tras acceder al listado de tareas, comprueba que existe la tarea “Comida de empresa a las 14:00”, que proviene de Google Tasks. Al medio día, tras haber ido a la comida y gustarle el restaurante, accede a la aplicación y en el gestor de puntos de interés agrega su nuevo restaurante favorito pulsando su ubicación en el mapa.

Por la tarde volviendo a casa, cansado de todo el día y olvidado del encargo de su novia, al pasar a cien metros del Mercadona de la esquina le salta una notificación de la aplicación advirtiéndole de la tarea y cercanía de dicho lugar, por lo que Paco ira a comprar patatas y tendrá contenta a su novia.

5.2.4 Casos de uso

Los casos de uso son los encargados de la abstracción que describe una clase de escenarios. A continuación se detallaran los diferentes casos de uso del sistema que han sido desarrollados por mí:

- ***Acceso a la aplicación***

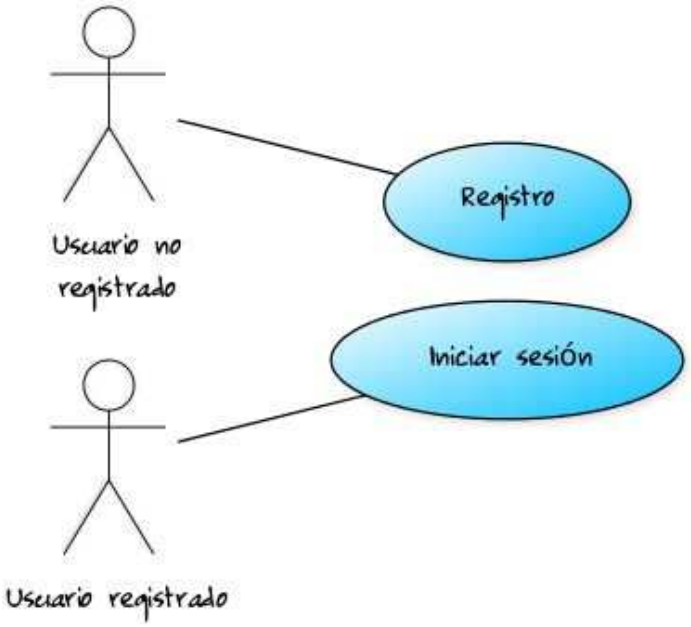


Ilustración 2 - Acceso a la aplicación

o *Registro*

Titulo:	Registro
Descripción	Los nuevos usuarios pueden crear una nueva cuenta que les permita acceder a todas las funcionalidades de la aplicación
Actor	Usuario no registrado
Relaciones	
Precondición	El correo electrónico no puede estar registrado
Comentarios	

- *Iniciar sesión*

Titulo:	Iniciar sesión
Descripción	El usuario registrado puede acceder a la aplicación a través de su correo electrónico
Actor	Usuario registrado
Relaciones	
Precondición	El correo electrónico tiene que estar registrado
Comentarios	

- ***Sincronización con Google Tasks***

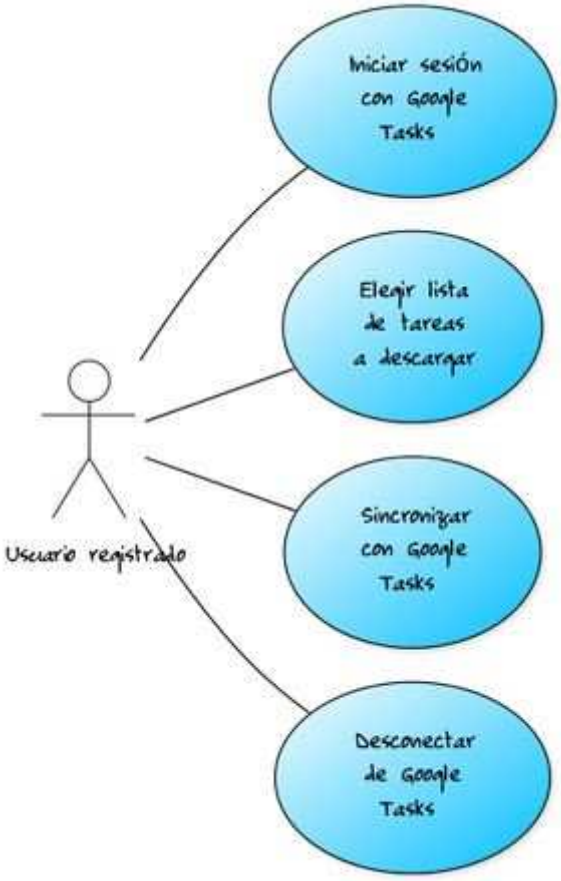


Ilustración 3 - Sincronización con Google Tasks

- *Iniciar sesión con Google Tasks*

Titulo:	Iniciar sesión con Google Tasks
Descripción	El usuario tiene la opción de conectarse a Google Tasks para poder sincronizar sus tareas
Actor	Usuario registrado
Relaciones	Conexión a Google Tasks
Precondición	Tener cuenta de google para poder acceder a Google Tasks
Comentarios	Al usuario se le pedirá permisos para tener acceso y poder sincronizar sus tareas de forma offline

- *Elegir la lista de tareas a descargar*

Titulo:	Elegir la lista de tareas a descargar
Descripción	El usuario puede seleccionar entre sus listas de tareas de Google Tasks cuál es la lista que quiere que se sincronice con la aplicación
Actor	Usuario registrado
Relaciones	Conexión a Google Tasks
Precondición	Estar conectado a Google Tasks
Comentarios	

- *Sincronizar con Google Tasks*

Titulo:	Sincronizar con Google Tasks
Descripción	Esta opción permite al usuario sincronizar sus tareas manualmente
Actor	Usuario registrado
Relaciones	Conexión a Google Tasks
Precondición	Estar conectado a Google Tasks
Comentarios	Automáticamente el usuario seleccione esta opción sus tareas serán sincronizadas con las de Google Tasks y viceversa

- *Desconectar de Google Tasks*

Titulo:	Desconectar de Google Tasks
Descripción	El usuario puede desconectarse de Google Tasks
Actor	Usuario registrado
Relaciones	Desconexión a Google Tasks
Precondición	Estar conectado a Google Tasks
Comentarios	Cuando el usuario se desconecta de Google Tasks la sincronización de tareas será desactivada

- ***Gestión de puntos de interés***

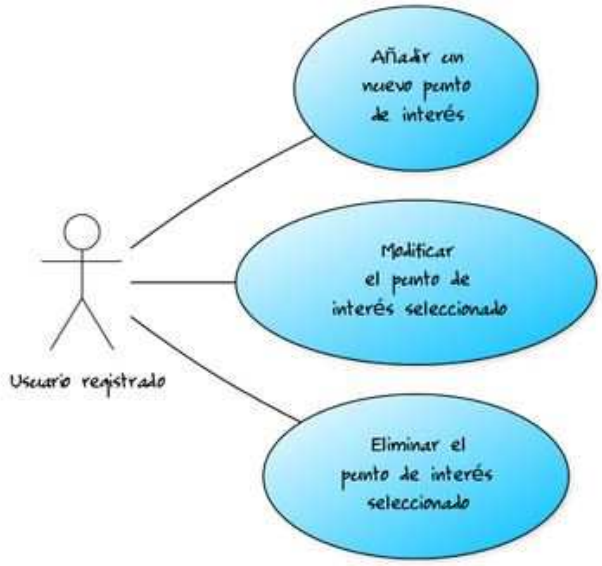


Ilustración 4 - Gestión de puntos de interés

- *Añadir un nuevo punto de interés*

Titulo:	Añadir un nuevo punto de interés
Descripción	El usuario tiene la opción de poder crear sus propios puntos de interés
Actor	Usuario registrado
Relaciones	
Precondición	
Comentarios	Cuando se vaya a crear el punto de interés se tiene la opción de poder elegir a que categoría pertenece entre las ofertadas en la aplicación

- *Modificar el punto de interés seleccionado*

Título:	Modificar el punto de interés seleccionado
Descripción	El usuario puede modificar el punto de interés que seleccione
Actor	Usuario registrado
Relaciones	
Precondición	Que el punto de interés haya sido creado previamente
Comentarios	Se puede modificar cualquier parámetro

- *Eliminar el punto de interés seleccionado*

Título:	Eliminar el punto de interés seleccionado
Descripción	El usuario puede elegir eliminar el punto de interés que tenga seleccionado
Actor	Usuario registrado
Relaciones	
Precondición	Que el punto de interés haya sido creado previamente
Comentarios	

5.3 Requisitos no funcionales

5.3.1 Usabilidad

La navegación debe ser ágil, correcta e intuitiva, y tener una funcionalidad correctamente definida. Los elementos importantes deben quedar bien definidos, siguiendo los patrones de diseño habituales para temas de color, tamaño y fuentes. El diseño de los iconos debe ser claro e intuitivo, y la disposición de los componentes debe ser consistente en todas las pantallas.

5.3.2 Apariencia

Las pantallas y transacciones entre ellas deben tener el mismo patrón de diseño, igual que los elementos explicados en el párrafo anterior deben de resultar simples y agradables.

5.3.3 Adaptación al dispositivo

Todos los elementos gráficos deben de ser adaptables a los dispositivos donde vayan a ser visualizados. Designando los tamaños de objetos y textos en cantidades proporcionales, y ajustables a todas las densidades de pantalla del mercado.

5.3.4 Almacenamiento y persistencia

El ciclo de vida de las actividades debe ser gestionado correctamente, y los ajustes de configuración tienen que estar almacenados de forma correcta. La gestión de la base de datos debe tratarse de forma eficaz y segura, gestionando las diferentes incidencias que ocurran.

5.3.5 Robustez

La aplicación no debe finalizar de forma abrupta o inesperada, ni deben de saltar excepciones sin manejar. Los mensajes de error han de ser legibles y mostrar soluciones orientadas al usuario, si es posible.

5.3.6 Mantenimiento

El código debe de ser claro y mantenible para modificaciones y actualizaciones futuras. Se debe establecer todos los parámetros posibles en constantes para su correcto tratamiento y evitar errores.

5.3.7 Documentación

Se debe realizar una documentación precisa y usable que de soporte al mantenimiento del código. Esto incluye, al menos, los diseños de las bases de datos, los javadoc y phpdoc.

5.3.8 Otros requisitos

El consumo de la batería debe intentar ser lo más óptimo posible. La precisión del GPS debe ser lo más ajustada posible. El sistema de notificaciones no debe ser intrusivo. La aplicación y el mapa tienen que ser navegables con fluidez. El consumo de datos de la aplicación debe minimizarse lo máximo.

6. Planificación

En base a la toma de requisitos realizada por los dos integrantes del grupo en el apartado anterior se han definido las funcionalidades que contiene la aplicación. Además, se ha utilizado una técnica llamada MoSCoW para priorizar los requisitos que deberá contener la aplicación. Está basado en los casos de esta memoria, así como la de mi compañera, que cumplimentan todo el sistema. Como último punto de este apartado, se puede visualizar el Diagrama de Gantt el cual indica el proceso de tiempo requerido y el dedicado en cada caso para cumplir los requisitos definidos.

6.1 Definición de las funcionalidades

- ***Ajustes del mapa:*** Contiene varias opciones que permiten al usuario personalizar el mapa a su gusto, entre ellas se puede destacar la elección del algoritmo para la actualización del GPS, la elección del número de tareas a visualizar como el número de puntos de interés por cada tarea y si quiere que se destaque la tarea seleccionada con colores diferentes.

- ***Ajustes de las notificaciones:*** Permite al usuario elegir el tipo de notificaciones que quiere recibir, según el tipo de notificación que seleccione tendrá la opción de cambiar sus parámetros.

- ***Acceso a la aplicación:*** Incluye todas las acciones relacionadas con el registro e iniciar sesión en la aplicación.

- ***Sincronización con Google Task:*** Funcionalidad que permite al usuario iniciar y autorizar la sincronización con Google Tasks dando acceso al servidor a la lista que el usuario elija para sincronizar. En el caso de que el usuario quiera que el servidor suspenda la sincronización solo tendrá que desconectarse de Google Tasks. A parte, existe la opción de que el usuario pueda sincronizar manualmente sus tareas.

- ***Gestión de puntos de interés:*** Proporciona al usuario la posibilidad de crear, modificar y eliminar los puntos de interés favoritos del usuario, para después poder seleccionarlos como localización en la creación de sus tareas.

- ***Gestión de tareas:*** Posibilita al usuario la creación, modificación y eliminación de sus tareas. También, tiene las opciones de compartir, visualizar tarea en el mapa y buscar tareas. Cada tarea esta categorizada y tiene una prioridad asignada, estos dos parámetros son elegidos al gusto del usuario en función de las opciones que proporciona la aplicación. A

parte, la localización de la tarea puede ser cualquiera que elija el usuario o la de cualquier punto de interés que se haya creado anteriormente.

- ***Mapa:*** Es la pantalla principal de la aplicación, en ella se muestran todas las tareas del usuario con las opciones que haya marcado en ajustes del mapa. Cada tarea y sus puntos de interés si tiene asignados por categoría serán pintados del color de la prioridad de la tarea, en tal caso las tareas con prioridad alta serán pintadas de color rojo, las tareas con prioridad media de color amarillo y las de prioridad baja de color verde. En el caso de que la opción de la tarea seleccionada se muestre cuando se le invoque desde gestión de tareas esta cambiara su localización principal a morado y sus puntos de interés si tiene a turquesa. Además, según el tipo de algoritmo que el usuario haya seleccionado en ajustes del mapa la sincronización del mapa será automática, por distancia o por tiempo. En cualquiera de los casos, el usuario en función de lo que haya elegido cuando su posición cambie vera como en la aplicación se actualizarán las tareas que estén a su alrededor. A parte, el usuario podrá seleccionar el botón que le lleva a su posición, permitiéndole navegar por cualquier sitio del mapa y poder volver a donde esta él de forma rápida y eficaz.

6.2 MoSCoW

MoSCoW es una técnica para asignar diferentes prioridades a las funcionalidades de la aplicación. Esta técnica permite que el desarrollo de la aplicación sea más fácil y eficaz, gracias a dar prioridades a las diferentes funcionalidades el programador puede orientarse de cuáles son las más importantes y que más relevancia tienen para que la aplicación sea funcional. Esta técnica se ha basado en los requisitos funcionales explicados en el apartado anterior. A continuación, se presenta una guía que le ayudara a entender el significado de cada carta:

- ***M – MUST (Debe de tener…):*** Define los requisitos fundamentales para el éxito del proyecto.
- ***S – SHOULD (Debería tener…):*** Representa un punto de alta prioridad que se debe incluir en la solución si es posible. Esto es a menudo un requisito crítico pero que puede estar satisfecho de otras maneras si es estrictamente necesario.
- ***C – COULD (Pueden tener…):*** Describe un requisito que se considera deseable, pero no necesario. Este será incluido si el tiempo y los recursos lo permiten.

- ***W - WON'T (No tendrá… por el momento):*** Detalla los requisitos que pueden ser pospuestos, pero que son buenos y se podrían incluir en el futuro.

A continuación, se detalla la prioridad de los requisitos de la aplicación en función a los casos de uso definidos en el apartado de Especificación de requisitos.

Tabla 1 - MoSCoW

Casos uso	Prioridad
Acceso a la aplicación	
• Registro	Must
• Iniciar sesión	Must
Ajustes del mapa	
• Seleccionar tipo de actualización del GPS	Should
• Cambiar el tiempo de actualización del GPS	Could
• Modificar la distancia de actualización del GPS	Could
• Escoger el número de tareas a visualizar	Should
• Elegir número de puntos de interés a visualizar por cada tarea	Should
• Destacar tarea seleccionada	Must
Ajustes de las notificaciones	
• Seleccionar tipo de notificaciones	Must
• Elegir la prioridad de la tarea a notificar	Should
• Escoger la distancia de las tareas a notificar	Should
• Indicar el tiempo de notificación para finalizar la tarea	Should
• Modificar vibración de la notificación	Won't
• Cambiar el sonido de la notificación	Won't
• Variar el color del led	Won't

Sincronización con Google Tasks	
• Iniciar session con Google Tasks	Must
• Elegir la lista de tareas a descargar	Must
• Sincronizar con Google Tasks manualmente	Could
• Desconectar de Google Tasks	Must
Gestión de puntos de interés	
• Añadir un nuevo punto de interés	Must
• Modificar el punto de interés seleccionado	Should
• Eliminar el punto de interés seleccionado	Should
Gestión de tareas	
• Añadir una nueva tarea	Must
• Modificar la tarea seleccionada	Must
• Eliminar la tarea seleccionada	Must
• Buscar tareas	Could
• Geo-localizar tarea seleccionada	Must
• Compartir tarea seleccionada	Could
• Sincronizar tareas	Must
• Busqueda de localización inteligente	Could
Mapa	
• Visualización de tareas	Must
• Seleccionar tarea	Could
• Ir a tu posición	Should
• Trazar ruta hacia la tarea	Won´t

6.3 Esquema cronológico

Como resultado de todo el análisis que se ha llevado a cabo en los apartados anteriores, se han planificado unas fases y el tiempo que llevarían realizarlas. Para representar este esquema se ha utilizado el Diagrama de Gantt, el cual es una representación gráfica y simultánea de la planificación como de la programación del proyecto. Esto permite que se visualice de forma fácil y rápida toda la planificación y programación de las fases de un solo vistazo.

En la primera Ilustración se puede ver las diferentes fases que han constituido el proyecto, a quien están asignadas, el tanto porciento realizado y la fecha de inicio y fin.

	Nombre de la tarea	Fecha de i	Fecha de fin	Duración	Predecesoras	% Completo	Asignado a
1	**Proyecto para geolocalización de tareas pendientes**	**12/02/14**	**23/08/14**	**193**		**98%**	
2	**Reunir requisitos**	12/02/14	25/02/14	14		1	
3	Toma de requisitos	12/02/14	17/02/14	6		100%	Ambos
4	Planificación	20/02/14	25/02/14	6		100%	Ambos
5	**Diseño**	**28/02/14**	**01/04/14**	**33**		1	
6	La interfaz grafica	28/02/14	15/03/14	16		100%	Emanuel
7	De las bases de datos	02/03/14	17/03/14	16		100%	Jessica
8	La aplicación móvil	18/03/14	22/03/14	5	7	100%	Jessica
9	Del servidor	22/03/14	01/04/14	11		100%	Emanuel
10	**Implementación**	05/04/14	26/07/14	113		1	
11	Realización de los requisitos MUST	05/04/14	20/05/14	46		100%	Ambos
12	Realización de los requisitos SHOULD	07/06/14	27/06/14	21		100%	Ambos
13	Realización de los requisitos COULD	05/07/14	26/07/14	22		100%	Ambos
14	Realización de los requistos WON'T					0%	Ambos
15	**Pruebas**	05/08/14	23/08/14	19		85%	Ambos

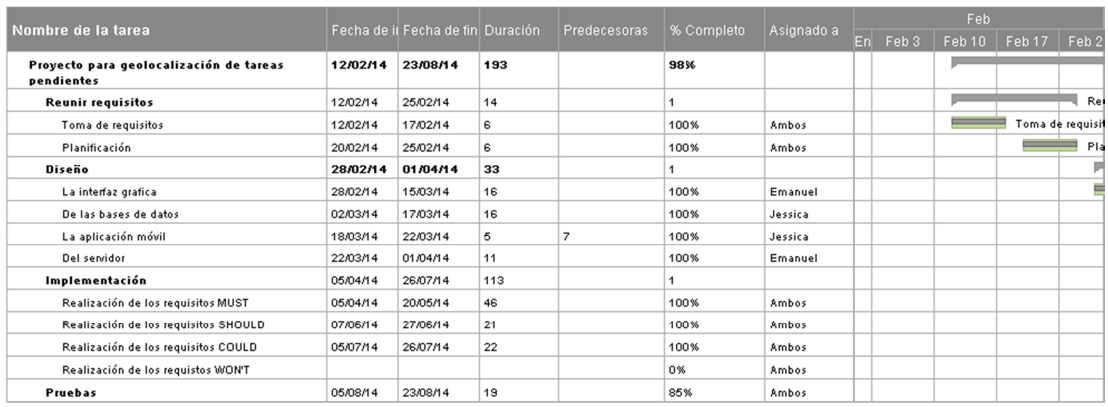


Ilustración 5 - Planificación 1

A continuación, se mostraran dos Ilustraciones que representan el tiempo que estaba planificado realizarlas.

Ilustración 6 - Planificación 2

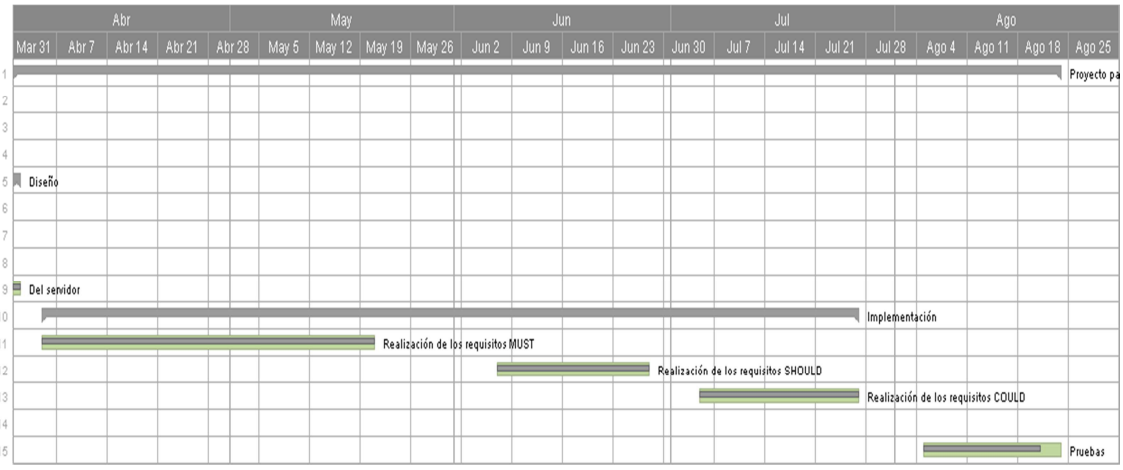


Ilustración 7 - Planificación 3

7. Diseño

En este apartado se describe el diseño de la interfaz gráfica de la aplicación la cual ha sido basada en un pequeño estudio de mercado con encuestas informales a usuarios típicos y en las guías de patrón de diseño actuales más relevantes. Todo esto está basado en el modelo de datos definido en la memoria de mi compañera.

7.1 La interfaz grafica

El diseño gráfico de la aplicación se ha pensado siguiendo las necesidades actuales de los clientes potenciales, en especial la cuota de mercado mayoritaria (jóvenes de 14 años hasta mayores de 65, basado en el tipo de la app) y con la base primordial de la guía de diseño de aplicaciones móviles para dispositivos Android de Google.

La elección de esta guía ha sido por ser la más utilizada a día de hoy y mejor valorada en varias encuestas de consumidores. Este patrón de diseño está focalizado en tres puntos principales: contenidos, objetivos y principios. Se han utilizado las ideas aplicables de sus apartados "Styles", "Layout", Components" y "Patterns".

En los siguientes apartados se muestran los mockups del diseño elegido para la aplicación y las razones de ciertos aspectos razonados por el patrón de diseño.

7.1.1 Inicio de sesión

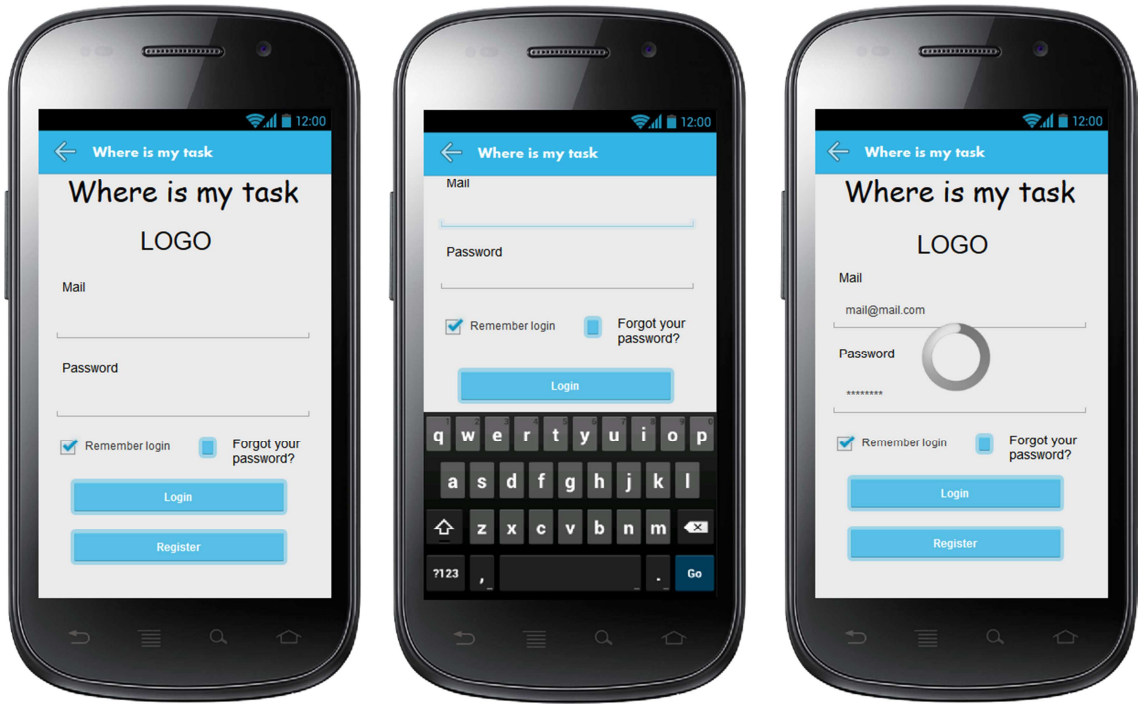


Ilustración 8 - Inicio de sesión

La interfaz presenta un diseño simple, se ha reducido la introducción de datos del usuario en la mayor medida (función recordar usuario) y el uso se concentra en las únicas acciones iniciales: Iniciar sesión, registro y solicitar nueva contraseña.

En todo momento al realizar una acción el usuario tiene un feedback visual de la situación, sea una barra de carga, un popup de aviso o el cambio a la ventana principal.

7.1.2 Registro

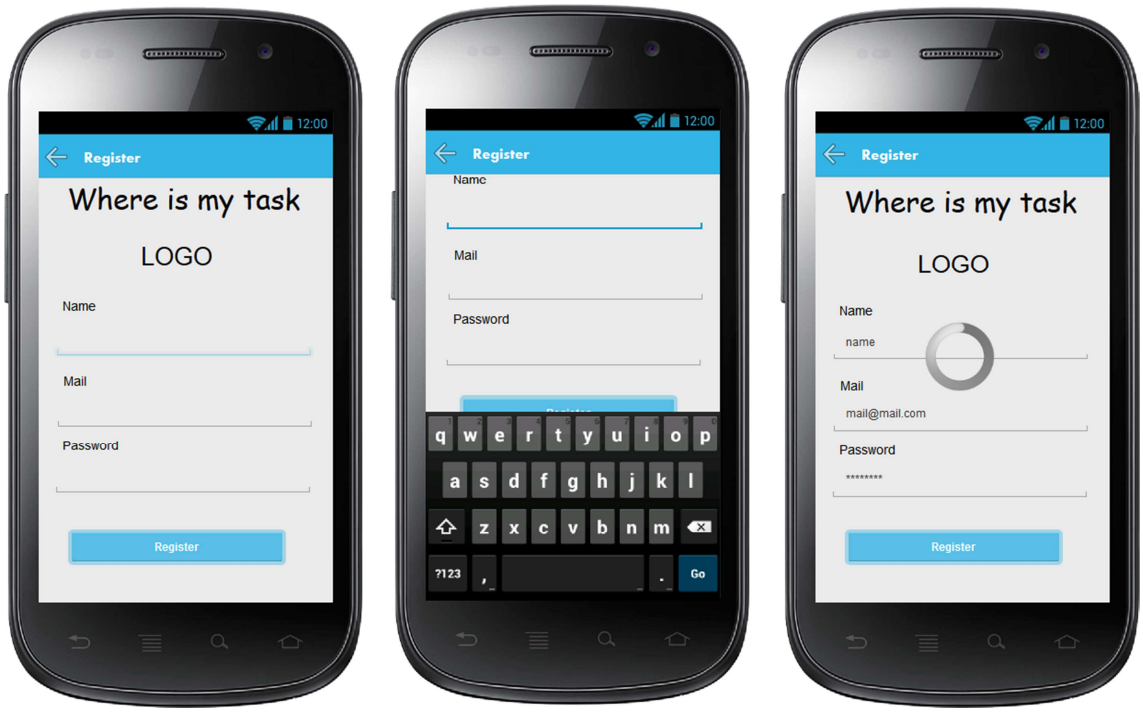


Ilustración 9 - Registro

Esta interfaz posee únicamente la acción de registro, e incluye tres campos de texto para realizar dicha tarea, devolviendo además feedback al usuario con un popup si existe algún campo erróneo o sin rellenar.

7.1.3 Mapa. Actividad principal

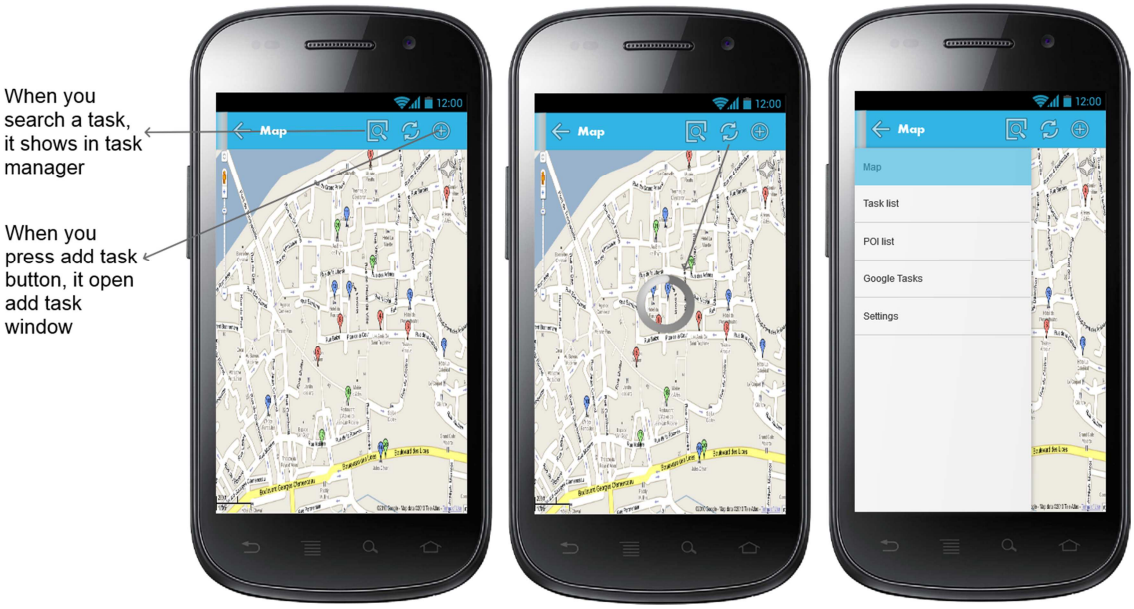


Ilustración 10 - Mapa

El diseño de la principal es totalmente ocupado por un mapa de nuestra localización con las herramientas habituales de Google Maps (botón de ir a mi posición) y con marcadores vistosos de colores estableciendo de forma fácil la identificación de las tareas a nivel de calle por parte del usuario, y basadas en la prioridad de dicha tarea.

Los marcadores para cualquier tarea en el mapa se dividen en formato semáforo, con verde para tareas menos prioritarias, amarillo para las medias y rojo para las más importantes. Como añadido, si seleccionamos una tarea concreta, nos dará su marcador principal en color morado, y si existen puntos adicionales donde poder realizarla, se pintarán en azul.

Estos códigos de colores ayudan al usuario a identificar de un vistazo toda la información relevante. Para completar esta ventana se incluyen tres acciones rápidas en la barra superior, que son un buscador, sincronización de tareas con el servidor, y agregar tarea. Los iconos utilizados son los propios de la galería ICS de Android para facilitar la curva de aprendizaje de la app.

7.1.4 Gestor de tareas

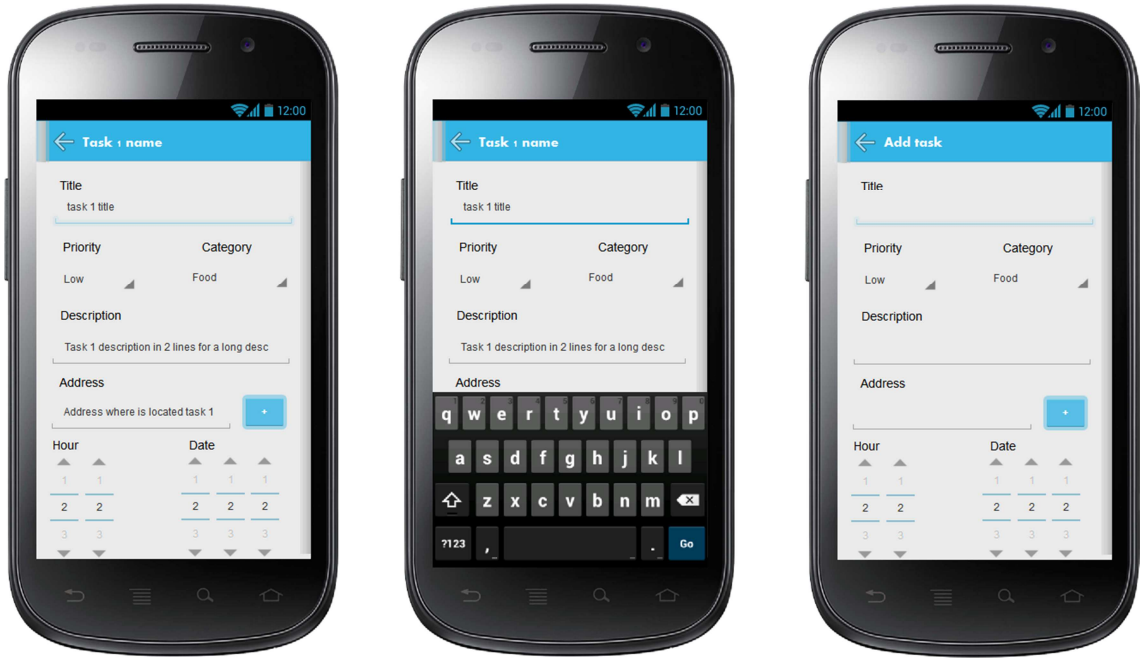


Ilustración 11 - Gestor de tareas

Se presenta una interfaz con una lista de tareas con la mínima información relevante, presentando el título, descripción y fecha de cada tarea. Los iconos de la barra de acción superior ofrecen la misma funcionalidad que en la actividad principal.

Si se pulsa sobre una tarea, esta abrirá una nueva ventana con toda la información de la tarea, complementando lo anterior con la dirección y un icono relacionado con su categoría, para identificar de forma más rápida la temática. La barra de acción superior incluye las acciones de visualizar los marcadores de localización en el mapa, editar y borrar la tarea.

7.1.5 Editor de tareas

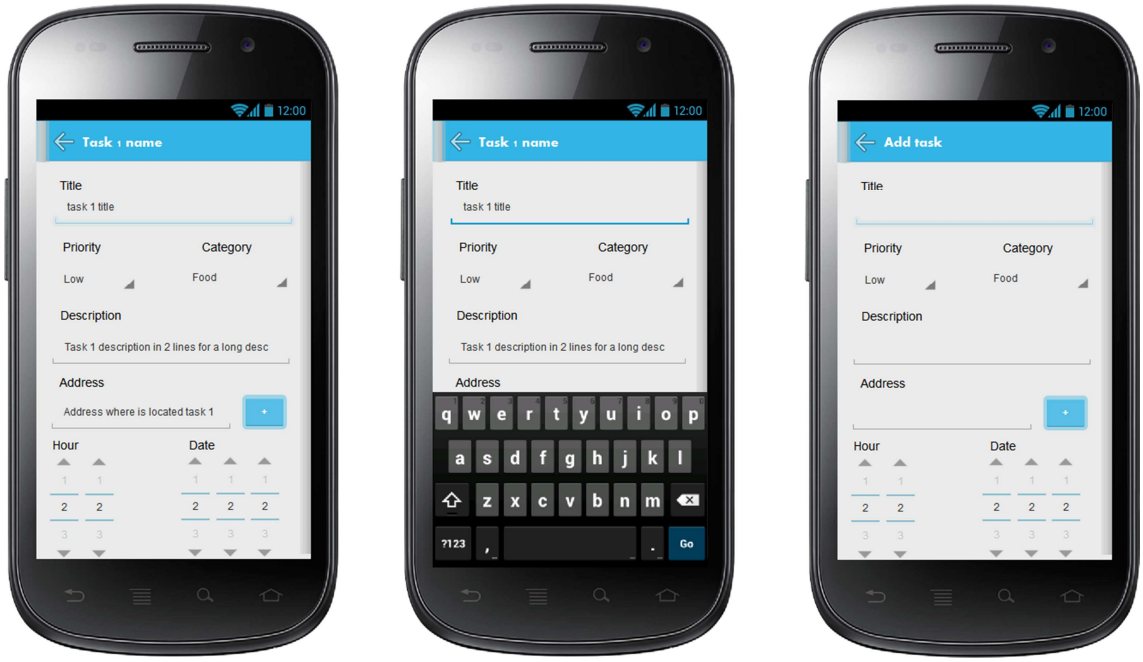


Ilustración 12 - Editor de tareas

Se compone de todos los elementos de la tarea en un formulario diseñado para facilitar la introducción de datos por parte del usuario, como ejemplo de ello se tiene en desplegables los campos de Prioridad y Categoría, la fecha y la hora con selectores de calendario, y se incluye un botón junto a la dirección que abre un popup con los puntos de interés favoritos, que rellenará el campo dirección.

Esta ventana se utiliza tanto para crear como para modificar una tarea, por lo que no requiere nuevo aprendizaje para el usuario, y ofrece feedback en caso de que exista algún fallo, aunque el diseño está reducido para lograr la mínima posibilidad de errores.

7.1.6 Gestor de puntos de interés

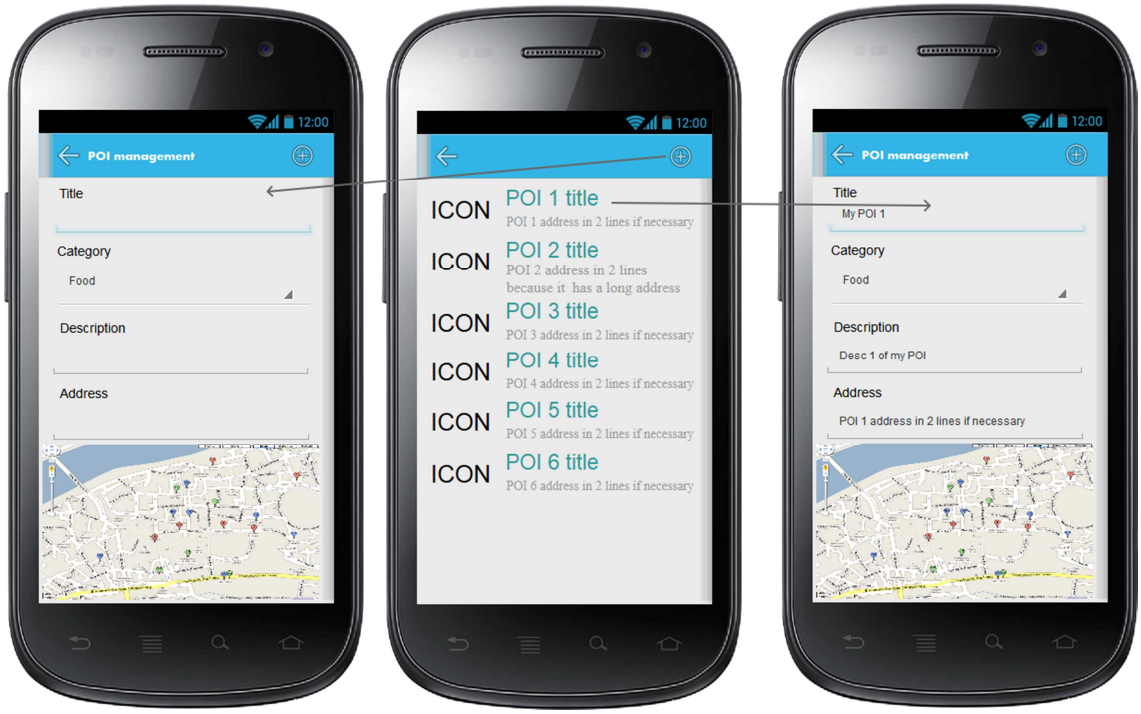


Ilustración 13 - Gestor de puntos de interés

Esta interfaz presenta un listado de puntos de interés propios del usuario, siguiendo el formato del listado de tareas. Ofrece la información mínima importante, título y dirección, y la barra de acción superior incluye un acceso a agregar POI.

Si se pulsa sobre un POI, se abrirá el editor con la información completa, que incluye además de lo visto en la ventana anterior un desplegable de categoría, un campo de descripción para que el usuario pueda personalizar información de dicha localización, y un mapa que registra la última pulsación y asigna esa localización y dirección al punto, para facilitar la inserción o modificación de los puntos. También se incluye un botón de borrado del punto de interés en la barra superior. De la misma manera que el editor de tareas, la interfaz es usada para editar o insertar.

7.1.7 Gestor de Google tareas

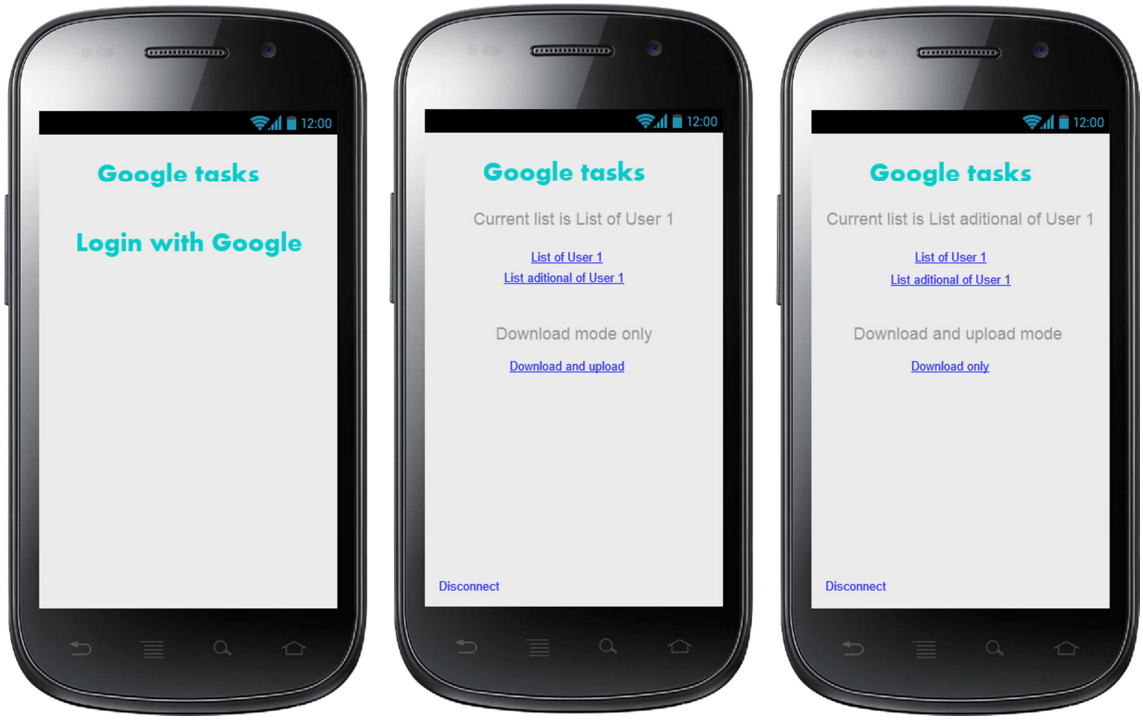


Ilustración 14 - Gestor de Google tareas

La funcionalidad de esta ventana será implementada en una pantalla web, por compatibilidad y expansión a otras aplicaciones futuras, siguiendo el estilo de colores de diseño corporativos. Su uso es básico, tras iniciar sesión con Google y conceder permisos, con un simple click permite asignar una de las listas de tareas del usuario a la sincronización automática, y de la misma forma, con un click cambiar entre los modos de sincronización sólo de descarga o descarga y subida. Por último, cuenta con un acceso de desconexión que desactivará la sincronización.

7.1.8 Ajustes

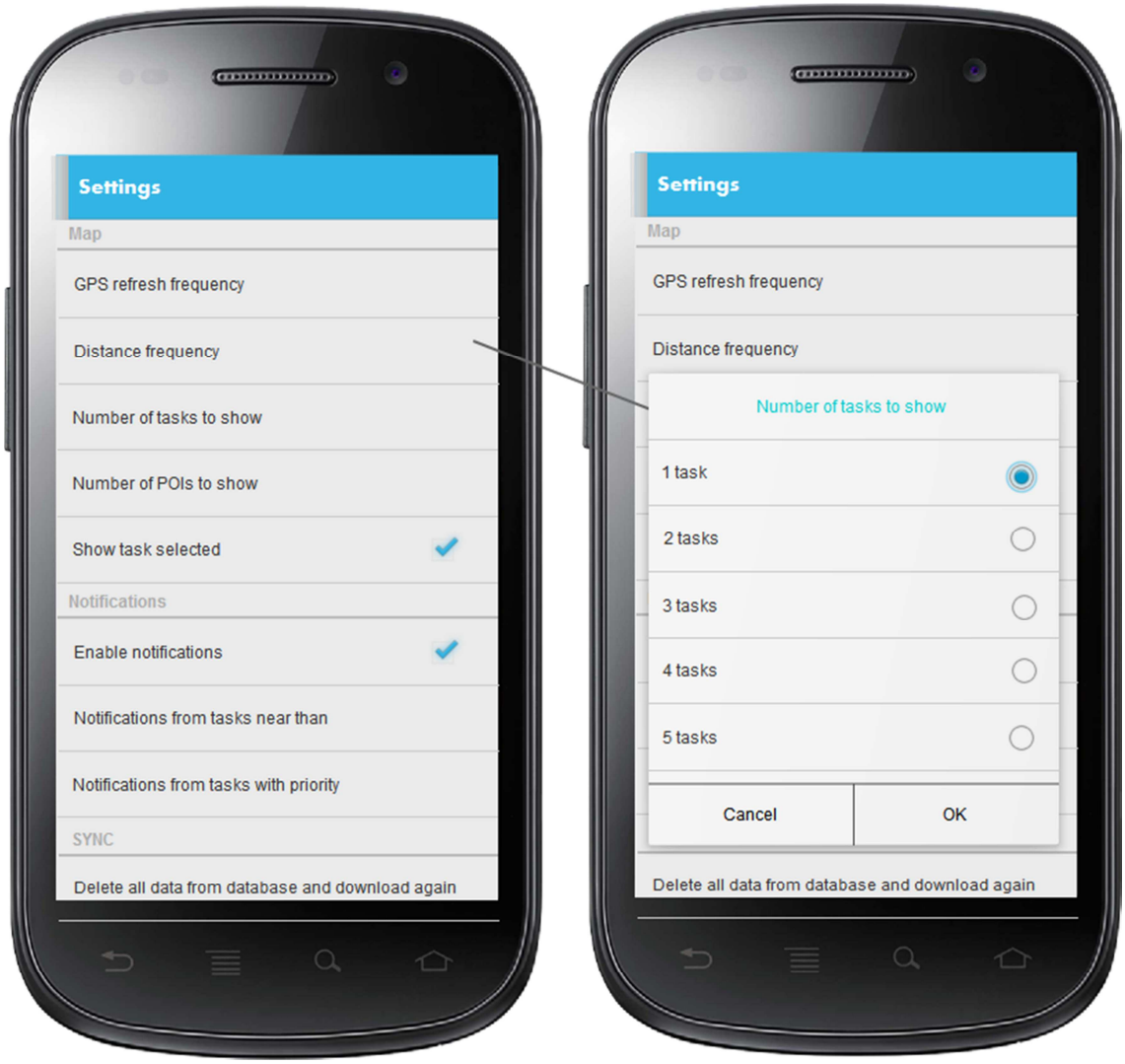


Ilustración 15 - Ajustes

Por último se presenta el diseño de la ventana de ajustes, siguiendo el esquema básico agrupado de las interfaces recomendadas por Google, con información clara y concisa de los ajustes disponibles y una selección muy rápida a través de opciones desplegables o cajas de habilitación. Los ajustes incluyen una pequeña descripción de ayuda y no requiere ninguna acción extra para almacenar los cambios.

8. Implementación

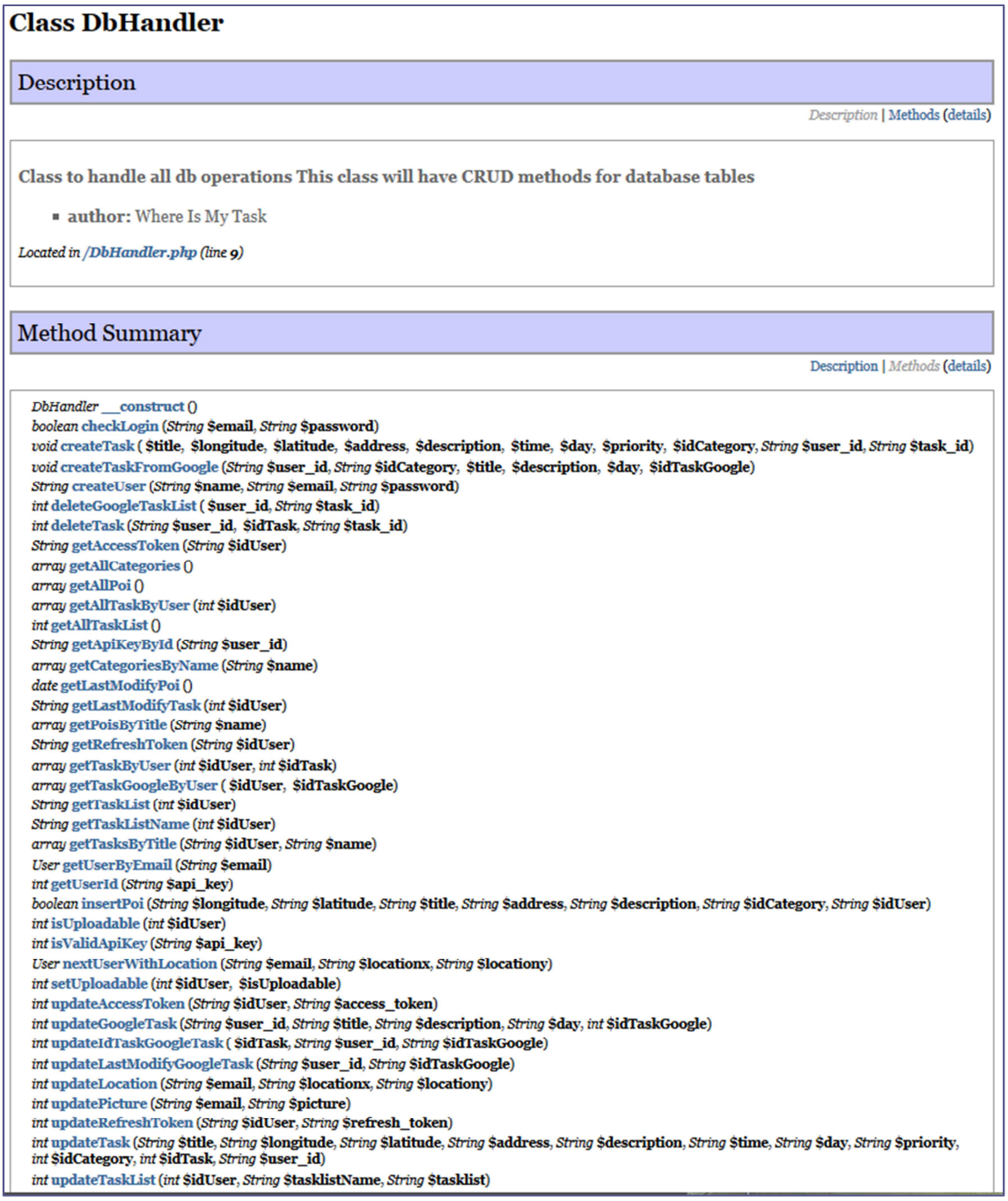
En este apartado se describen los desarrollos del servidor y las clases correspondientes de la aplicación Android que me corresponden. El resto de funcionalidades esta explicado en le memoria complementaria de mi compañera.

8.1 Servidor

En este punto se explica la estructura y funcionamiento del servidor gestor de tareas. Se ha diseñado e implementado por Emanuel Alloza Álvarez separando en capas la parte lógica y persistencia de la interfaz REST. Esto ofrece un estilo modular muy cómodo para su mantenimiento.

8.1.1 Lógica y persistencia

El diseño de la base de datos está definido en el apartado 7.1 Modelo de datos y la implementación se ha realizado en una base de datos MySQL del servidor. Para la gestión de dicha DB desde php se ha creado una clase principal llamada DbHandler.php que incluye un constructor y todas las funciones citadas en la ilustración 1.



Class DbHandler

Description

Description | Methods (details)

Class to handle all db operations This class will have CRUD methods for database tables

- **author:** Where Is My Task

Located in /DbHandler.php (line 9)

Method Summary

Description | Methods (details)

DbHandler **__construct** ()
boolean **checkLogin** (*String* **$email**, *String* **$password**)
void **createTask** (**$title**, **$longitude**, **$latitude**, **$address**, **$description**, **$time**, **$day**, **$priority**, **$idCategory**, *String* **$user_id**, *String* **$task_id**)
void **createTaskFromGoogle** (*String* **$user_id**, *String* **$idCategory**, **$title**, **$description**, **$day**, **$idTaskGoogle**)
String **createUser** (*String* **$name**, *String* **$email**, *String* **$password**)
int **deleteGoogleTaskList** (**$user_id**, *String* **$task_id**)
int **deleteTask** (*String* **$user_id**, **$idTask**, *String* **$task_id**)
String **getAccessToken** (*String* **$idUser**)
array **getAllCategories** ()
array **getAllPoi** ()
array **getAllTaskByUser** (*int* **$idUser**)
int **getAllTaskList** ()
String **getApiKeyById** (*String* **$user_id**)
array **getCategoriesByName** (*String* **$name**)
date **getLastModifyPoi** ()
String **getLastModifyTask** (*int* **$idUser**)
array **getPoisByTitle** (*String* **$name**)
String **getRefreshToken** (*String* **$idUser**)
array **getTaskByUser** (*int* **$idUser**, *int* **$idTask**)
array **getTaskGoogleByUser** (**$idUser**, **$idTaskGoogle**)
String **getTaskList** (*int* **$idUser**)
String **getTaskListName** (*int* **$idUser**)
array **getTasksByTitle** (*String* **$idUser**, *String* **$name**)
User **getUserByEmail** (*String* **$email**)
int **getUserId** (*String* **$api_key**)
boolean **insertPoi** (*String* **$longitude**, *String* **$latitude**, *String* **$title**, *String* **$address**, *String* **$description**, *String* **$idCategory**, *String* **$idUser**)
int **isUploadable** (*int* **$idUser**)
int **isValidApiKey** (*String* **$api_key**)
User **nextUserWithLocation** (*String* **$email**, *String* **$locationx**, *String* **$locationy**)
int **setUploadable** (*int* **$idUser**, **$isUploadable**)
int **updateAccessToken** (*String* **$idUser**, *String* **$access_token**)
int **updateGoogleTask** (*String* **$user_id**, *String* **$title**, *String* **$description**, *String* **$day**, *int* **$idTaskGoogle**)
int **updateIdTaskGoogleTask** (**$idTask**, *String* **$user_id**, *String* **$idTaskGoogle**)
int **updateLastModifyGoogleTask** (*String* **$user_id**, *String* **$idTaskGoogle**)
int **updateLocation** (*String* **$email**, *String* **$locationx**, *String* **$locationy**)
int **updatePicture** (*String* **$email**, *String* **$picture**)
int **updateRefreshToken** (*String* **$idUser**, *String* **$refresh_token**)
int **updateTask** (*String* **$title**, *String* **$longitude**, *String* **$latitude**, *String* **$address**, *String* **$description**, *String* **$time**, *String* **$day**, *String* **$priority**, *int* **$idCategory**, *int* **$idTask**, *String* **$user_id**)
int **updateTaskList** (*int* **$idUser**, *String* **$tasklistName**, *String* **$tasklist**)

Ilustración 16 - Lógica y persistencia

Por motivos de seguridad y eficiencia se ha utilizado la versión reciente más estable del lenguaje, PHP 5.4, y todo el sistema está tratado con objetos tipo STMT y uso de “mysqli”, versión segura del obsoleto mysql para php. Estas funciones están diseñadas pensando en la seguridad y estabilidad del programa, con comprobaciones de parámetros, y retorno de variables de control para el tratamiento adecuado de la capa REST.

A modo de ejemplo se muestran dos funciones encargadas de obtener una tarea concreta de un usuario y de crear una tarea nueva:

```php
/**
 * Fetching task by idTask
 * @param int $idTask id of the task
 * @param String $idUser id of the user
 * @return array $res returns an multidimensional array with the info of a single task from de id user
 */
public function getTaskByUser($idUser, $idTask)
{
    $stmt = $this->conn->prepare("SELECT title, longitude, latitude, address, last_modify, description, time, day, priority,
     idCategory FROM task where idUser = '" . $idUser . "' and idTask = '" . $idTask . "'");

    if ($stmt->execute())
    {
        $stmt->bind_result($title, $longitude, $latitude, $address, $last_modify, $description,
            $time, $day, $priority, $idCategory);
        $res = array(array());
        $cont = 0;

        while ($stmt->fetch())
        {
            $res[$cont]["title"] = utf8_encode($title);
            $res[$cont]["longitude"] = $longitude;
            $res[$cont]["latitude"] = $latitude;
            $res[$cont]["address"] = $address;
            $res[$cont]["last_modify"] = $last_modify;
            $res[$cont]["description"] = utf8_encode($description);
            $res[$cont]["time"] = $time;
            $res[$cont]["day"] = $day;
            $res[$cont]["priority"] = $priority;
            $res[$cont]["idCategory"] = $idCategory;
            $cont++;
        }
        $stmt->close();
        return $res;
    } else
    {
        return null;
    }
}
```

Ilustración 17 - Gestión de tareas en el servidor

En este caso se recoge una tarea y utilizamos bind_result para asignar variables y montar el array multidimensional. Si bien es cierto que al recogerse solamente una tarea, se puede realizar un array unidimensional, por formato para la capa REST y la codificación json, se ha utilizado array multidimensional para todos los casos posibles, un ejemplo de ello sería consultar todas las tareas.

```php
/**
 * Function to assign a task to user
 * @param String $user_id id of the user
 * @param String $task_id id of the task
 */
public function createTask($title, $longitude, $latitude, $address, $description,
    $time, $day, $priority, $idCategory, $user_id)
{
    $stmt = $this->conn->prepare("INSERT INTO task(title, longitude, latitude, address, description, time, day, priority,
     idCategory, idUser) values(?,?,?,?,?,?,?,?,?,?)");
    $stmt->bind_param("sddssssiii", $title, $longitude, $latitude, $address, $description,
        $time, $day, $priority, $idCategory, $user_id);
    $result = $stmt->execute();
    $stmt->close();
    return $this->conn->insert_id;
}
```

Ilustración 18 - Consultar tareas

En este otro ejemplo se asocian las variables a los parámetros de la consulta con bind_param, y se devuelve la id insertada a la interfaz REST.

8.1.2 Interfaz

Esta capa va a la ser la encargada de la comunicación mediante HTTP (RESTFUL) con el exterior. Para ello se ha hecho uso de un conocido framework de PHP llamado Slim, el cual a través de una instancia suya, es capaz de recoger las llamadas GET, POST, PUT y DELETE junto al recurso solicitado, e indicar si incluye cabeceras o requiere el uso de alguna función adicional (en nuestro caso autentificación).

Hemos desarrollado un sistema que hace uso de una función de autenticación recogiendo una clave llamada api_key a través de una cabecera “X-Authorization” que es devuelta al cliente cuando solicita inicio de sesión al servidor, siendo única por cliente y evitando el acceso a datos privados de cualquier otro usuario o simplemente uso de la aplicación sin registro. Gracias a esto todas las operaciones hacen uso de la api_key (excepto inicio de sesión y registro) y no es necesario enviar la contraseña continuamente.

La interfaz REST desarrollada también hace uso de los códigos de estado HTTP, con 200 OK en casos favorables, 400 por falta de api_key, 401 acceso denegado, etc. Estos códigos son incluidos en las respuestas codificadas en json, procurando un formato completamente funcional de una API REST.

Adicional a esto se hace uso de una clase para el cifrado de contraseñas y una función aleatoria para la generación de la api_key. Seguidamente se muestra tres funciones a modo de ejemplo para esta capa, que incluyen el registro, la autenticación (utilizada en toda la capa, como se ha explicado) y obtener tarea en función de su identificador.

En la Ilustración se puede observar la función que recibe por POST la solicitud del recurso /register donde se verifican los parámetros, se valida que el correo electrónico no exista en la base de datos y si todo es correcto se crea. Se devuelve un código de estado correcto y una respuesta codificada en json con dos parámetros. Este formato es seguido a lo largo de toda la capa REST. Los dos parámetros mínimos existentes en todas las respuestas incluyen un campo error para indicar si es todo correcto, y un campo mensaje con los datos de respuesta. En este caso concreto, además de lo mínimo, se incluye el número de identificación del usuario recién registrado.

```
/**
 * User Registration
 * url - /register
 * method - POST
 * params - name, email, password
 */
$app->post('/register', function() use ($app) {
            // check for required params
            verifyRequiredParams(array('name', 'email', 'password'));

            $response = array();

            // reading post params
            $name = $app->request->post('name');
            $email = $app->request->post('email');
            $password = $app->request->post('password');

            // validating email address
            validateEmail($email);

            $db = new DbHandler();
            $res = $db->createUser($name, $email, $password);

            if ($res["message"] == USER_CREATED_SUCCESSFULLY) {
                $response["error"] = false;
                $response["message"] = "User registered correctly. Please back and login";
                $response["id"] = $res["idInserted"];
            } else if ($res["message"] == USER_CREATE_FAILED) {
                $response["error"] = true;
                $response["message"] = "Oops! An error occurred while registering";
            } else if ($res["message"] == USER_ALREADY_EXISTED) {
                $response["error"] = true;
                $response["message"] = "Sorry, this email already existed";
            }
            // echo json response
            echoRespnse(201, $response);
        });
```

Ilustración 19 - Registro servidor

En la próxima Ilustración, se puede visualizar la función que recoge a través de GET en la dirección /tasks/ y el número de identificación de la tarea, realiza la autenticación del usuario y realiza la llamada DbHandler para consultar dicha tarea (ejemplo Ilustración anterior). Con la contestación de la base de datos devuelve una respuesta válida o un mensaje de aviso.

```php
/**
 * Listing single task from the user
 * method GET
 * url /task/:idtask
 */
$app->get('/tasks/:idtask', 'authenticate', function($idTask) use($app) {
            global $user_id;
            $response = array();
            $db = new DbHandler();

            // fetching all tasks
            $tasks = $db->getTaskByUser($user_id, $idTask);
            if (isset($tasks)) {
            $response["error"] = false;
            $response["message"] = $tasks;

             } else {
                $response["error"] = true;
                $response['message'] = "An error occurred. Please try again";
            }
            echoRespnse(200, $response);
        });
```

Ilustración 20 - Obtener tarea servidor

En esta Ilustración se puede comprobar la función de autenticado, la cual recupera las cabeceras incluidas para revisar si contiene la api_key en la cabecera X-Authorization y revisar si es válido. En caso de no serlo o no ir incluido se rechaza el acceso al servidor para la consulta formulada.

```php
 * Checking if the request has valid api key in the 'X-Authorization' header
 */
function authenticate(\Slim\Route $route) {
    // Getting request headers
    $headers = apache_request_headers();
    $response = array();
    $app = \Slim\Slim::getInstance();

    // Verifying From Header
    if (isset($headers['X-Authorization'])) {
        $db = new DbHandler();

        // get the api key
        $api_key = $headers['X-Authorization'];

        // validating api key
        if (!$db->isValidApiKey($api_key)) {
            // api key is not present in users table
            $response["error"] = true;
            $response["message"] = "Access Denied. Invalid Api key";
            echoRespnse(401, $response);
            $app->stop();
        } else {
            global $user_id;
            // get user primary key id
            $user = $db->getUserId($api_key);
            if ($user != NULL)
                $user_id = $user;
        }
    } else {
        // api key is missing in header
        $response["error"] = true;
        $response["message"] = $headers;
        echoRespnse(400, $response);
        $app->stop();
    }
}
```

Ilustración 21 - Autenticación servidor

En las siguientes Ilustraciones se muestra el código correspondiente a la ventana web que gestiona la asignación de lista de tareas desde Google Tasks de un usuario. Esta clase utiliza los tokens de acceso y refresco concedidos por Google cuando el usuario nos concede permisos. En ella están incluidos la inserción, actualización de listas de Google Tasks del usuario en nuestra base de datos, así como el refresco de tokens.

```php
if(isset($_SESSION['user_id'])) {
  if(!isset($_SESSION['access_token'])) $_SESSION['access_token'] = $db->getAccessToken($_SESSION['user_id']);
$client = new Google_Client();
$client->setClientId($client_id);
$client->setClientSecret($client_secret);
$client->setRedirectUri($redirect_uri);
$client->setAccessType('offline');
$client->addScope("https://www.googleapis.com/auth/tasks");
/************************************************
Create Tasks service.
 ************************************************/
$task_service = new Google_Service_Tasks($client);

if (isset($_GET['logout'])) {

  unset($_SESSION['access_token']);
  $client->revokeToken($db->getRefreshToken($_SESSION['user_id']));
  $valueDB = $db->deleteGoogleTaskList($_SESSION['user_id']);
  header('Location: ' . $redirect_uri . '?user_id=' . $_SESSION['user_id']);
}

if (isset($_GET['code'])) {
  $client->authenticate($_GET['code']);
  $_SESSION['access_token'] = $client->getAccessToken();
  $isFirstTime = $client->getRefreshToken();

   //saving refresh token
   if(isset($isFirstTime) && $isFirstTime != ''){
        $isSavedRefreshToken = $db->updateRefreshToken($_SESSION['user_id'],$isFirstTime);
        if(!$isSavedRefreshToken) echo "<h2><span style='color:#FCC'>" . ERROR_SAVING_TOKEN . "</span></h2>";
   }

  $redirect = 'http://' . $_SERVER['HTTP_HOST'] . $_SERVER['PHP_SELF'];
  header('Location: ' . filter_var($redirect, FILTER_SANITIZE_URL));
}
if (isset($_SESSION['access_token']) && $_SESSION['access_token']) {
   //saving actual access token
  $client->setAccessToken($_SESSION['access_token']);
  $db->updateAccessToken($_SESSION['user_id'], json_encode(json_decode($client->getAccessToken())));
} else {
  $authUrl = $client->createAuthUrl();
}
 if($client->isAccessTokenExpired()){
/* get from db and apply the refresh token */
   $client->revokeToken($_SESSION['access_token']);
   $refresh_token_from_db = $db->getRefreshToken($_SESSION['user_id']);
   if(isset($refresh_token_from_db)) $client->refreshToken($refresh_token_from_db);
    }
```

Ilustración 22 - Gestión de Google Tasks 1

```php
if (isset($_GET['tasklist'])) {
   $isUpdated = $db->updateTaskList($_SESSION['user_id'], $_GET['tasklistName'], $_GET['tasklist']);
   if(!$isUpdated) echo "<h2><span style='color:#FCC'>" . ERROR_UPDATING_TASKLIST . "</span></h2>";
    }

if (isset($_GET['uploadable'])) {
   $isUpdated = $db->setUploadable($_SESSION['user_id'], $_GET['uploadable']);
   if(!$isUpdated) echo "<h2><span style='color:#FCC'>" . ERROR_UPDATING_UPLOADABLE . "</span></h2>";
    }

if (isset($_SESSION['access_token'])) {
   $tasklist_results = $task_service->tasklists->listTasklists();
   $tasklist_selected = $db->getTaskListName($_SESSION['user_id']);
   $isUploadable = $db->isUploadable($_SESSION['user_id']);
}
} else {
    echo "<h2><span style='color:#FCC'>" . FAILED_TO_CONNECT . "</span></h2>";
    exit();
    }
echo pageHeader(GOOGLE_TASKS);
echo "<link href='http://breakwebs.com/whereismytask/google-api-php-client-master/styles/style.css' rel='stylesheet' type='text/css' />";
?>
<div class="box" style="border: none; width: 90%; min-height: 80px;" align="center">

    <?php if (isset($authUrl)) { ?>
      <a class='login' href='<?php echo $authUrl; ?>'><?php LOGIN_WITH_GOOGLE; ?></a>
    <?php } else {
        if(isset($tasklist_results)) {
      echo "<h2>" . PLEASE_SELECT_LIST . "</h2><br />";
      if($tasklist_selected != '') echo "<h3>" . CURRENT_LIST_SELECTED_IS . $tasklist_selected . "</h3><br />";
      foreach ($tasklist_results as $item) {

        echo "<a href='".$redirect_uri."?tasklist=".$item->id."&tasklistName=".$item->title."' >" . $item->title . "</a><br /> \n";
      }
      echo "<br /><h2>" . YOU_WANT_TO_UPLOAD . "</h2>";

         if($isUploadable){
            echo "<h3>" . CURRENT_UPLOADABLE_TRUE . "</h3>";
            echo "<a href='".$redirect_uri."?uploadable=0' >" . CURRENT_UPLOADABLE_FALSE . "</a><br /> \n";
         } else {
            echo "<h3>" . CURRENT_UPLOADABLE_FALSE . "</h3>";
            echo "<a href='".$redirect_uri."?uploadable=1' >" . CURRENT_UPLOADABLE_TRUE . "</a><br /> \n";
         }

     } else echo YOU_DONT_HAVE_TASKLISTS;
    }
     ?>
</div>
```

Ilustración 23 - Gestión de Google Tasks 2

Esta clase es la que se ejecuta periódicamente (a través de una tarea programa del servidor) y recoge las tareas de Google Tasks insertando o actualizándolas en el servidor. Esto se aplicara para lista que el usuario haya asignado, y si así lo ha indicado, también las sincronizara desde el servidor a Google Tasks. Está preparado para devolver una respuesta en json por si se quiere realizar la sincronización de forma manual.

```php
 $client_id = '236419216150-7mtdcgbp078qh690utf326e9ck93bbji.apps.googleusercontent.com';
 $client_secret = 'r1yaqJhB2YQcfKZxhpITT6Wz';
 $redirect_uri = 'http://breakwebs.com/whereismytask/google-api-php-client-master/synctasks.php';
 $db = new DbHandler();
 if(isset($_GET['user_id'])) $_SESSION['user_id'] = $_GET['user_id'];
 $refresh_token = $db->getRefreshToken($_SESSION['user_id']);
 if(!isset($_SESSION['access_token'])) $_SESSION['access_token'] = $db->getAccessToken($_SESSION['user_id']);

if(isset($_SESSION['user_id'])) {
$client = new Google_Client();
$client->setClientId($client_id);
$client->setClientSecret($client_secret);
$client->setRedirectUri($redirect_uri);
$client->setAccessType('offline');
$client->addScope("https://www.googleapis.com/auth/tasks");
if(($client->getAccessToken() == '') && isset($_SESSION['access_token'])) $client->setAccessToken($_SESSION['access_token']);

/************************************************
Create Tasks service.
 ************************************************/
$task_service = new Google_Service_Tasks($client);

if (isset($_GET['code'])) {
  $client->authenticate($_GET['code']);
  $_SESSION['access_token'] = $client->getAccessToken();

  $redirect = 'http://' . $_SERVER['HTTP_HOST'] . $_SERVER['PHP_SELF'];
  header('Location: ' . filter_var($redirect, FILTER_SANITIZE_URL));
}

if (isset($_SESSION['access_token'])) {
   //saving actual access token
  $client->setAccessToken($_SESSION['access_token']);
  $db->updateAccessToken($_SESSION['user_id'], json_encode(json_decode($client->getAccessToken())));
} else {
  $authUrl = $client->createAuthUrl();
}

 if($client->isAccessTokenExpired()){
   unset($_SESSION['access_token']);
/* get from db and apply the refresh token */
   $client->revokeToken($client->getAccessToken());
   //$client->revokeToken($_SESSION['access_token']);
   $refresh_token_from_db = $db->getRefreshToken($_SESSION['user_id']);
   if(isset($refresh_token_from_db)) $client->refreshToken($refresh_token_from_db);
}
```

Ilustración 24 - Sincronización Google Tasks con el servidor 1

```
if ($client->getAccessToken()) {
  $_SESSION['access_token'] = $client->getAccessToken();
  $tasklist_selected = $db->getTaskList($_SESSION['user_id']);
  $isUploadable = $db->isUploadable($_SESSION['user_id']);
  $task_results = $task_service->tasks->listTasks($tasklist_selected);
}
} else {
        $response["error"] = true;
        $response["message"] = 'Failed to connect.';
        echoRespnse(400, $response);
    }
 if (!isset($authUrl)) {
      foreach ($task_results as $item) {
       $dateTask = date('Y-m-d H:i:s', strtotime(' +2 hours', strtotime($item->updated)));
       if(isset($item->due)) $day = date('Y-m-d', strtotime($item->due));
       else $day = '0000-00-00';
// from here pick up the tasks, I check if there are on the server (idTaskGoogle) if your update is not my most recent stay, otherwise I always
       $task_result = $db->getTaskGoogleByUser($_SESSION['user_id'], $item->id);
        if(isset($task_result)){
          //check date if more recent update in $ db
            if($dateTask > $task_result['last_modify'])
                $db->updateGoogleTask($_SESSION['user_id'], $item->title, $item->notes,  $day, $item->id);
          // but, on the server is newer, updates and update tasks google my date of last modification
            else {
                $db->updateLastModifyGoogleTask($_SESSION['user_id'], $item->id);
                }
        } else {
            // if I do not have it on my server, I think I
            if($item->status == "needsAction") $db->createTaskFromGoogle($_SESSION['user_id'], 1, $item->title, $item->notes, $day, $item->id);
        }
      }
        if($isUploadable){
        foreach ($db->getAllTaskByUser($_SESSION['user_id']) as $task_from_server){
            $task = new Google_Service_Tasks_Task;
            if(($task_from_server['idTaskGoogle']) != '') {
                $task = $task_service->tasks->get($tasklist_selected, $task_from_server['idTaskGoogle']);
                $task->setTitle(utf8_decode($task_from_server['title']));
                $task->setNotes(utf8_decode($task_from_server['description']));
                if($task_from_server['day'].'T'.$task_from_server['time'].'.000Z' != "0000-00-00T00:00:00.000Z")
                    $task->setDue($task_from_server['day'].'T'.$task_from_server['time'].'.000Z');
                $result = $task_service->tasks->update($tasklist_selected,$task_from_server['idTaskGoogle'], $task);
               } else {// If no field idTaskGoogle task, insert into google tasks and save the id in $ db
                $task->setTitle(utf8_decode($task_from_server['title']));
                $task->setNotes(utf8_decode($task_from_server['description']));
                if($task_from_server['day'].'T'.$task_from_server['time'].'.000Z' != "0000-00-00T00:00:00.000Z")
                    $task->setDue($task_from_server['day'].'T'.$task_from_server['time'].'.000Z');
                $result = $task_service->tasks->insert($tasklist_selected, $task);
                $db->updateIdTaskGoogleTask($task_from_server['idTask'], $_SESSION['user_id'], $result->getId());
```

Ilustración 25 - Sincronización Google Tasks con el servidor 2

En la próxima tabla se visualiza la capa REST implementada con su recurso y solicitud, autenticación, parámetros de entrada y respuestas devueltas.

Tabla 2 - REST

Recurso	Auth	Parámetros entrada	Respuesta json devuelta
POST /register	No	• name • email • password	• error: false o true • message: mensaje • id: si todo ok
POST /login	No	• email • password	• error: false o true • message: mensaje • idUser: id usuario • name: nombre • email: email • apiKey: api_key • createdAt: fecha
POST /updateLocation	SI	• email • locationx • locationy	• error: false o true • message: mensaje
POST	SI	• email	• error: false o true

/nearUserLocation		• locationx • locationy	• message: mensaje
POST /updatePicture	SI	• email • picture	• error: false o true • message: mensaje
GET /googleTasks	SI	•	• Redirección a versión móvil de gestión de listas Google Tasks
GET /categories	NO	•	• error: false o true • message: array categorias
GET /category/:name	NO	• name en la url será el parámetro de búsqueda	• error: false o true • message: array categorias
GET /pois	NO	•	• error: false o true • date: fecha última actualización • pois: array de POIs • message: mensaje
GET /poisif/:date	NO	• date es la fecha que comprobará si está actualizado	• error: false o true • update: si está o no actualizado • date: fecha última actualización • message: array de POIs • message: mensaje
GET /poi/:name	NO	• name es el texto a buscar en el POI	• error: false o true • pois: array de POIs • message: mensaje
POST /newpoi	SI	• longitude • latitude • title • address • description • idCategory • idUser	• error: false o true • message: mensaje
GET /alltasks	SI	•	• error: false o true • message: mensaje o array de tasks
GET /tasks:idtask	SI	• idtask en la url será la id de la tarea	• error: false o true • message: mensaje o array de task
GET /tasksearch/:name	SI	• name será el texto a buscar en las tareas	• error: false o true • message: mensaje o array de task
POST /tasks	SI	• title • longitude	• error: false o true • message: mensaje

		• latitude • address • description • time • day • priority • idCategory	• idTask: id creado si es correcto
PUT /tasks/:idtask	SI	• idtask por url • title • longitude • latitude • address • description • time • day • priority • idCategory	• error: false o true • message: mensaje
DELETE /tasks/:idtask	SI	• idtask por url	• error: false o true • message: mensaje

En el siguiente apartado se explicara la estructura de la aplicación móvil donde se implementaran todas las funcionalidades que dan servicio al usuario para que pueda gestionar sus tareas pendientes.

8.2 Aplicación

En este apartado se dispone a explicar la organización de la aplicación móvil. Se ha diseñado e implementando siguiendo un modelo estructurado de tres capas que separan la persistencia, la lógica y las interfaces, siguiendo una arquitectura orientada a objetos. Se ha diseñado así pensando en el mantenimiento y futuras ampliaciones modulares de este software.

La aplicación esta implementada por los dos integrantes del grupo pero cada uno ha hecho una parte de los siguientes sub-apartados. En la siguiente Ilustración se puede observar tal estructuración, como se puede visualizar el proyecto tiene dos partes: la principal donde se implementan todas las clases que nos proporcionan las funcionalidades descritas, más el paquete res donde se almacena toda la capa visual, de idiomas, preferencias,…. Esta última ha sido diseñada completamente por mí. Además, he realizado la clase de inicio de sesión y registro, así como la capa de comunicación HTTP y pantalla de gestor de las listas de tareas de Google Tasks.

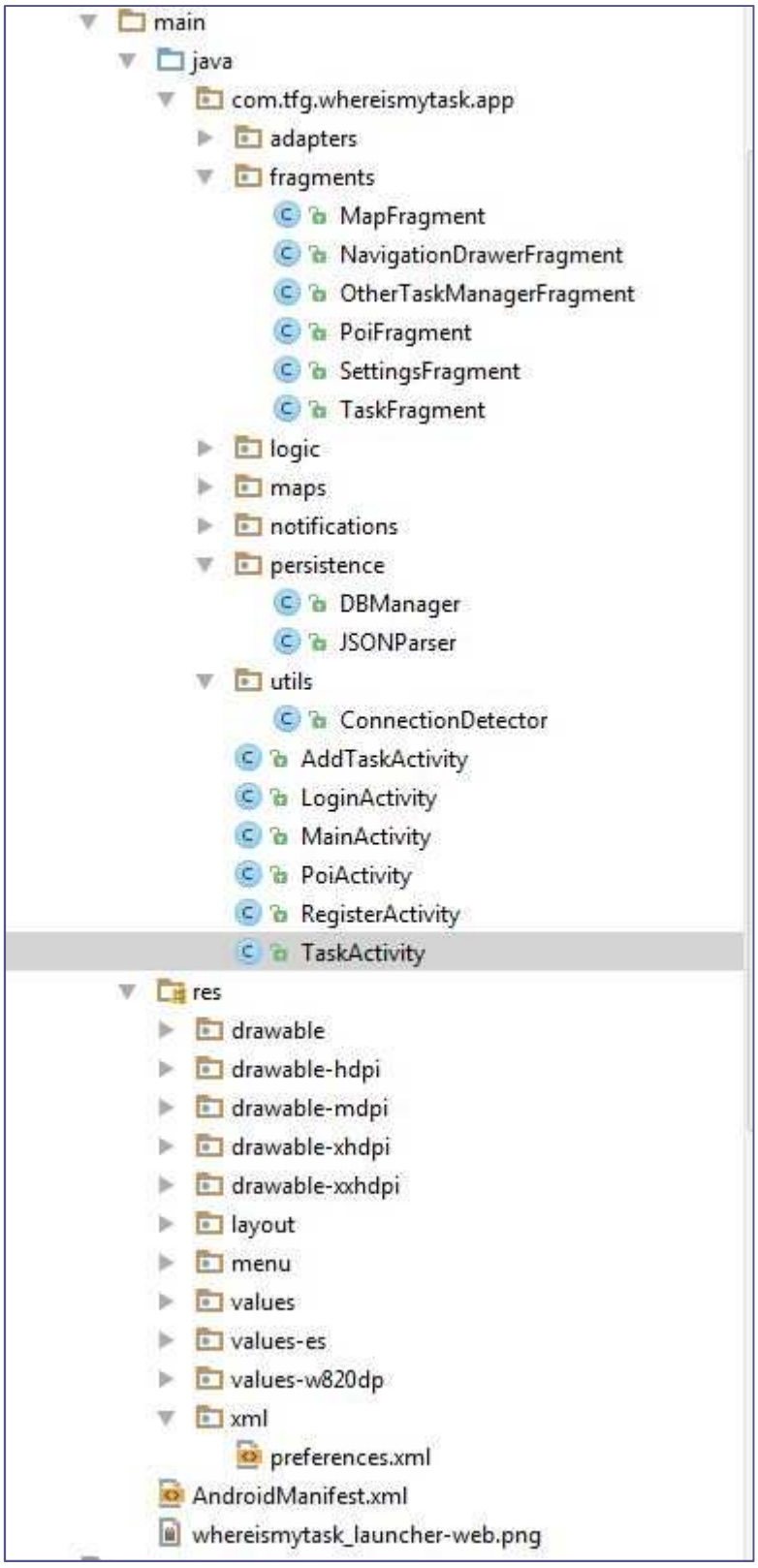


Ilustración 26 – Estructura

Una vez vista la estructuración en la Ilustración se va a proceder a explicar los diferentes paquetes que se visualizan en ella:.

8.2.1 Fragments: En esta carpeta se incluye la implementación de las clases que componen la aplicación en el menú principal. En el reparto de trabajo, se decidió asignarme la clase encargada de la integración del gestor de listas de Google Tasks.

8.2.1.1 *OtherTaskManagerFragment:* Es un WebView (navegador web con navegación desactiva que muestra la página web que gestiona la Google Tasks del usuario. En la Ilustración se puede visualizar como se ha implementado esta funcionalidad:

```
/**
 * Method responsible for creating the fragment with the data
 * @param inflater
 * @param container
 * @param savedInstanceState
 * @return
 */
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {

    View rootView = inflater.inflate(R.layout.fragment_other_task_manager, container, false);

    currentURL = "http://breakwebs.com/whereismytask/restapi/index.php/googleTasks?language=" + Locale.getDefault().getLanguage();
    dbManager = dbManager.getInstance(getActivity());
    _prefs = getActivity().getSharedPreferences(PREFERENCES_FILE, Context.MODE_PRIVATE);
    idUser = _prefs.getInt("idUserCurrent", 0);
    if (currentURL != null) {
        Map<String, String> mapHeaders = new HashMap<String, String>();
        mapHeaders.put("X-Authorization", dbManager.getApiKeyById(idUser));

        WebView wv = (WebView) rootView.findViewById(R.id.webPage);
        wv.getSettings().setJavaScriptEnabled(true);
        wv.setWebViewClient(new WebClient());
        wv.loadUrl(currentURL, mapHeaders);
    }
    return rootView;
}
```

Ilustración 33 - Carga de la Web de Google Tasks

8.2.2 Persistence: Paquete donde se encuentra la clase encargada de crear los datos que devuelve el servidor en formato json.

8.2.2.1 *JSONPaser*: Clase encargada de realizar la llamada al servidor a través de HTTP y cuando recoge los datos los codifica en formato JSON. En la primera Ilustración se puede visualizar como se realiza la llamada a través de HTTP y se recogen los datos.

```
// Making HTTP request
try {
    // check for request method
    if(method == "POST"){
        // request method is POST
        // defaultHttpClient
        DefaultHttpClient httpClient = new DefaultHttpClient();
        HttpPost httpPost = new HttpPost(url);
        httpPost.setEntity(new UrlEncodedFormEntity(params));
        if(authorization != "") httpPost.setHeader("X-Authorization", authorization);
        HttpResponse httpResponse = httpClient.execute(httpPost);
        HttpEntity httpEntity = httpResponse.getEntity();
        is = httpEntity.getContent();
    }else if(method == "GET"){
        // request method is GET
        DefaultHttpClient httpClient = new DefaultHttpClient();
        if (params != null){
            String paramString = URLEncodedUtils.format(params, "utf-8");
            url += "?" + paramString;
        }
        HttpGet httpGet = new HttpGet(url);
        if(authorization != "") httpGet.setHeader("X-Authorization", authorization);
        HttpResponse httpResponse = httpClient.execute(httpGet);
        HttpEntity httpEntity = httpResponse.getEntity();
        is = httpEntity.getContent();
    } else if (method == "PUT"){
        // request method is PUT
        // defaultHttpClient
        DefaultHttpClient httpClient = new DefaultHttpClient();
        HttpPut httpPut = new HttpPut(url);
        httpPut.setEntity(new UrlEncodedFormEntity(params));
        if(authorization != "") httpPut.setHeader("X-Authorization", authorization);
        HttpResponse httpResponse = httpClient.execute(httpPut);
        HttpEntity httpEntity = httpResponse.getEntity();
        is = httpEntity.getContent();
    }else if (method == "DELETE"){
        // request method is DELETE
        // defaultHttpClient
        DefaultHttpClient httpClient = new DefaultHttpClient();
        if (params != null){
            String paramString = URLEncodedUtils.format(params, "utf-8");
            url += "?" + paramString;
        }
        HttpDelete httpDelete = new HttpDelete(url);
        if(authorization != "") httpDelete.setHeader("X-Authorization", authorization);
        HttpResponse httpResponse = httpClient.execute(httpDelete);
        HttpEntity httpEntity = httpResponse.getEntity();
        is = httpEntity.getContent();
    }
```

Ilustración 34 - Capa REST

En la siguiente Ilustración se puede ver como codifica estos datos a formato JSON, para después poder ser tratados en las diferentes llamadas al servidor y poder guardarlos en la base de datos local.

```
try {
    BufferedReader reader = new BufferedReader(new InputStreamReader(
            is, "iso-8859-1"), 8);
    StringBuilder sb = new StringBuilder();
    String line = null;
    while ((line = reader.readLine()) != null) {
        sb.append(line + "\n");
    }
    is.close();
    json = sb.toString();
} catch (Exception e) {
    Log.e("Buffer Error", "Error converting result " + e.toString());
}

// try parse the string to a JSON object
try {
    jObj = new JSONObject(json);
} catch (JSONException e) {
    Log.e("JSON Parser", "Error parsing data " + e.toString());
}
```

Ilustración 35 - Codificación JSON

8.2.3 Activitys: En este apartado se explica la funcionalidad de cada actividad que he desarrollado.

8.2.3.1 *LoginActivity*: Clase encargada de gestionar el acceso a la aplicación y descargar del servidor las categorías y puntos de interés genéricos. A continuación, se van a mostrar las diferentes Ilustraciones con los métodos encargados de realizar estas funcionalidades.

En la primera Ilustración se muestra el método de inicio cuando se crea la actividad, como se puede observar en él es donde se instancia todos los objetos. Además, aquí esta implementada la funcionalidad de mostrar los datos de acceso del usuario cuando le ha dado al botón de recordar datos. A parte es encargado de comprobar si los datos de categorías y punto de interés han sido ya descargados, en el caso de que no estén descargados llamara a la clase que realiza la descarga de datos en el servidor.

```java
/**
 * I charge method to initialize the parameters, and collect data on the server or database,
 * depending on whether data exists in local or not
 * @param savedInstanceState
 */
@Override
protected void onCreate(Bundle savedInstanceState) {
    // TODO Auto-generated method stub
    super.onCreate(savedInstanceState);
    setContentView(R.layout.login_layout);

    dbManager = dbManager.getInstance(this);
    categoryChecked = false;
    poiList = new ArrayList<PoiVO>();

    mail = (EditText) findViewById(R.id.mail);
    pass = (EditText) findViewById(R.id.password);
    remember = (CheckBox) findViewById(R.id.remember);

    mSubmit = (Button) findViewById(R.id.login);
    mRegister = (Button) findViewById(R.id.register);
    resetpassword = (ImageButton) findViewById(R.id.resetpassword);

    mSubmit.setOnClickListener(this);
    mRegister.setOnClickListener(this);
    resetpassword.setOnClickListener(this);

    _prefs = getSharedPreferences(PREFERENCES_FILE, Context.MODE_PRIVATE);
    _prefsEditor = _prefs.edit();
    mail.setText(_prefs.getString(EMAIL, ""));
    pass.setText(_prefs.getString(PASSWORD, ""));
    getWindow().setSoftInputMode(WindowManager.LayoutParams.SOFT_INPUT_STATE_ALWAYS_HIDDEN);

    cd = new ConnectionDetector(getApplicationContext());
    if (!dbManager.existsDataInTable(POI) || !dbManager.existsDataInTable(CATEGORY)) {
        if (cd.isConnectingToInternet()) {
            generateDirectories();
             if (!dbManager.existsDataInTable(CATEGORY)) new AttemptCategory().execute();
            if (!dbManager.existsDataInTable(POI)) new AttemptPoi().execute();
            categoryChecked = true;
        } else {
            Toast.makeText(this, "You need internet connection", Toast.LENGTH_LONG).show();
        }
    } else {
        categoryChecked = true;
    }
}
```

Ilustración 36 - Ventana de iniciar sesión

En la siguiente Ilustración se puede ver el método que controla las diferentes acciones de cada botón, como se puede observar en el caso de acceso a la aplicación se comprueba primero de todo si hay conexión, si la hay se procede a comprobar si el usuario está registrado. En el caso de no estar se le mostrara un mensaje de error y si lo esta se le mostrara la pantalla del mapa.

A parte, también se gestiona el botón de registro que simplemente llamada a su clase correspondiente. Y por último, está la funcionalidad de haber olvidado la

contraseña que mandará un correo electrónico que permitirá al usuario poder modificarla.

```
case R.id.login:
    ConnectionDetector cd = new ConnectionDetector(getApplicationContext());
    Boolean isInternetPresent = cd.isConnectingToInternet();
    //We check if there is internet connection,
    // if any make the call to the server if no ne retrieve database data, if the data exist.
    if (isInternetPresent) new AttemptLogin().execute();
    else {
        try {
            int logued = dbManager.checkLogin(mail.getText().toString().trim(), pass.getText().toString().trim());
            if (logued > 0) {
                _prefsEditor.putInt(USER_CURRENT, logued);
                if (remember.isChecked()) {
                    _prefsEditor.putString(EMAIL, mail.getText().toString().trim());
                    _prefsEditor.putString(PASSWORD, pass.getText().toString().trim());
                } else {
                    _prefsEditor.putString(EMAIL, "");
                    _prefsEditor.putString(PASSWORD, "");
                }
                _prefsEditor.commit();
                Intent i = new Intent(LoginActivity.this, MainActivity.class);
                i.putExtra(POI_LIST, poiList);
                finish();
                startActivity(i);
            } else {
                Toast.makeText(LoginActivity.this, "Login incorrect. Are you already registered?", Toast.LENGTH_LONG).show();
            }
        } catch (Exception e) {
            Toast.makeText(LoginActivity.this, "Login incorrect. Are you already registered?", Toast.LENGTH_LONG).show();
        }
    }
    break;
//We called the activity that makes user registration
case R.id.register:
    Intent i = new Intent(this, RegisterActivity.class);
    startActivity(i);
    break;
//Call any activity that allows the user to send an email to change your password
case R.id.resetpassword:
    Intent intent = new Intent(Intent.ACTION_SEND);
    intent.setType("text/plain");
    intent.putExtra(Intent.EXTRA_EMAIL, "whereismytask@gmail.com");
    intent.putExtra(Intent.EXTRA_SUBJECT, "I have a problem with my password");
    intent.putExtra(Intent.EXTRA_TEXT, "My email is " + mail.getText().toString().trim());
    startActivity(Intent.createChooser(intent, "Send Email"));
    break;
```

Ilustración 32 - Llamada a registro

A continuación se puede observar en la siguiente Ilustración el método encargado de descargar del servidor las imágenes de las categorías y almacenarlas en la tarjeta externa del dispositivo. Es necesaria esta descarga porque las imágenes son usadas en la actividad de tareas que más adelante será explicada.

```
 */
public void downloadAndSaveImages(String imageCategory) {
    String url_images = "http://breakwebs.com/whereismytask/images/";
    //Saving file to SD
    try {
        File f = new File(getLocalFolder() + File.separator + imageCategory);
        if (!f.exists()) {
            FileOutputStream fos = null;
            URL url = null;

            url = new URL(url_images + imageCategory);

            //Connect to the server to download the images
            HttpURLConnection connection = (HttpURLConnection) url.openConnection();
            connection.setDoInput(true);
            connection.connect();
            Integer answerCode = connection.getResponseCode();

            if (answerCode == 200) {
                InputStream input = connection.getInputStream();
                Bitmap image = BitmapFactory.decodeStream(input);
                ByteArrayOutputStream bytes = new ByteArrayOutputStream();
               // To use JPG and 100% quality
                image.compress(Bitmap.CompressFormat.JPEG, 100, bytes);
                //create a new file on SD card
                File file = new File(getLocalImageFolder() + File.separator + imageCategory);

                file.createNewFile();

        /*--- create a new FileOutputStream and write bytes to file ---*/
                fos = new FileOutputStream(file);
                if (fos != null) {
                    fos.write(bytes.toByteArray());
                    fos.close();
                }
            }
        }
    } catch (MalformedURLException e1) {
        e1.printStackTrace();
    } catch (FileNotFoundException e1) {
        e1.printStackTrace();
    } catch (IOException e1) {
        e1.printStackTrace();
    }
}
```

Ilustración 33 - Descarga y guardado de imágenes de las categorías

Después se puede visualizar el método encargado de hacer el acceso a la aplicación desde el servidor el cuál comprueba si el usuario existe. Y si es así guarda sus datos en la base de datos, como un control de si el usuario es nuevo o el anterior para descargar los datos correspondientes a cada uno.

```
String username = mail.getText().toString();
String password = pass.getText().toString();
try {
    // Building Parameters
    params.add(new BasicNameValuePair(EMAIL, username));
    params.add(new BasicNameValuePair(PASSWORD, password));
    Log.d("request!", "starting");
    // Getting product details by making HTTP request
    JSONObject json = jsonParser.makeHttpRequest(
            LOGIN_URL, "POST", "", params);
    // Check your log for json response
    Log.d("Login attempt", json.toString());
    // Json success tag
    success = json.getString("error");
    if (success == "false") {
        Log.d("User logeado servidor!", json.toString());
        Log.d("id User", json.getInt(ID_USER) + "");
        int idUser = json.getInt(ID_USER);
        _prefs = getSharedPreferences(PREFERENCES_FILE, MODE_PRIVATE);
        _prefsEditor = _prefs.edit();

        if (_prefs.getInt(USER_CURRENT, 0) == idUser || _prefs.getInt(USER_CURRENT, 0) == 0)
            _prefsEditor.putInt(USER_CURRENT, idUser);
        else {
            _prefsEditor.putInt(USER_OLD, _prefs.getInt(USER_CURRENT, 0));
            _prefsEditor.putInt(USER_CURRENT, idUser);
        }
        if (remember.isChecked()) {
            _prefsEditor.putString(EMAIL, mail.getText().toString().trim());
            _prefsEditor.putString(PASSWORD, pass.getText().toString().trim());
        } else {
            _prefsEditor.putString(EMAIL, "");
            _prefsEditor.putString(PASSWORD, "");
        }
        _prefsEditor.commit();
        if (dbManager.getUser(idUser) == null) {
            dbManager.addUser(new UserVO(idUser, json.getString("email"), json.getString("name"), json.getString("apiKey"),
                    json.getString("createdAt"), "", 0, 0));
        }
        Intent i = new Intent(LoginActivity.this, MainActivity.class);
        finish();
        startActivity(i);
        return json.getString(TAG_MESSAGE);
    } else {
        Log.d("Login Failure!", json.getString(TAG_MESSAGE));
        return json.getString(TAG_MESSAGE);
```

Ilustración 34 - Iniciar sesión

Seguidamente se encuentra el método encargado de recoger las categorías del servidor y guardarlas en la base de datos.

```
@Override
protected String doInBackground(String... args) {
    // TODO Auto-generated method stub
    String success;
    JSONObject jsonCategory = null;
    try {
        // Getting product details by making HTTP request
        JSONObject json = jsonParser.makeHttpRequest(
                CATEGORY_URL, "GET", "", null);

        // json success tag
        success = json.getString("error");
        if (success == "false") {
            for (int i = 0; i < json.getJSONArray("message").length(); i++) {
                jsonCategory = (JSONObject) json.getJSONArray("message").get(i);
                CategoryVO category = new CategoryVO();
                category.setName(jsonCategory.getString("nameCategory"));
                category.setIdParentCategory(jsonCategory.getInt("idParentCategory"));
                category.setImageCategory(jsonCategory.getString("imageCategory"));
                category.setLast_modify(jsonCategory.getString("last_modify"));
                if(category.getImageCategory() != null) downloadAndSaveImages(category.getImageCategory());
                dbManager.addCategory(category);
            }
            return json.getString(TAG_MESSAGE);
        } else {
            Log.d("Category Failure!", json.getString(TAG_MESSAGE));
            return json.getString(TAG_MESSAGE);
        }
    } catch (JSONException e) {
        e.printStackTrace();
    }

    return null;
}
```

Ilustración 35 - Descargar categorías del servidor

Y por último, el método encargado de realizar la llamada al servidor para recoger los puntos de interés genéricos.

```
String success, update;
JSONObject jsonPoi = null;
JSONObject json;
String datePois = "";
try {
    _prefs = getSharedPreferences("myPreferences", MODE_PRIVATE);
    _prefsEditor = _prefs.edit();
    datePois = _prefs.getString("datePois", "2014-01-01 00:00:00");

    json = jsonParser.makeHttpRequest(
            POI_DATE_URL + "/" + datePois.replace(" ", ""), "GET", "", null);
    // json success tag
    success = json.getString("error");

    if ("false".equals(success)) {
        update = json.getString("update");
        if ("false".equals(update)) {
            //delete table pois
            dbManager.deleteAllPoi();
            // date a sharedpref
            String date = json.getString("date");
            _prefsEditor = _prefs.edit();
            _prefsEditor.putString("datePois", date);
            _prefsEditor.commit();

            for (int i = 0; i < json.getJSONArray("message").length(); i++) {
                jsonPoi = (JSONObject) json.getJSONArray("message").get(i);
                PoiVO poi = new PoiVO();
                poi.setIdPOI(jsonPoi.getInt("idPOI"));
                Location l = new Location("");
                l.setLongitude(jsonPoi.getDouble("longitude"));
                l.setLatitude(jsonPoi.getDouble("latitude"));
                poi.setLocation(l);
                poi.setTitle(jsonPoi.getString("title"));
                poi.setAddress(jsonPoi.getString("address"));
                poi.setDescription(jsonPoi.getString("description"));
                poi.setLastModify(jsonPoi.getString("last_modify"));
                poi.setIdCategory(jsonPoi.getInt("idCategory"));
                poiList.add(poi);
                dbManager.addPoi(poi);
            }
        }
        return json.getString(TAG_MESSAGE);
    } else {
        Log.v("Poi Failure!", json.getString(TAG_MESSAGE));
        return json.getString(TAG_MESSAGE);
```

Ilustración 36- Descargar puntos de interés del servidor

8.2.3.2 *RegisterActivity:* Clase encargada de crear la cuenta del usuario y guardarla en servidor como en la base de datos local. A continuación, se muestra como realiza estas funcionalidades.

En la primera Ilustración se puede visualizar el método encargado de gestionar el click cuando el usuario le da a registrar, en el caso de haber internet este método llamara al método encargado de realizar la conexión con el servidor.

```java
/**
 * Method that the call to the method responsible for connecting to the server
 * @param v
 */
@Override
public void onClick(View v) {
    // TODO Auto-generated method stub
    ConnectionDetector cd = new ConnectionDetector(getApplicationContext());

    Boolean isInternetPresent = cd.isConnectingToInternet();
    if(isInternetPresent){
        new CreateUser().execute();
    }
    else Toast.makeText(RegisterActivity.this, "Internet connection is required", Toast.LENGTH_LONG).show();

}
```

Ilustración 42 - Acción de registrar

En la siguiente Ilustración se puede visualizar el método que hace la conexión con el servidor y le pasa los datos para que el usuario sea registrado. A parte, también guarda los datos en la base de datos local.

```java
    @Override
    protected String doInBackground(String... args) {
        // TODO Auto-generated method stub
        // Check for success tag
        String success;
        String username = user.getText().toString();
        String textMail = mail.getText().toString();
        String password = pass.getText().toString();

        try {
            // Building Parameters
            List<NameValuePair> params = new ArrayList<>();
            params.add(new BasicNameValuePair(NAME, username));
            params.add(new BasicNameValuePair(EMAIL, textMail));
            params.add(new BasicNameValuePair(PASSWORD, password));


            //Posting user data to script
            JSONObject json = jsonParser.makeHttpRequest(
                    LOGIN_URL, "POST","", params);

            // json success element
            success = json.getString("error");
            if (success == "false") {
                int idUser = json.getInt(ID_USER);
                _prefs = getSharedPreferences(PREFERENCES_FILE, Context.MODE_PRIVATE);
                _prefsEditor = _prefs.edit();
                _prefsEditor.putInt(USER_CURRENT, idUser);
                _prefsEditor.commit();
                UserVO user = new UserVO(idUser, textMail,username, password,"", "",0.0, 0.0);
                dbManager.addUser(user);
                return json.getString(TAG_MESSAGE);

            }else{
                return json.getString(TAG_MESSAGE);
            }
        } catch (JSONException e) {
            e.printStackTrace();
        }

        return null;

    }
```

Ilustración 43 - Registro en el servidor

9 Versión final

Aquí se muestra el resultado final de la aplicación desarrollada por los dos integrantes del grupo a lo largo de este proyecto con unas capturas de pantalla. Se puede probar la versión final con la apk instable suministrada.

Ilustración 44 - Pantalla inicio

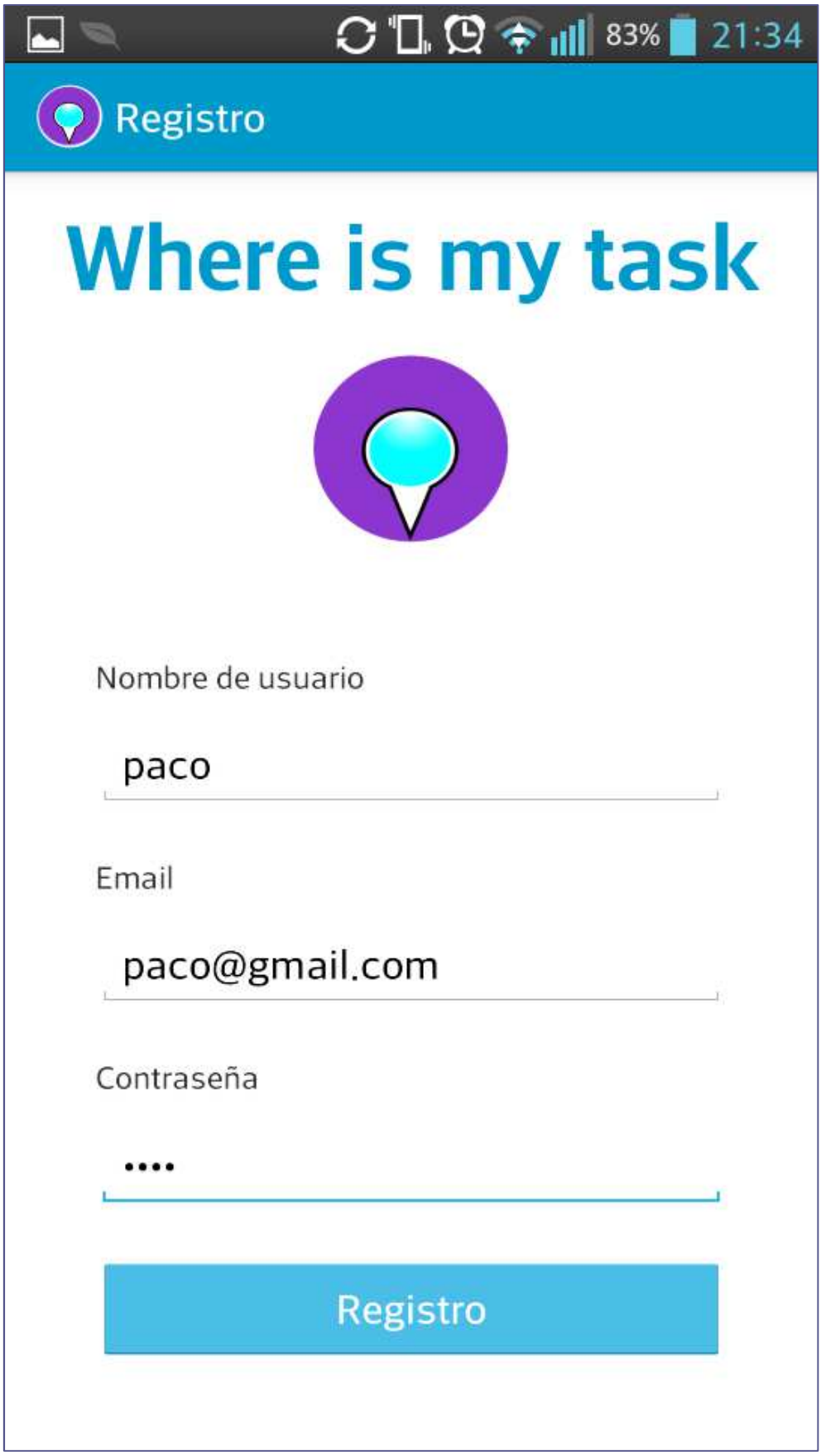


Ilustración 45 - Pantalla registro

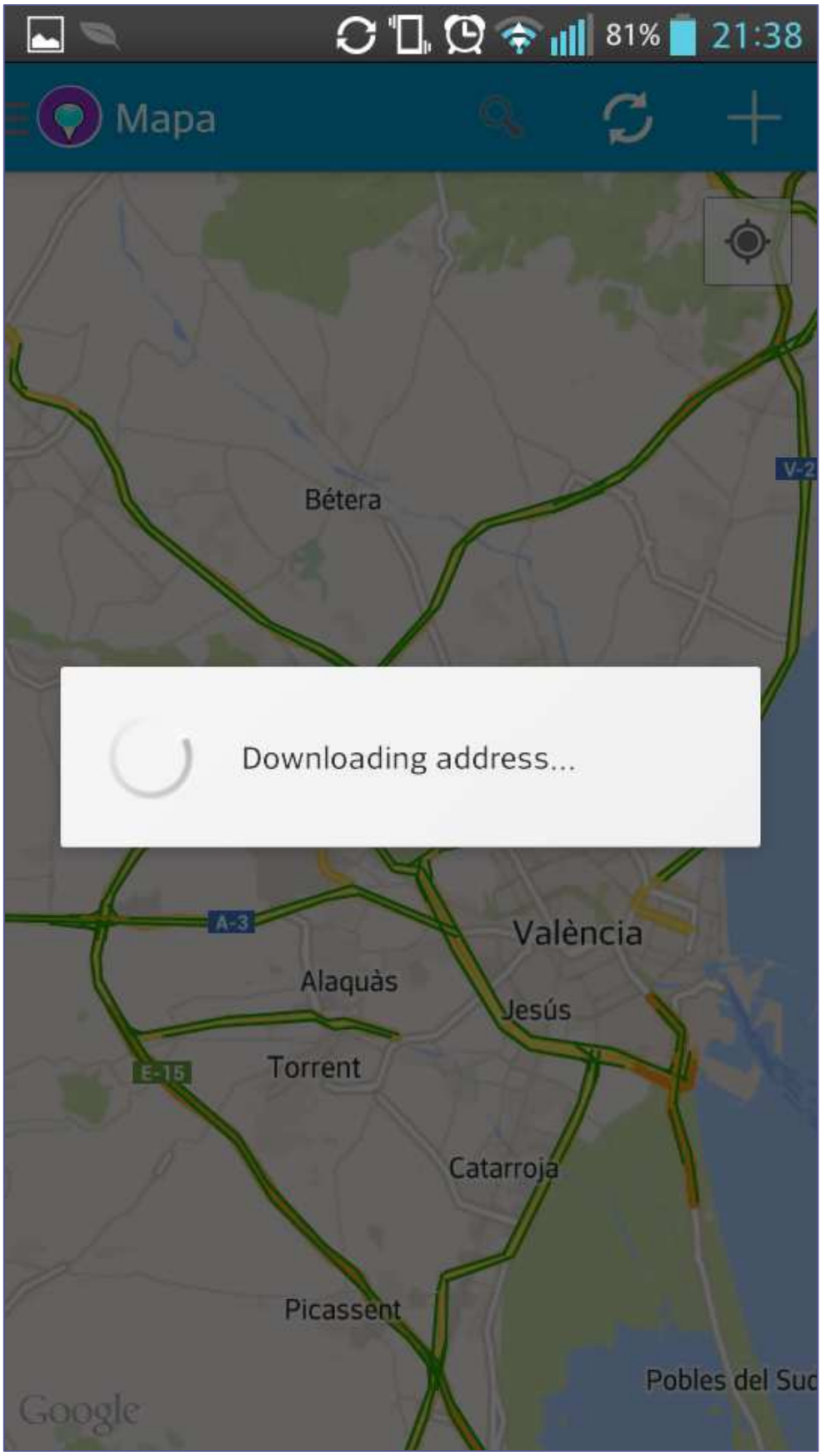


Ilustración 46 - Sincronización de datos

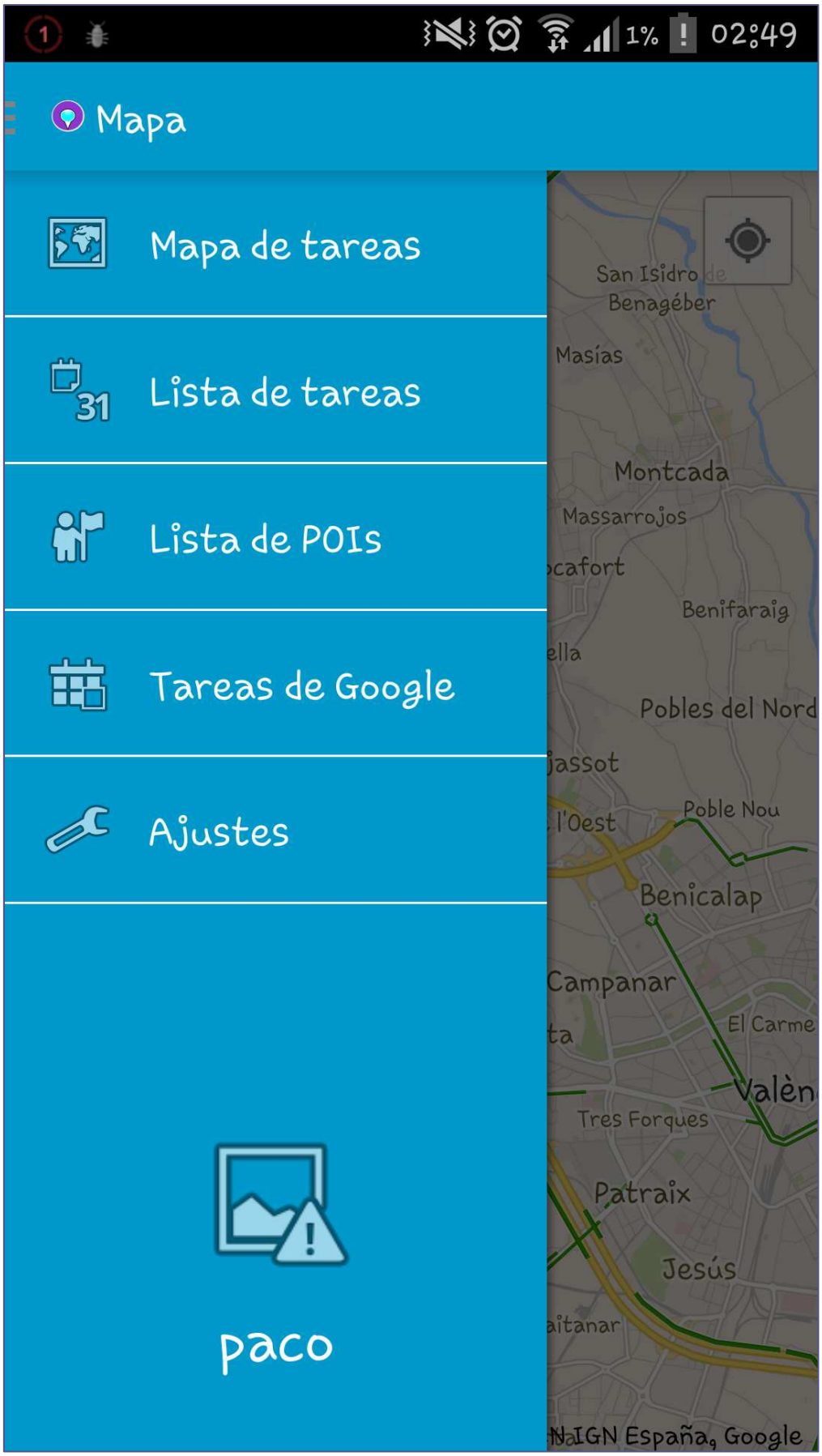


Ilustración 37 - Menú principal

Ilustración 38 - Visualización de mapa

Ilustración 39 - Visualización de tarea seleccionada

Ilustración 40 - Lista de tareas

Ilustración 41 - Tarea seleccionada

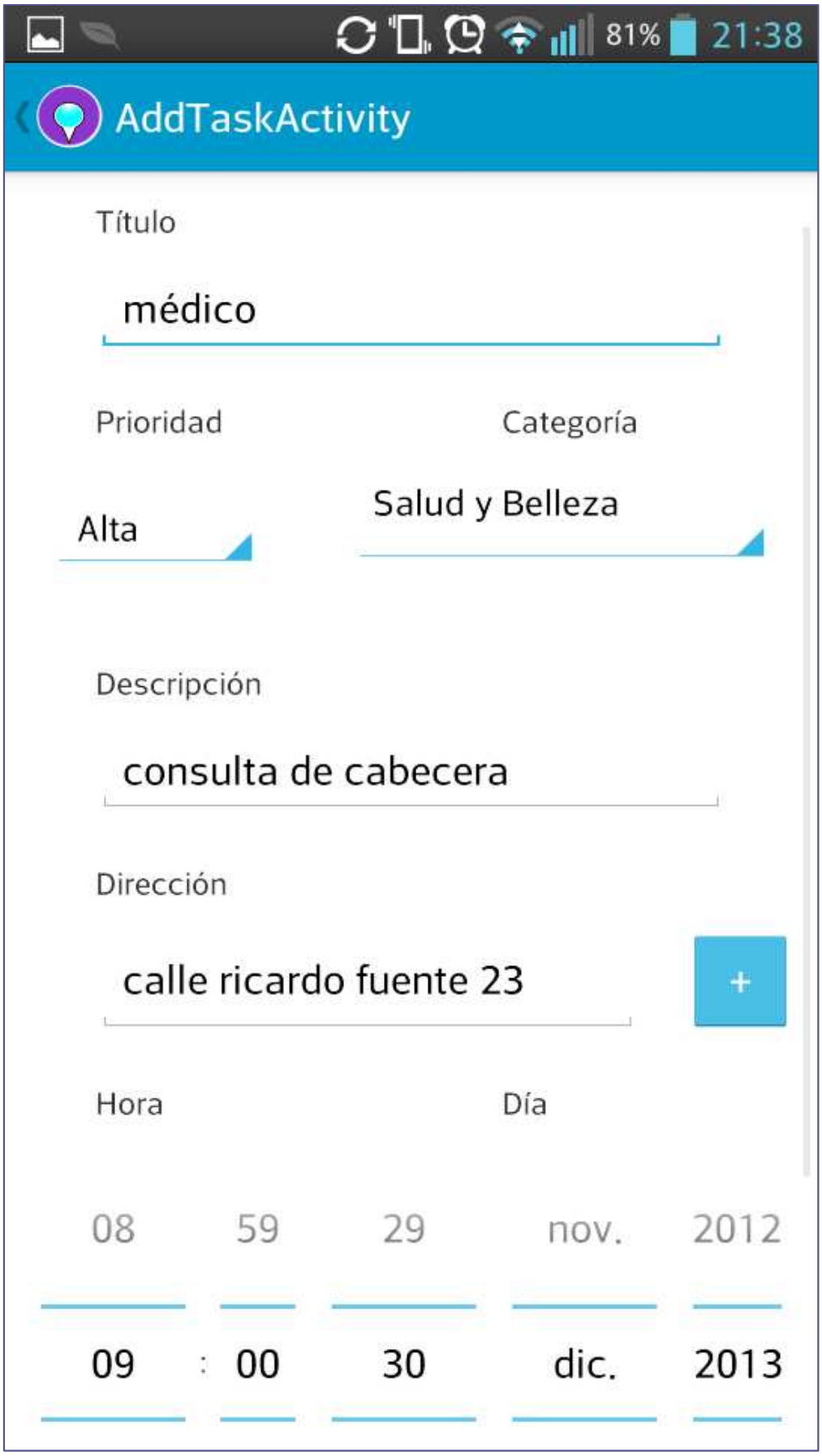


Ilustración 42 - Añadir tarea

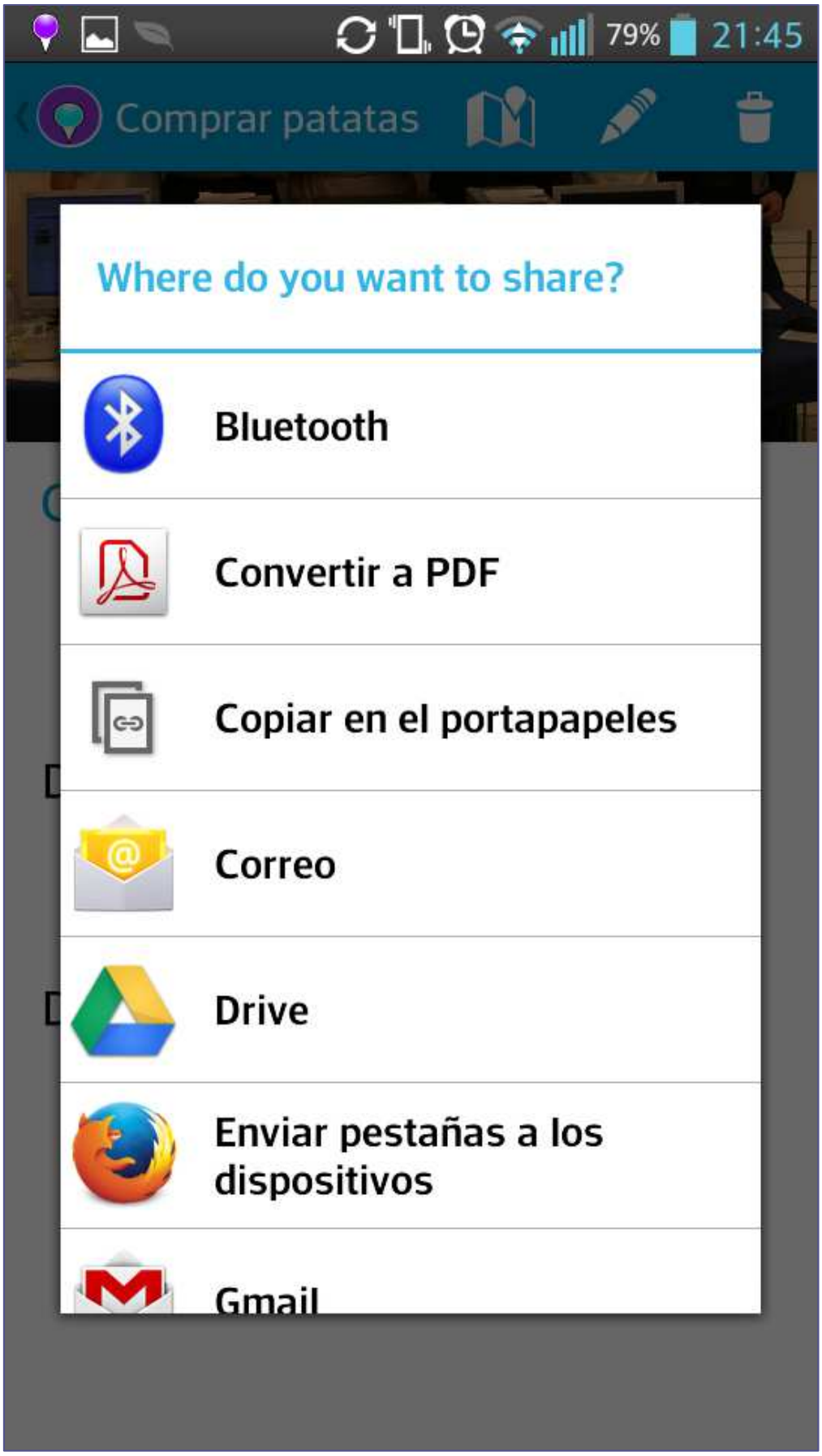


Ilustración 43 - Compartir tarea

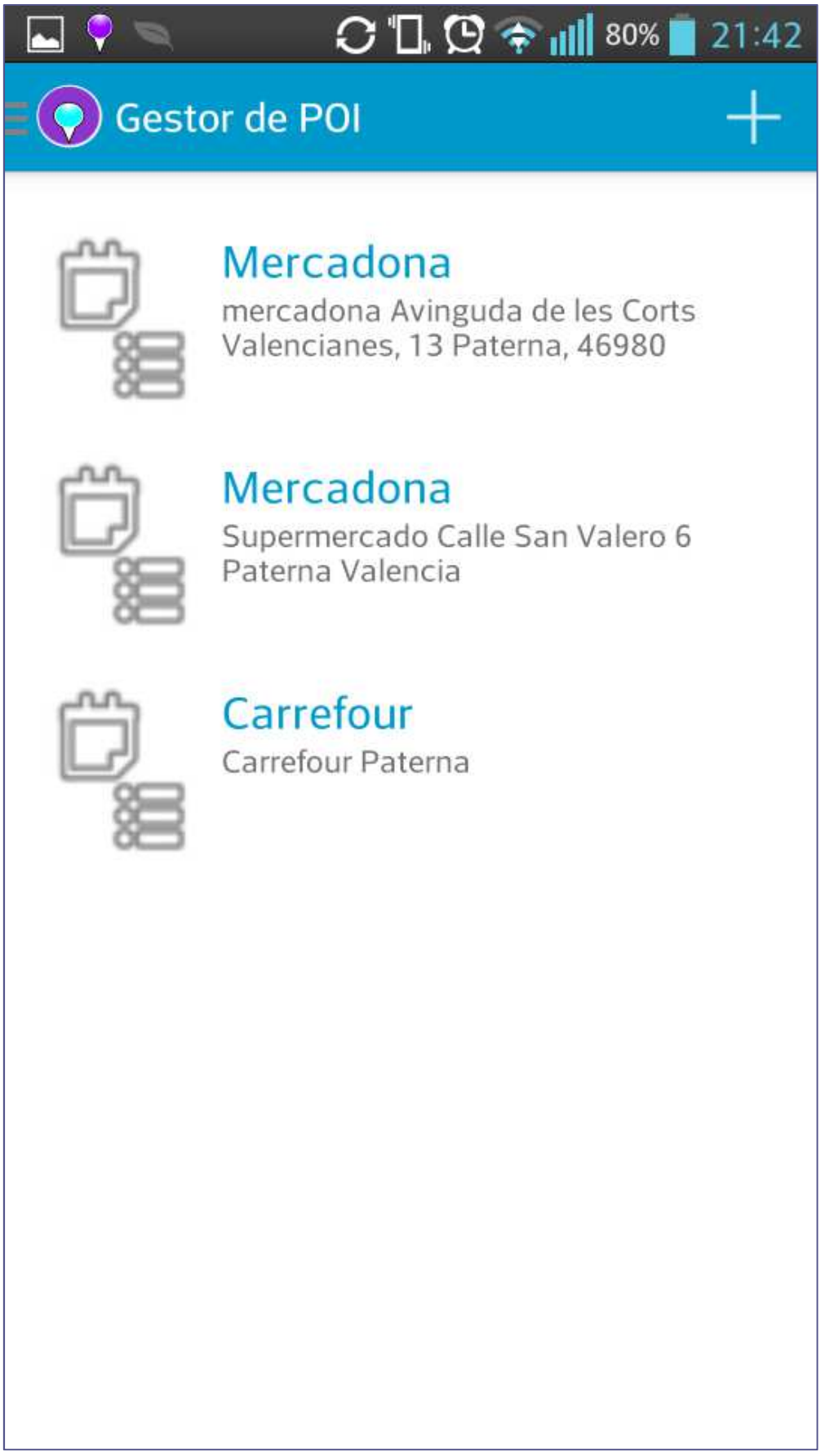


Ilustración 44 - Lista puntos de interés

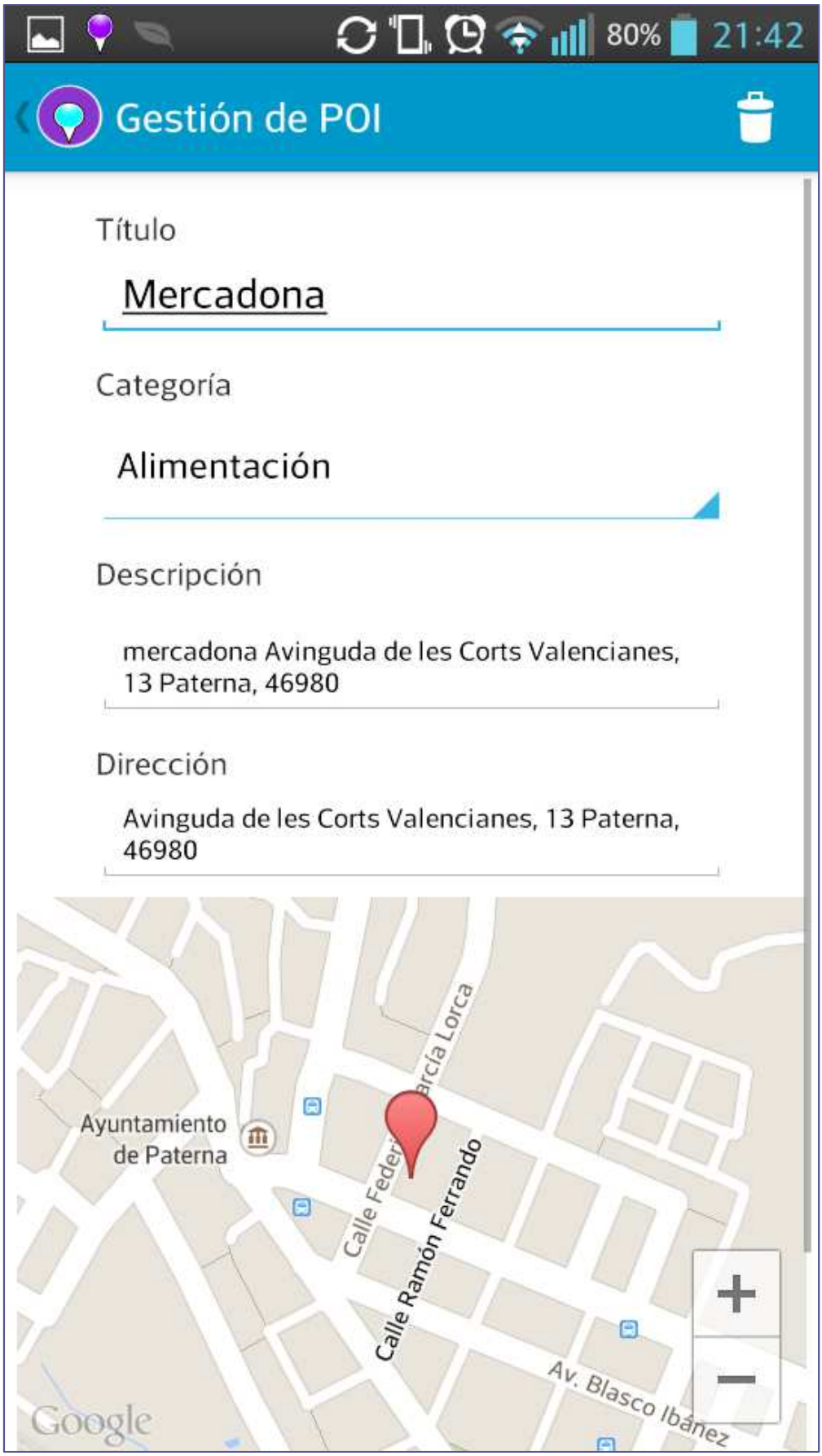


Ilustración 45 - Gestión punto de interés

Ilustración 46 - Sincronizar con Google Tasks

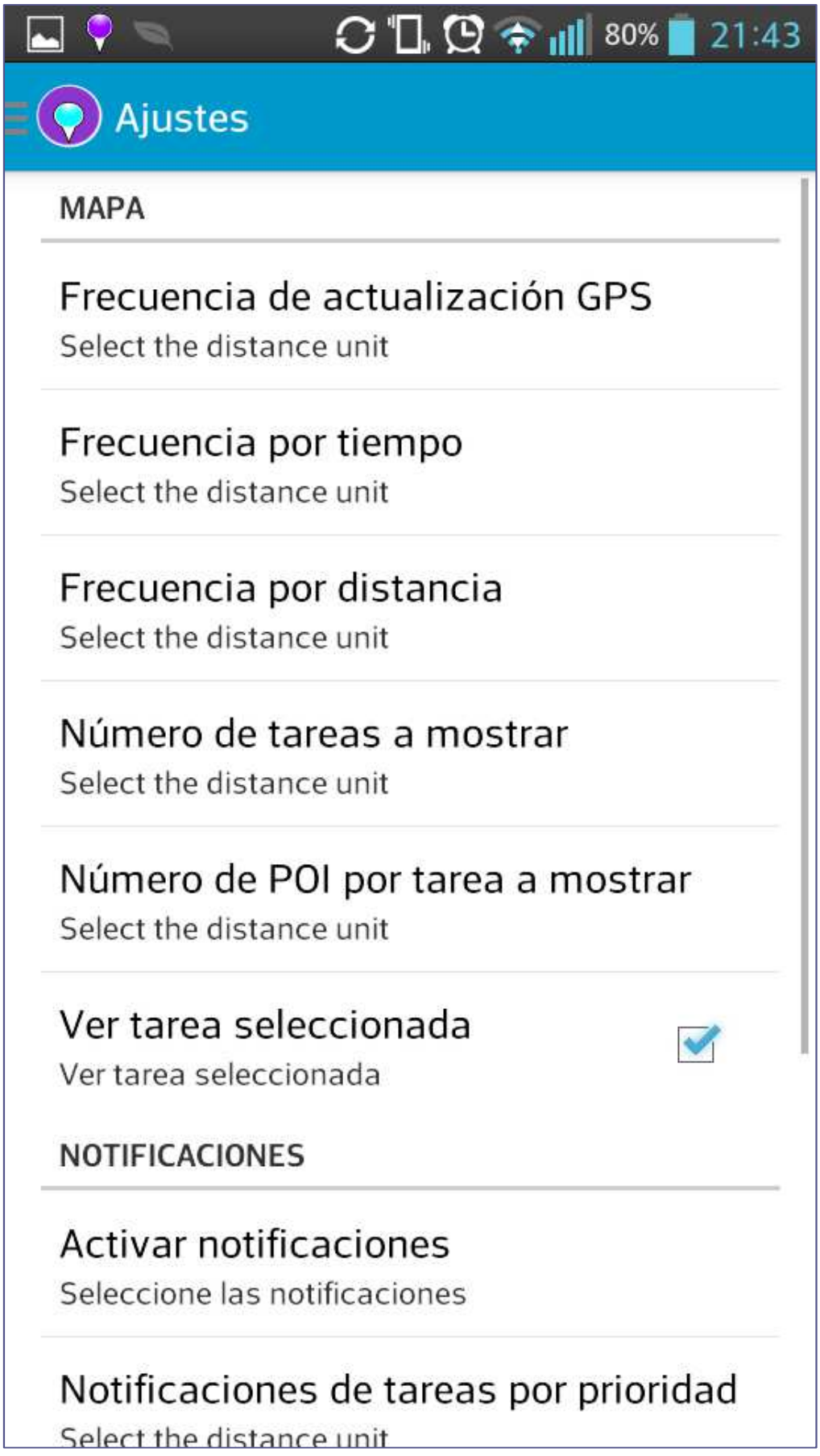


Ilustración 47 - Ajustes

Ilustración 48 - Búsqueda de tarea

10 Pruebas

En este apartado se va a comentar las diferentes pruebas realizadas por mi el resto de pruebas están definidas en la memoria de mi compañera. La gran mayoría han sido desarrolladas sin problemas aunque ciertas funcionalidades han dado ciertos problemas, ellas son:

10.1 ***Sincronización con Google Task:*** A través, de una tarea programada conecta con la API de Google y sincroniza (descarga, y si el usuario lo desea, sube) las tareas del servidor. El problema se encontró cuando se intentaba mantener la sesión iniciada sin atención de cliente, para poder solucionarlo se tuvo que implementar con el modo offline de Google y que registra el Access token que permite acceder a la sesión como un tercero.

10.2 ***Problemas con el uso de la batería:*** Al realizar las pruebas se comprobó que el consumo de batería por parte del GPS es muy elevado, para solucionar esto se ha implementado un algoritmo que calcula el tiempo de refresco del GPS en función de la trayectoria recorrida por el usuario.

11 Conclusiones

Tras la realización de este proyecto se evalúa el trabajo realzado y el nivel de cumplimiento de objetivos. Tal como se especificó en el apartado de objetivos se procede a detallar el nivel de implementación llevado a cabo.

La implementación del mapa y posición geográfica del usuario ha sido una de las funcionalidades más importante y costosas de realizar. En concreto la visualización de los marcadores adecuados a cada momento. A pesar de ello se ha conseguido la funcionalidad esperada con añadidos como un algoritmo propio de actualización del GPS.

Otro de los objetivos más costosos fue la sincronización con el gestor de Google Tasks, por el modo de trabajo offline sin intervención del usuario. Pero finalmente se consiguió solucionar, e incluso habilitar al usuario a subir sus tareas, lo cual proporciona mayor integración.

La gestión de la información en las base de datos no fue tan difícil pero si costosa como se esperaba. La inserción de información genérica para el uso de la aplicación (categorías y puntos de interés) se ha reducido al mínimo para realizar pruebas por falta de tiempo.

El diseño y posterior implementación de la capa REST se ha realizado tal y como se había esperado debido a la experiencia previa en la asignatura de IAP (Integración de aplicaciones) y otros proyectos.

El sistema de notificaciones ha sido costoso pero es asumible porque aún no teniendo conocimientos previos se ha conseguido dar funcionalidad a la notificación requerida e incluso se ha desarrollado otra notificación no esperada.

Por todo ello, la planificación inicial a pesar de haberse complicado los objetivos requeridos por problemas externos y no contemplados, algunos puntos representados en el diagrama cronológico no se han cumplido en los plazos esperados. Esta experiencia nos ha aportado una visión realista de un proyecto de mayor envergadura a los realizados, también nos aportado un punto de vista más cauteloso en las planificaciones de tiempo. En general nos ha sido una experiencia gratificante y nos ha contribuido conocimientos muy útiles cara a nuestro fututo laboral.

12 Ideas de ampliación

Como ideas de ampliación por parte del grupo para una versión más profesional y completa que la actual se ha pensado las siguientes funcionalidades. A continuación se detallaran cada una de ellas.

- Desarrollo de un sistema de sincronización push-up para descarga de tareas al dispositivo tras sincronizarse con otros gestores.
- Implementar una versión móvil o aplicación en otras plataformas para cubrir un mayor rango de clientes.
- Realizar la sincronización con gestores adicionales (Evernote,…).

13 Bibliografía

Amaro, J. E. (2012). *El gran libro de programación avanzada con Android.* Marcombo.

Tomás, J. (2013). *El gran libro de Android.* Marcombo.

V.A. (s.f.). *PHP*. Obtenido de http://php.net/

V.A. (s.f.). *MySql*. Obtenido de http://www.mysql.com/