ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

INGENIERÍA TÉCNICA INFORMÁTICA DE GESTIÓN

AGENDA BASADA EN GEOLOCALIZACIÓN

Realizado por

Juan Antonio García Lucena

Dirigido por

José Ramón Portillo Fernández

Departamento

Matemática Aplicada I

Junio 2013

Índice

RESUMEN

Esta documentación corresponde a la memoria del Proyecto Fin de Carrera, agenda basada en geolocalización para dispositivos Android. Desarrollado para la titulación de Ingeniería Técnica en Informática de Gestión de la Universidad de Sevilla.

Se incluye, además de una introducción al sistema operativo de Android, todos los pasos realizados para lograr el objetivo, desde el nacimiento de la idea hasta completar su desarrollo, incluida la publicación en la tienda Google Play de Android y la puesta a disposicion de todos los usuarios.

Ademas de esta documentación, adjuntamos el código fuente de la aplicación en un fichero a parte, así como su ejecutable listo para ser instalado en un terminal Android.

AGRADECIMIENTOS

No quisiera dejar pasar la oportunidad de agradecer a todas aquellas personas que siempre estuvieron ahí, tanto en los buenos como en los malos momentos. A mi madre, a mi padre y a mi hermano por estar siempre encima mía preocupándose por mi. A Mario y Julio por esos ratos en la calle Faura que tan bien nos vienen. A mis compañeros de clase y amigos Iván, Eduardo y Andrés porque sin ellos las larguísimas sesiones de biblioteca no hubieran sido tan llevaderas. Muchas gracias a todos.

1. INTRODUCCIÓN

1.1. Motivación

Desde la aparición de los primeros sistemas informáticos, siempre se tuvo como objetivo la construcción de unos de nueva generación más pequeños y más potentes. Actualmente, y cada vez con menor intervalo de tiempo, se lanzan al mercado nuevos dispositivos con mucha más potencia que la generación anterior. Entre estos dispositivos nos encontramos con los teléfonos móviles. Anteriormente destinados exclusivamente a realizar y recibir llamadas telefónicas, actualmente esta característica es, curiosamente, la que menos en cuenta se tiene (se considera algo intrínseco en el terminal).

Con el nacimiento de los smartphones también nació una nueva forma se usar nuestros teléfonos y también nació el desarrollo de aplicaciones para este tipo de terminales.

Tan extendido está el uso de este tipo de terminales, que hoy en dia podemos encontrar cientos de miles de aplicaciones de todo tipo para nuestros dispositivos en los diferentes markets de aplicaciones (AppStore, Play Store, etc).

Cada vez hay más gente que se sube al barco del desarrollo de aplicaciones para dispositivos móviles. No es para menos, dado que cualquiera, sin necesitar de caros equipos de desarrollo, puede subir sus propias aplicaciones a los markets y obtener fama, beneficios económicos, etc.

Dado que consideramos que el mundo del desarrollo de aplicaciones para dispositivos móviles va a ser una de las piezas más importantes en el mundo de desarrollo software del futuro, con el desarrollo de GeoAgenda queremos dar un primer paso para aprender la idiosincracia de este tipo de desarrollo software y en un futuro ir mejorando nuestras habilidades.

1.2. Estructura de la memoria

A continuación detallaremos cada uno de los apartados de esta memoria, para aportar una idea global sobre la temática que se tratará en cada uno de ellos.

- En lo que resta del capítulo uno realizaremos una pequeña descripción del proyecto y estableceremos cual va a ser el objetivo principal de nuestra aplicación.
- En el capítulo dos realizaremos un pequeño estudio de los orígenes de las aplicaciones para la gestión de nuestro tiempo a través de varios ejemplos. Finalmente comentaremos cual ha sido nuestra aportación con el desarrollo de nuestro proyecto.
- En el capítulo tres enumeraremos todos los objetivos que hemos querido alcanzar a la hora de desarrollar GeoAgenda, realizaremos una pequeña descripción de ellos y llegaremos a la conclusión de si hemos alcanzado dicho objetivo o no.
- En el capítulo cuatro describiremos los diferentes tipos de requisitos que se han planteado para el correcto desarrollo de GeoAgenda.

- En el capítulo cinco realizaremos un exhaustivo análisis de las partes más importantes del proyecto y que hemos pensado que merecen una explicación más detallada al respecto.
- En el capítulo seis analizaremos desde todos los puntos de vista el coste del proyecto.
- En el capítulo siete llevaremos a cabo una serie de pruebas a GeoAgenda para comprobar que funciona correctamente.
- En el capítulo ocho desarrollaremos un manual de usuario para que al usuario no se le escape ningún detalle.
- En el capítulo nueve analizaremos las posibles alternativas a GeoAgenda actualmente existente en Play Store.
- El siguiente capítulo, el 10, estará dedicado a todas aquellas conclusiones que hayamos sacado del desarrollo de GeoAgenda y todos aquellos desarrollos futuros que se le podrían aplicar.
- Seguidamente, en el capítulo 11, tendremos dos apéndices en el que explicaremos grosso modo dos asuntos bastante importantes que han surgido mientras desarrollabamos GeoAgenda.
- Finalmente el último capítulo estará dedicado a mostrar toda la información bibliográfica que hemos consultado para el desarrollo de GeoAgenda.

1.3. Rápida descripción del proyecto

La aplicación a desarrollar, GeoAgenda, recordará al usuario los eventos que tiene registrado en el calendario para ese día. GeoAgenda dispone de su propio calendario de eventos como, cualquier otra aplicación similar ya existente en Google Play. Para cada evento, el usuario debe proporcionar un título, dirección, ciudad, hora de inicio, tono de alerta y un recordatorio.

Tal y como su nombre indica, GeoAgenda, calculará nuestra posición en caso de llegar un recordatorio de evento y mostrará por pantalla la ruta a seguir para poder alcanzar el lugar de la celebración del evento, que es la que introdujimos previamente en la creación del evento.

1.4. Objetivo principal

El objetivo principal de GeoAgenda es avisar al usuario de la cercanía en el tiempo de la celebración de un evento que previamente ha registrado en la aplicación. Otro objetivo de GeoAgenda es ofrecer un funcionamiento simple, detallado y rico en posibilidades al usuario para que éste pueda organizar su tiempo de la forma mas eficiente posible.

2. ANÁLISIS DE ANTECEDENTES Y APORTACIÓN REALIZADA

En esta sección haremos enfoque en todas aquellas aplicaciones que de una manera u otra han hecho que acabáramos desarrollando una aplicacion como GeoAgenda. Estas aplicaciones no tienen por qué ser estrictamente diferentes a la nuestra, de hecho puede que existan cientos de aplicaciones como la que estamos desarrollando, pero como todos los desarrolladores buscamos, intentaremos dejar nuestro sello personal en ella intentando aportar siempre algo nuevo o mejorar lo ya existente.

2.1. Antecedentes

Muchas formas ha habido a lo largo de la historia de poder organizar nuestro tiempo para así poder optimizarlo lo máximo posible. Como ejemplos tenemos los calendarios, las PDA's, agendas electrónicas, etc. En esta memoria no vamos a hacer un análisis de las metodológias de organización del tiempo desde la edad de piedra, por lo que nos centraremos solamente en aplicaciones que nos han inspirado para el desarrollo de GeoAgenda

2.1.1. Glympse

Con glympse podemos compartir nuestra ubicación en tiempo real durante un tiempo determinad a través de WhatsApp, email, SMS, Twitter, Facebook, etc. Muestra tambien el tiempo que se tarda en llegar a un lugar marcado como destino, indicándonos también un mapa con la ruta a seguir. Es una aplicación bastante completa y fácil de utilizar. El único inconveniente es que su calendario es independiente y no puede sincronizarse con ningún otro.

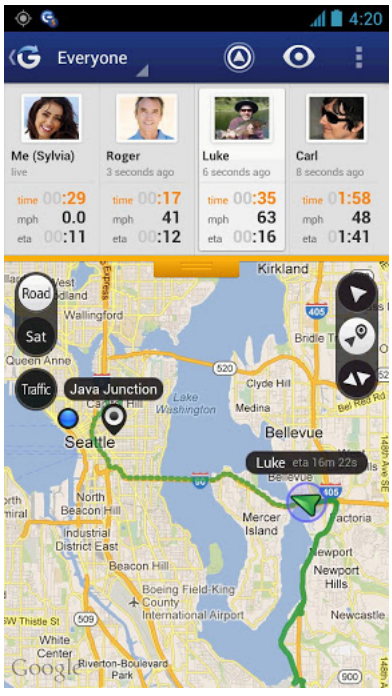

Figura 1: Glympse, mapa.

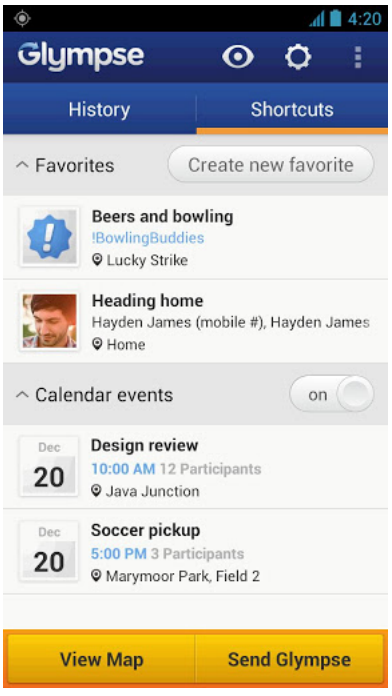

Figura 2: Glympse, eventos.

2.1.2. ArrivAlert

Aplicación desarrollada por Juan León Padilla como proyecto de fin de carrera en diciembre de 2012. Es una aplicación que mantiene informado al usuario de los eventos registrados en su calendario, avisando a los invitados a los eventos si el usuario llega tarde. Se trata de una aplicación bastante útil pero quizás su interfaz de usuario podría ser mejorable dado que es fácil perderse entre las diferentes pantallas.

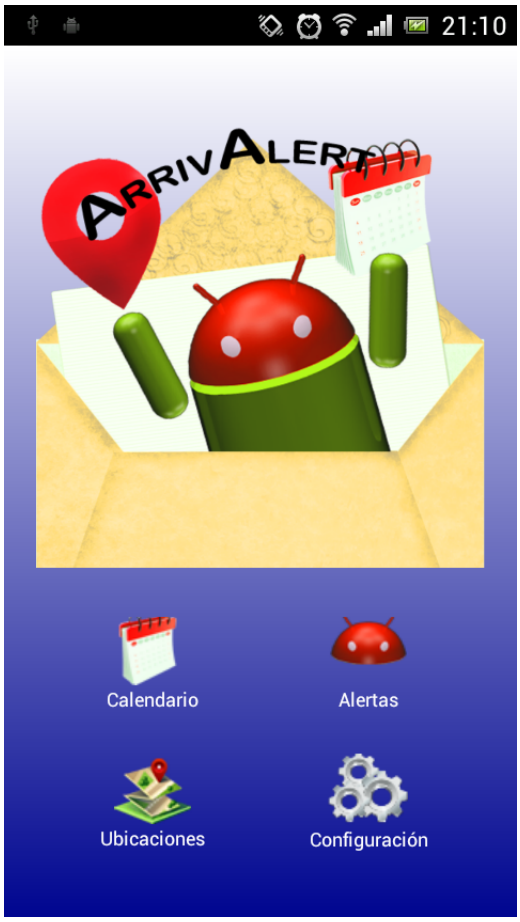

Figura 3: ArrivAlert, Menú.

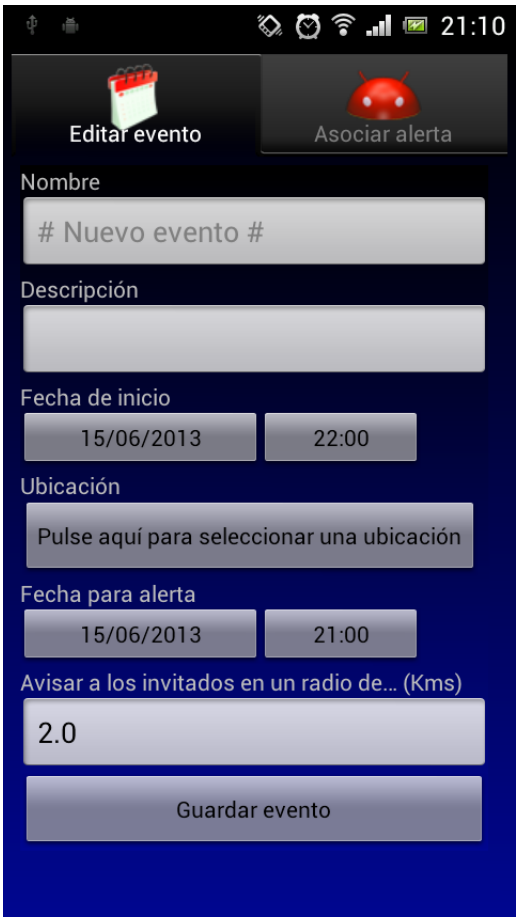

Figura 4: ArrivAlert, eventos.

2.1.3. GeoTask

Aplicación desarrollada por Francisco Javier Martín Otero como proyecto de fin de carrera en junio de 2012. Aplicación que puede realizar una serie de tareas de modo automático, al cumplirse alguna condición previamente establecida, sin necesidad de que el usuario tenga que manipularla. Nuevamente nos encontramos con una aplicación con gran potencial pero que la interfaz de usuario hace bastante complicada la usabilidad de la misma. Aún así tanto GeoTask como ArrivAlert han sido de grandísima ayuda a la hora de desarrollar nuestro proyecto.

Figura 5: GeoTask, perfil.

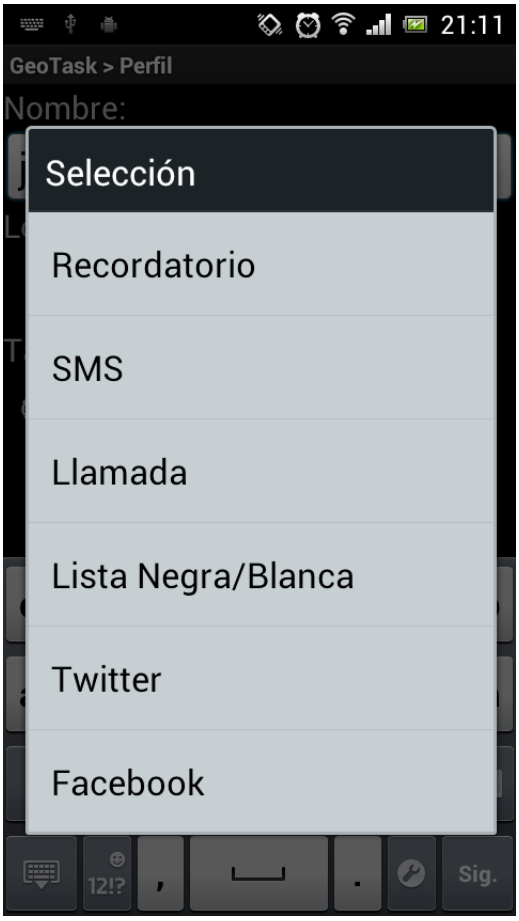

Figura 6: GeoTask, evento.

2.1.4. egoTimer

Aplicación gratuita que consiste en un organizador personal que permite administrar el tiempo de una forma eficaz. Con esta aplicación podemos sincronizar los calendarios de Google Calendar, invitar a contactos a las tareas que vayamos creando, añadir recordatorios, etc. Posee una interfaz de usuario bastante atractiva que hace que egoTimer sea realmente disfrutable al ser usada. Su única pega es que no tiene ninguna funcionalidad de geolocalización.

Figura 7: egoTimer, WeekView.

Figura 8: egoTimer, dayView.

2.2. Aportación realizada

A la hora de realizar los pequeños análisis de las aplicaciones anteriores, se puede observar que hemos hecho bastante hincapié en la interfaz de usuario. Esta interfaz de usuario es lo que hemos tenido más en cuenta a la hora de desarrollar GeoAgenda, debido a que no es la primera vez que ocurre que un proyecto brillante se queda en el olvido por culpa de tener una interfaz de usuario deficiente. Es bastante importante que esta sea vistosa y fácil de usar. Por ejemplo, ArrivAlert, es un proyecto bastante útil, pero mi experiencia con ella podría haber sido mas satisfactoria si la interfaz de usuario estuviera mejor desarrollada.

En definitiva, creemos que hemos aportado un proyecto que está bastante lejos de ser pretenciosa a la hora de incorporar mil y una funcionalidades pero, que si esta bastante cerca de ser una aplicación bastante vistosa y fácil de usar respecto a las aplicaciones anteriormente analizadas.

GeoAgenda también aporta la posibilidad de alertar al usuario de que puede llegar tarde a un evento una hora antes de que llegue la hora del recordatorio, por lo que es una aplicación en la que intentamos asegurar que el usuario llegue a tiempo a todos sus eventos programados.

3. DEFINICIÓN DE OBJETIVOS

En este capítulo enumeraremos y haremos una pequeña descripción de todos los objetivos que hemos querido conseguir con el desarrollo de GeoAgenda.

El objetivo principal es, sin duda, es el desarrollo de una aplicación que sea capaz de registrar nuestros eventos, nos notifique cuando llegue el momento y sea capaz de mostrar la ruta desde nuestra posición actual, pero para poder cumplir con este gran objetivo, antes hemos tenido que obtener éxito en otros objetivos más básicos como son los siguientes:

1. **Estudiar y comprender** el entorno de desarrollo de Android.
2. **Aprender a desarrollar** aplicaciones sencillas para Android de forma nativa.
3. Desarrollar una aplicación que no sea susceptible a **errores**.
4. Desarrollar una aplicación con una **interfaz de usuario** simple, fácil de manejar pero rica en posibilidades.
5. Calcular la **posición actual.**
6. Mostrar **rutas** a partir de dos puntos en el mapa.
7. Integrar **peticiones REST** en Android.

3.1. Estudiar y comprender

Este es quizás el más trivial de los objetivos, dado que si no supieramos manejarnos con el entorno de desarrollo de Android no seríamos capaces si quiera de realizar un Hello World. Al principio nos pareció un mundo y nos costó bastante trabajo adaptarnos al entorno pero una vez adaptado a él es fácil comprender algunas cosas. Por poner un ejemplo, la clase R fue una de las cosas que más extraña nos parecía, pero cuando comprendimos que es una clase que se genera automáticamente y sirve para poder hacer referencia a los recursos de la aplicación (imágenes, localización de la aplicación, etc) pudimos empezar a desarrollar la aplicación con bastante soltura.

3.2. Aprender a desarrollar

Puede que este objetivo y el anterior se parezcan un poco pero cuando nos referimos a aprender a desarrollar lo que intentamos decir es que cada plataforma tiene sus peculiaridades y cuando realizamos nuestras primeras investigaciones somos completamente ajenos a ellas. Hay diferentes formas de desarrollar aplicaciones para todos los dispositivos, de forma nativa y no nativa.

Cuando desarrollamos de forma no nativa, cuando la aplicación se despliega lo que obtenemos es una emulación de dicha aplicación, lo que hace que no se aprovechen todos los recursos del dispositivo. Gracias a mi corta experiencia laboral, conocía como se desarrollaba de forma no nativa con diferentes frameworks pero para la realización de este proyecto consideramos que sería más positivo aprender a programar de verdad para dispositivos Android dado que así aprenderíamos cosas nuevas y tendríamos un abanico más amplio de experiencia en el desarrollo Android.

3.3. Errores

Uno de los principales objetivos que hemos intentado conseguir es que la aplicación no sea susceptible a errores. Aún así, durante varias comprobaciones hemos sufrido de algún que otro error, pero la tónica general de la aplicación es, primero, detectar errores y, segundo, gestionarlas de forma de el usuario se vea afectado lo menos posible.

3.4. Interfaz de usuario

No es la primera vez que se menciona (ni será la ultima). Nuestro objetivo es que el usuario tenga una experiencia satisfactoria a la hora de hacer uso de GeoAgenda y para ello requerimos de una interfaz de usuario que sea atractiva para este y que no necesite de ningún manual de instrucciones para empezar a disfrutar de la aplicación.

3.5. Posición actual

Uno de los pilares de la aplicación es calcular nuestra posición actual. Como más tarde describiremos en el capítulo 5, no ha sido una tarea intuitiva y, en cierto modo, tampoco ha sido sencilla dado que en un primer desarrollo no actualizaba la posición actual si no la que ocupabamos la última vez que la calculamos, es decir, no se actualizaba nuestra posición. Solucionar este error no ha sido tan difícil como se pensó en un principio pero finalmente conseguimos solucionarlo.

3.6. Rutas

El segundo pilar de la aplicación. Una vez calculada nuestra posición actual el segundo paso era calcular la ruta hasta nuestro destino. Como también explicamos más detalladamente en el capítulo 5, estuvimos dándole muchas vueltas a la forma de conseguir representar estas rutas hasta que llegamos a la conclusión que, creemos que era la mejor solución de todas las posibles que se nos planteaban.

3.7. Servicios REST

Es bastante importante saber realizar peticiones a servicios web (SOAP o REST), dado que la mayor parte de las aplicaciones, la comunicación entre el lado servidor de estas y el lado cliente (lo que el usuario manipula), se realizará casi exclusivamente con este tipo de servicios. Aunque no se hace mucho uso de este tipo de peticiones en la aplicación (solamente para calcular el tiempo que se tarda en llegar desde un punto a otro en coche), ha sido importante aprender a implementar este tipo de soluciones para un futuro uso.

4. ANÁLISIS DE REQUISITOS, DISEÑO E IMPLEMENTACIÓN

GeoAgenda es una aplicación basada en eventos. Estos eventos serán introducidos por el usuario y son un requisito indispensable para el objetivo principal de GeoAgenda, que no es otro que un gestor de tareas.

Por un lado tenemos una aplicación que, desde sus primeros días de desarrollo, pretende ser una aplicación completa y fácil de usar. Para ello se han considerado varias alternativas de interfaz gráfica, desde pestañas (tabs) hasta un simple tablero con iconos. Ha sido por esto último por lo que nos hemos decidido finalmente por ser algo fácilmente reconocible y de fácil usabilidad para el usuario.

Por otro lado se ha querido hacer también lo más "usable" posible tanto la lista de eventos como la creación, edición y eliminación de éstos. Para ello nos ayudamos de la nueva interfaz "Holo" que Android nos ofrece desde su versión 3, en tablets, y 4, en teléfonos móviles. Los elementos del menú más importantes se encuentran ahora en la llamada "ActionBar" que no es mas que una pequeña barra en la parte superior de la pantalla (tal y como vemos en la figura 7. Esta ActionBar también nos permite navegar hacia atrás en la aplicación, aunque seguiremos teniendo disponible el botón fisico "back" de nuestro terminal.

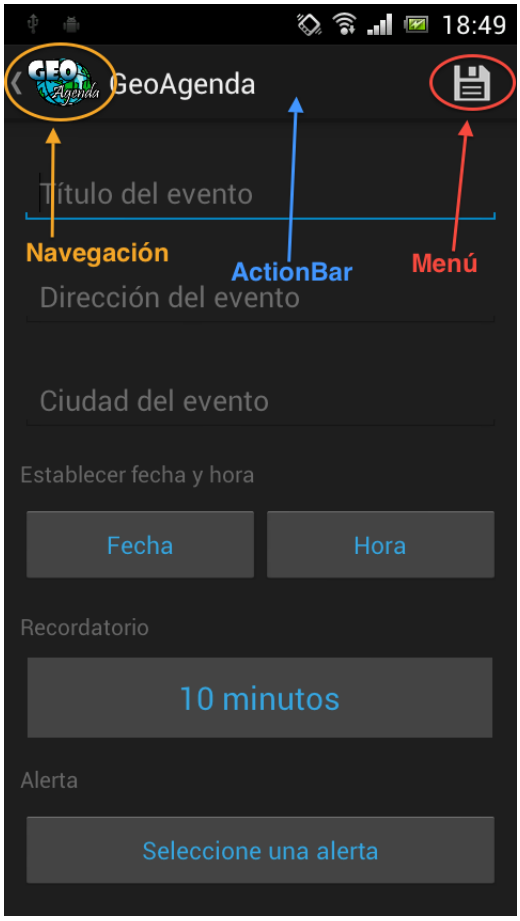

Figura 9: ActionBar

Funcionalmente hemos decidido que el elemento principal de la aplicación sean los eventos porque es el que contendrá toda la información necesaria para que el resto de funcionalidades se ejecuten, como las alertas, los mapas, etc. En los siguientes apartados explicaremos mas detalladamente todos los requisitos necesarios para el correcto funcionamiento de GeoAgenda.

4.1. Requisitos de información

GeoAgenda debe aportar toda la información guardada en un evento cuando llegue el momento en el que el usuario definió previamente. Entre otras los datos más importantes que debe mostrar la aplicación al usuario son las siguientes:

- **Eventos:** Los eventos están definidos cada uno con un nombre que el usuario define, internamente se distinguirá de otros eventos a través de un identificador único. También debe aportar información sobre el lugar en el que se celebrará dicho evento y la ciudad del mismo, asimismo, el sistema de eventos debe proporcionar también información sobre la fecha y hora del evento, un recordatorio, que no es más que definir cuanto tiempo antes del comienzo del evento quiere el usuario que suene la alarma. Con los eventos también podemos definir que tipo de melodía de alarma sonará cuando llegue el momento.

- **Ubicación:** El sistema, a partir de la información suministrada por el usuario, debe ser capaz de encontrar la ubicación donde se celebrará el evento y crear una ruta para llegar a dicho lugar a partir de la posición actual.

- **Alertas:** La aplicación debe ser capaz de hacer sonar la alarma que nos comunica que un evento esta próximo a comenzar.

Algo a tener en cuenta, a modo de restricción, es que no se podrán guardar eventos que tengan lugar antes de la fecha actual.

4.2. Requisitos funcionales

- **Base de datos:** En la base de datos se guardarán todos los eventos que el usuario cree, tanto si hay conexión a internet como si no.

- **Localización:** La aplicación debe suministrar información sobre la posición actual del usuario.

- **Alertas:** En caso de la proximidad de un evento, la aplicación debe ser capaz de alertar al usuario.

- **Proximidad:** GeoAgenda debe ser capaz de dar datos sobre la ubicación del usuario y su destino con un margen de error aceptable.

4.3. Requisitos no funcionales

- **Usabilidad:** La aplicación debe tener una interfaz simple pero a la vez atractiva y completa para facilitar la usabilidad a todo tipo de usuarios.

- **Software Libre:** GeoAgenda debe basarse en estándares y tecnologías libres.

- **Portabilidad:** La aplicación debe ser visionada perfectamente en todo tipo de dispositivos moviles con sistema operativo Android y con diferentes resoluciones

- **Calidad:** La aplicación debe ser robusta y no ser susceptible a errores.

5. DISEÑO E IMPLEMENTACIÓN

Para el desarrollo de GeoAgenda hemos decidido usar como versión objetivo de Android la 4.0.3 dado que hemos querido tocar, aunque sea solo un poco, la nueva interfaz Holo que proporciona Google. Con dicha interfaz desaparece practicamente el uso del boton físico "back" ya que la navegación hacia atrás se llevará a cabo meddiante un botón que, para todas las aplicaciones, estará en la esquina superior derecha (normalmente con el icono de la aplicación), como podemos ver en la siguiente imagen.

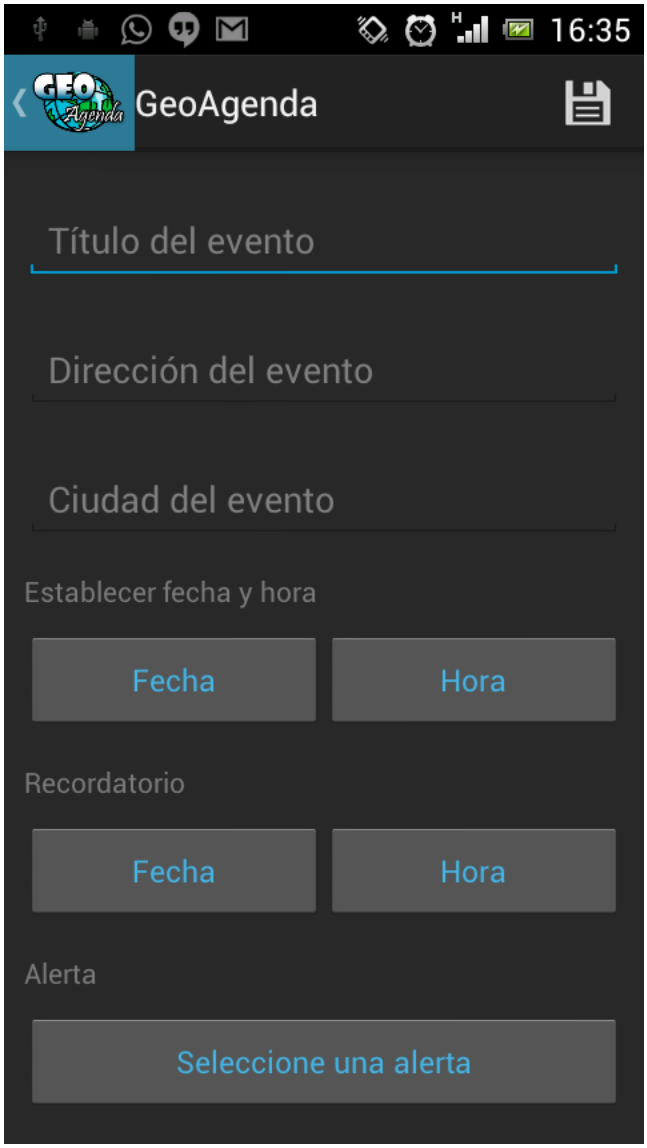

Figura 10: Navegación

5.1. Estructura de clases

La estructura de clases de GeoAgenda esta dividida en tres paquetes para que no suponga un caos a la hora de refactorizar y todo este bien ordenado.

En el primer paquete tenemos lo que son las Activities. Una Activity, grosso modo, es una pantalla en Android. Dentro de cada activity tenemos definidos la interfaz de usuarios y la interactuación con otras Activities. Normalmente para comunicarnos entre Activities usaremos los Intent. Estos Intent a parte de para poder enviar datos entre Activities tambien nos permitiran abrir una Activity desde otra por, por ejemplo, la acción de pulsar un botón.

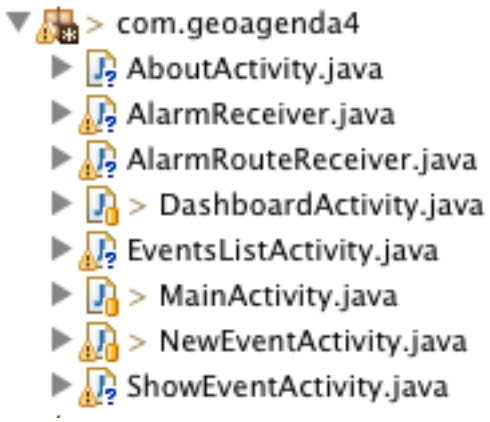

Figura 11: Paquete principal

En el segundo paquete encontramos los llamados Fragments. Un fragment podría definirse como una porción de la interfaz de usuario que puede añadirse o eliminarse de una interfaz de forma independiente al resto de elementos de la actividad, y que por supuesto puede reutilizarse en otras actividades. Los fragment surgen con la versión 3 de Android para tablets para que la aplicación se adaptara al nuevo tipo de pantallas de estos.

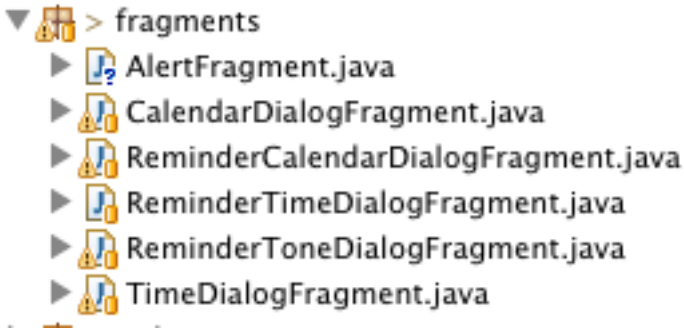

Figura 12: Paquete de fragments

En el ultimo paquete, llamado utils, encontramos todos aquellos componentes que hemos ido necesitando y a medida que ibamos avanzando en el deesarrollo hemos ido creando.

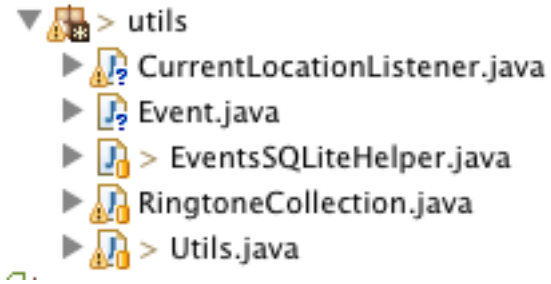

Figura 13: Paquete de utils

5.2. Descripción de las Activities

A continuación pasaremos a realizar una breve descripción sobre la funcionalidad de cada una de las activities de la aplicación.

- AboutActivity: Mostramos la información respecto al desarrollo del proyecto.
- AlarmReceiver: Aunque AlarmReceiver no es una activity al uso conviene hacer una descripción de ella dada la importancia que tiene en el proyecto. Cuando generamos un nuevo evento, este, valga la redundancia, genera una alarma a una hora y fecha determinada. Aqui es donde entra AlarmReceiver ya que va a ser el que cuando llegue el momento de mostrar la alarma va a hacer que suene el tono de alerta y se muestre en el sistema de notificaciones de Android la notificación de que se ha llegado a la hora de recordatorio del evento.
- AlarmRouteReceiver: Estamos en el mismo caso que AlarmReceiver solo que esta alarma se disparará cuando GeoAgenda estime que es posible llegar tarde a un evento.
- DashboardActivity: Es la pantalla principal. Donde se muestran todos los iconos. Su única funcionalidad es la de darnos la posibilidad de acceder las diferentes áreas de la aplicación.
- EventsListActivity: En esta activity se muestran los eventos que ya han sido creados para poder acceder a los detalles de los eventos.
- MainActivity: Es la splashscreen que salta cuando ejecutamos la aplicación.
- NewEventActivity: La activity más importante de toda la aplicación. Esta activity entre otras cosas se encarga de comprobar si el formulario de creación de evento esta correctamente completado, generar las alertas, guardar los eventos creados en la base de datos, etc.
- ShowEventActivity: Esta activity muestra los detalles de los eventos que ya hemos creado. Nos da la posibilidad de eliminar el evento o de acceder al mapa para ver la ruta para llegar al lugar del evento a partir de la posición actual.

5.3. Elementos más importantes del desarrollo

Una vez que hemos descrito de una forma generalizada la funcionalidad de todas las activities, vamos a detenernos en los aspectos más importantes del desarrollo de GeoAgenda

5.3.1. Geolocalización

Calcular la posición actual al principio puede parecer que no es una tarea sencilla o intuitiva. Más allá de estas primeras impresiones, podemos decir que la geolocalización en Android puede no ser intuitiva pero no es una tarea demasiado complicada.

Para calcular la posción actual necesitaremos un proveedor de ubicación que es el que se encargará de proporcionar un conjunto de mediciones relacionadas con la localización, como son la latitud y la longitud.

Necesitaremos también de hacer uso de LocationManager que es la clase principal que usaremos para interactuar con los datos de posición proporcionados. El método de esta clase, getBestProvider(), nos permitirá obtener el mejor proveedor de localización que esté activado para calcular nuestra posición.

La interfaz LocationListener nos permite interactuar con los eventos de localización. En concreto el que más nos interesa es el evento onLocationChanged(Location location). Este evento se dispara cada vez que cambiamos de posición, suministrandonos la latitud y la longitud de la posición que actualmente ocupamos.

En el método requestLocationUpdates de la clase LocationManager definimos cada cuanto tiempo o distancia actualizaremos la posición (los parámetros de tiempo y distancia son milisegundo y metros respectivamente).

El proceso de geolocalización lo podemos ver en la siguiente imagen

```
LocationManager locationManager = (LocationManager) this.getSystemService(Context.LOCATION_SERVICE);
CurrentLocationListener locationListener = new CurrentLocationListener();
locationManager.requestLocationUpdates(LocationManager.NETWORK_PROVIDER, 0, 0, locationListener);

Criteria criteria = new Criteria();
String provider = locationManager.getBestProvider(criteria, true);
Location currentLocation = locationManager.getLastKnownLocation(provider);
if(currentLocation!=null){
    locationListener.onLocationChanged(currentLocation);
}
locationManager.requestLocationUpdates(provider, (long)20000, (float)100, locationListener);
```

Figura 14: Localización

```
private class CurrentLocationListener implements LocationListener{

    @Override
    public void onLocationChanged(Location location) {
        showMap(location);
    }

    @Override
    public void onProviderDisabled(String provider) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onProviderEnabled(String provider) {
        // TODO Auto-generated method stub

    }

    @Override
    public void onStatusChanged(String provider, int status, Bundle extras) {
        // TODO Auto-generated method stub

    }

}
```

Figura 15: LocationListener

```
public void showMap(Location currentLocation){

    Double latitude = currentLocation.getLatitude();
    Double longitude = currentLocation.getLongitude();

    Double latitude_dest = addresses.get(0).getLatitude();
    Double longitude_dest = addresses.get(0).getLongitude();

    Intent i = new Intent(Intent.ACTION_VIEW, Uri.parse("http://maps.google.com/maps?saddr="+latitude+","+longitude+
            "&daddr="+latitude_dest+","+longitude_dest));
    startActivity(i);
}
```

Figura 16: showMap

5.3.2. Coordenadas del destino

No obstante el cometido de GeoAgenda es el de mostrar rutas entre la posición actual y un destino. hasta ahora nos hemos centrado en calcular la longitd y la latitud de la posición que actualmente ocupamos pero, ¿Cómo calcular la latitud y la longitud de nuestro destino?

Como hemos podido ver previamente a la hora de crear un nuevo evento, se requiere que el usuario introduzca la dirección y la ciudad donde se va a celebrar el evento.

En nuestro paquete utils en la clase Utils, entre otras funciones estáticas, podemos encontrar una que se llama getLocationInfo en el que le pasamos como parámetros una dirección. Dentro de este método usamos la función getFromLocationName de la clase Geocoder. Esta función va a ser la que nos proporcionará toda la información necesaria sobre una dirección, entre otras, la latitud y la longitud.

NOTA: Como podemos observar getFromLocationName devuelve una lista de objetos Address. Esto se debe a que el nombre de una dirección puede coincidir con varios lugares en el planeta. En caso de que se devolviera más de una localización se pretendía realizar un menú en el que poder elegir la localización correcta pero, por falta de tiempo en el desarrollo hemos decidido que, llegados a este caso, seleccionaremos la primera opción de todas.

```
public static List<Address> getLocationInfo(String address, Context context){
    Geocoder coder = new Geocoder(context);
    List<Address> addresses = new ArrayList<Address>();
    try {
        addresses = coder.getFromLocationName(address, 5);
        Log.d("locations", "Entra en el try");
        if(address.isEmpty()){
            Log.d("locations", "No existen direcciones");
            Toast.makeText(context, "No existen direcciones", Toast.LENGTH_SHORT).show();
        }
    }catch(IOException e){
        Log.d("locations", "PASO4");
        e.printStackTrace();
    }
    return addresses;
}
```

Figura 17: Coordenadas del destino

5.3.3. ¿Mapa propio o GoogleMaps?

Una de las decisiones quizás más controvertidas a la hora de el desarrollo del proyecto, ha sido si usar un mapa propio o llamar a la aplicación GoogleMaps para representar las rutas en GeoAgenda. A continuación expondremos las principales razones que nos han llevado a seleccionar como mejor solución la llamada a la aplicación GoogleMaps.

El primer motivo que nos llevó a pensar que era la mejor solución es el simple hecho de que GoogleMaps esta presente en todos los dispositivos Android que hay en el mercado ya que viene de serie y, ademas, no puede ser eliminado (a no ser que se sea superusuario).

No vamos a engañar a nadie si decimos que otro de los motivos que nos llevo a usar GoogleMaps es que es más fácil implementar esta solución que la anterior. Si quisieramos pintar una ruta en un mapa propio, necesitariamos conectarnos a la API de GoogleMaps, enviarle nuestra posición actual y la de destino, parsear el JSON o el XML que nos devuelva en busca de una serie de coordenadas (puntos) los cuales tendríamos que situar en el mapa y pintar rayas de uno a otro. En este aspecto se ha querido también cuidar los tiempos de ejecución de la aplicación ya que, supongamos que si la API de GoogleMaps nos devuelve una ruta de 5 puntos, esto supondría pintar 4 lineas. Para pintar las lineas requeriríamos de un bucle for que se recorrería n-1 veces (donde n es el número de puntos), suponiendo que la ruta no es cerrada (el punto de inicio no es el punto de destino). Obtendríamos un tiempo lineal de ejecución que supondría un serio problema en el caso que la API de GoogleMaps nos devolviera un número elevado de puntos. Si embargo la llamada a la aplicación GoogleMaps nos libraría de estos problemas ya que está optimizado para el cálculo de rutas entre dos puntos.

Una consecuencia del anterior punto es la cantidad de código que se necesitaría para pintar rutas en un mapa propio y para llamar a la aplicación de GoogleMaps. Mientras con la primera necesitariamos gran cantidad de código con la segunda opción sólo nos llevaría un par de lineas de código, tal y como podemos ver en la imagen.

```
Intent i = new Intent(Intent.ACTION_VIEW, Uri.parse("http://maps.google.com/maps?saddr="+latitude+","+longitude+
        "&daddr="+latitude_dest+","+longitude_dest));
startActivity(i);
```

Figura 18: Código para abrir GoogleMaps

El último motivo que nos llevó a la utilización de la aplicación de GoogleMaps fue la reutilización de código, uno de los pilares en los que se basa la programación orientada a objeto. No pretendemos revolucionar el mundo de los mapas y la geolocalización, asi que decidimos usar uno de los servicios de mapas más fiables y completos que hay en el mercado actualmente y así aprovecharíamos todas las opciones de GoogleMaps para enriquecer aún más las posibilidades de GeoAgenda.

5.3.4. Sistema de notificaciones

Para explicar el sistema de notificaciones tenemos que explicar antes que la clase broadcast. Un broadcast es un proceso que esta "por detrás", esperando a ser ejecutado. Para poder crear un broadcast necesitaremos que el proceso que vaya a estar esperando a ejecutarse sea "hijo" de la case BroadcastReceiver. Una vez que el broadcast tenga permiso para ejecutarse, se llamará al método onReceive de la clase hija de BroadcastReceiver.

```
public class AlarmReceiver extends BroadcastReceiver{

    private String eventTitle;

    @Override
    public void onReceive(Context context, Intent intent) {

        NotificationManager notificationManager = (NotificationManager) context.getSystemService(Context.NOTIFICATION_SERVICE);
        Intent i = new Intent(context,EventsListActivity.class);
        PendingIntent pendingIntent = PendingIntent.getActivity(context, 0, i, 0);
        //Creamos la notificacion
        Notification notification = createNotification();
        CharSequence cs = intent.getStringExtra("eventTitle");
        CharSequence cs2 = "Evento próximo";
        String notificationTone = intent.getStringExtra("notificationTone");
        Uri u = Uri.parse(notificationTone);
        RingtoneManager rM = new RingtoneManager(context);
        rM.setType(RingtoneManager.TYPE_NOTIFICATION);
        Ringtone r = rM.getRingtone(context, u);
        notification.setLatestEventInfo(context, cs2, cs, pendingIntent);
        //Notificamos
        notification.vibrate = new long[]{100, 100};
        notificationManager.notify(0, notification);
        r.play();
        while(true){
            if(!r.isPlaying()){
                r.stop();
                break;
            }
        }


    }
```

Figura 19: Alertas

```
private Notification createNotification() {
    Notification notification = new Notification();
    notification.icon = R.drawable.geoa;
    notification.when = System.currentTimeMillis();
    notification.flags = Notification.FLAG_AUTO_CANCEL;
    notification.defaults = Notification.DEFAULT_VIBRATE;
    notification.ledOnMS = 1500;
    notification.ledOffMS = 1500;

    return notification;
}
```

Figura 20: Notificaciones

Las imagenes anteriores muestra la clase hija de BroadcastReceiver que sirve para indicar a un usuario que la fecha y hora del recordatorio de un evento ha llegado. Como se puede observar dentro de la clase onReceive, tenemos declarado todo lo que va a pasar cuando este se ejecute. Primero creamos una notificación para mostrarla en la

barra de notificaciones de nuestro terminal Android.y, más tarde, ejecutamos el tono de alerta que el usuario definió cuando creó el evento.

NOTA: El terminal debería también vibrar pero por algún motivo que se nos escapa no lo hace.

Ya hemos creado nuestro broadcast, pero **¿Dónde lo usamos?**

```
Intent in = new Intent(NewEventActivity.this, AlarmReceiver.class);
Bundle b = new Bundle();
b.putString("eventTitle", eventTitle.getText().toString());
b.putString("notificationTone", uri);
in.putExtras(b);
PendingIntent appIntent = PendingIntent.getBroadcast(NewEventActivity.this, 0, in, 0);
AlarmManager am = (AlarmManager)getSystemService(ALARM_SERVICE);
am.set(AlarmManager.RTC, finalDate, appIntent);
```

Figura 21: Alertas

Cuando pulsamos el botón de guardar evento pasan muchas cosas y una de ellas es que se genera el broadcast a traves de AlertManager que es la clase que nos permite interactuar con todo lo relacionado con las alertas. Declaramos primero un Intent que ejecutará nuestro broadcast y al cual le enviamos los datos necesarios para que en la notificación muestre los datos del evento. Seguidamente declaramos un PendingIntent. Un PendingIntent no es más que un Intent normal pero que sigue esperando a ejecutarse cuando la aplicación que lo contiene se ha cerrado. Finalmente gracias al método set de AlarmManager podemos definir que tipo de alarma vamos a usar (En este caso una alarma despertador, que suena solo una vez), la hora a la que la alarma debe sonar, y el PendingIntent que se va a ejecutar (en este caso se va a ejecutar el Intent que declaramos al principio que es el que hace que se ejecute a su vez nuestro broadcast).

5.3.5. Tonos de alerta

Para la obtener los tonos de alerta hemos usado una clase propia RingtoneCollection en la que el método getAllRingtones() nos devolvera un Map<Integer,Ringtone> (pares clave valor) y serán los que el usuario podrá elegir para notificar sus eventos. Declaramos un objeto RingtoneManager al cual le diremos que solo queremos los tonos del teléfono del tipo alarma. Con un Cursor (una especie de puntero) obtenemos todas las URI's (rutas donde se encuentran los tonos) de los tonos de alerta.

Para añadir las URI's en nuestro mapa necesitamos obtener la posición que ocupan y una vez obtenida esta posición, nos creamos un objeto Ringtone cuyo método getRingtone nos devuelve un tono de alerta.

```java
public class RingtoneCollection{

    private Activity activity;
    private RingtoneManager rM;
    public Cursor alarmCursor;

    public RingtoneCollection(Activity activity, RingtoneManager rM){
        this.activity = activity;
        this.rM = rM;
    }

    public Map<Integer,Ringtone> getAllRingtones(){
        Map<Integer,Ringtone> m = new HashMap<Integer, Ringtone>();
        rM.setType(RingtoneManager.TYPE_ALARM);
        alarmCursor = rM.getCursor();
        int alarmsCount = alarmCursor.getCount();
        if(alarmsCount == 0 && !alarmCursor.moveToFirst()){
            return null;
        }

        while(!alarmCursor.isAfterLast() && alarmCursor.moveToNext()){
            int currentPosition = alarmCursor.getPosition();
            Ringtone r = rM.getRingtone(currentPosition);
            m.put(currentPosition, r);
        }
        //TODO cerrar cursor
        //alarmCursor.close();
        return m;
    }
}
```

Figura 22: Tonos de alerta

6. ANÁLISIS TEMPORAL Y COSTES DE DESARROLLO

6.1. Material empleado

Debido a la gran diversidad de dispositivos Android que existen en la actualidad, hemos querido testear la aplicación GeoAgenda en el mayor número de dispositivos posibles con diferentes resoluciones.

El simulador de Android, aunque útil, a veces puede ser un escollo importante dado que no esta todo lo optimizado que sería deseable. Consume gran cantidad de memoria RAM por lo que hemos necesitado de un equipo informático con una cantidad de este tipo de memoria importante.

Dado que esta aplicación es software libre, hemos venido desarrollándola también con software libre. Seguidamente realizaremos una breve descripción de todos los materiales, tanto a nivel de recursos técnicos como humanos, que han sido necesarios para llevar a cabo el desarrollo de GeoAgenda.

6.1.1. Hardware

A nivel de hardware, tal y como comentamos anteriormente, hemos necesitado de un equipo rico en memoria RAM para poder utilizar el emulador de una forma eficaz. Para ello hemos usado el siguiente equipo:

- MacBook 2007
- Procesador: Intel Core 2 Duo 2.2GHz
- Memoria: 4GB
- Sistema Operativo: OSX Lion (64 bits)

Con este equipo hemos desarrollado la totalidad del proyecto. Haciendo pruebas con un emulador que, pese a algunas carencias, funcionaba con la capacidad suficiente para hacer correr la aplicación sin ningún tipo de problemas.

A pesar de que hemos podido usar el emulador en todo momento, hay diversas pruebas con las que nos ha sido imposible usarlo, por ejemplo para hacer las pruebas de localización. Aunque se puede activar y emular una localización con el emulador de Android, acabamos por descartarlo debido a que este proceso resultaba algo engorroso y para nada intuitivo. Para ello hemos probado la correcta funcionalidad de la aplicación en los siguientes dispositivos Android:

Figura 23: Huawei Ascend G300

- Modelo: HUAWEI Ascend G300
- Sistema Operativo: Android 4.0.3 Ice Cream Sandwich
- Dimensiones: 122.5 x 63 x 10.5 mm
- Peso : 140 gr.
- Tamaño pantalla: 4.0 pulgadas
- Resolución pantalla: 480 x 800
- Acelerómetro: Sí
- MP3: Sí
- GPS: Sí
- WiFi
- Etc

Figura 24: SONY XPERIA Miro

- Modelo: Sony XPERIA Miro
- Sistema Operativo: Android 4.0.3 Ice Cream Sandwich
- Dimensiones: 113 x 59.4 x 9.9 mm
- Peso : 110 gr.
- Tamaño pantalla: 3.5 pulgadas
- Resolución pantalla: 320 x 480
- Acelerómetro: Sí
- MP3: Sí
- GPS: Sí
- WiFi
- Etc

Figura 25: SONY Ericsson XPERIA ARC S

- Modelo: Sony Ericsson XPERIA ARC S
- Sistema Operativo: Android 4.0.3 Ice Cream Sandwich
- Dimensiones: 125 x 63 x 8.7 mm
- Peso : 117 gr.
- Tamaño pantalla: 4.2 pulgadas
- Resolución pantalla: 480 x 854
- Acelerómetro: Sí
- MP3: Sí
- GPS: Sí
- WiFi
- Etc

6.1.2. Software

En esta sección daremos una pequeña descripción sobre las herramientas software utilizadas para la realización del proyecto. Todas ellas son herramientas libres y están a disposición de cualquiera que desee utilizarlas.

Eclipse

Es un entorno integrado de desarrollo integrado compuesto por un conjunto de herramientas de programación de código abierto multiplataforma. Desarrollado inicialmente por IBM como sucesor de su familia de herramientas para VisualAge. Eclipse es ahora desarrollado por la Fundación Eclipse, una organización independiente sin ánimo de lucroque fomenta la comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios. Con Eclipse a parte de poder programar en Java también podremos hacerlo en C/C++, PHP, Python, Ruby, Cobol, etc.

Además es el elegido por Google para que toda la comunidad de desarrolladores de Android podamos crear nuestras aplicaciones hasta la presentación de Android Studio en la Google I/O de 2013 (Ver apéndice 2).

GitHub

Es un servicio web para el alojamiento de proyectos de desarrollo de software que usa el sistema de control de versiones Git. GitHub es una herramienta libre en la que podemos alojar proyectos públicos (visibles para todo el mundo) o privados (visibles solo para el desarrollador) previo pago de una cuota. GitHub también es una red social en la que los desarrolladores de software pueden compartir opiniones y poder seguirle la pista a tus desarrolladores favoritos. A continuación podemos ver una imagen demostrativa de lo que es GitHub.

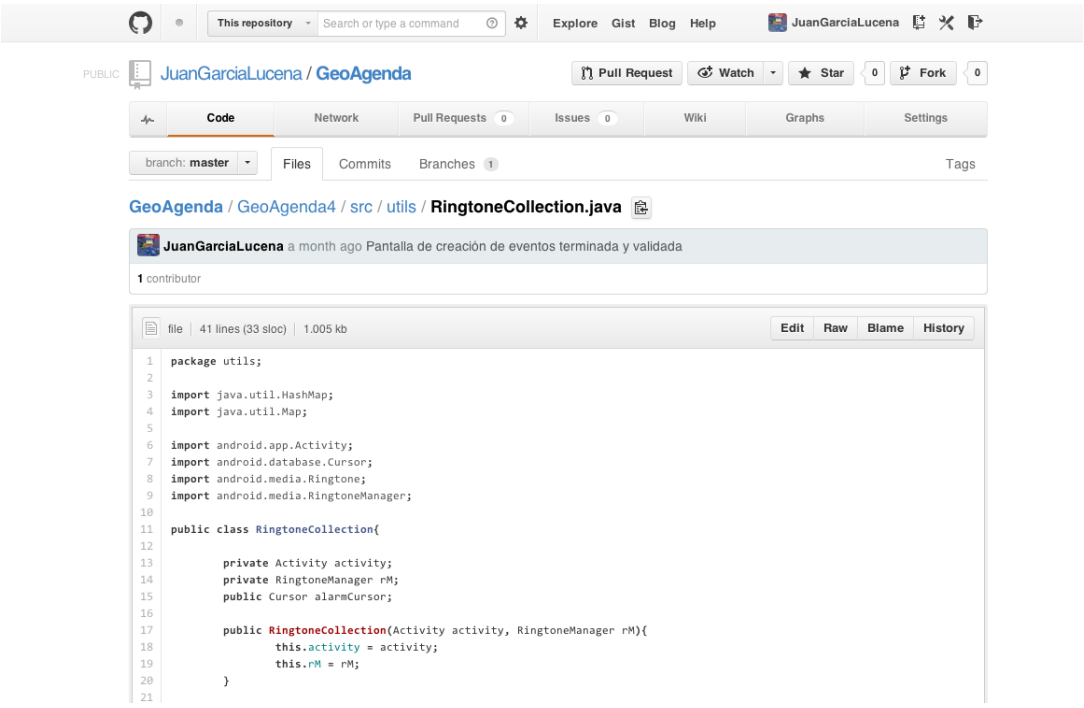

Figura 26: Vista de código en Github

Forja RedIris

En la forja de RedIRIS podemos encontrar numeroso material gracias al repositorio de proyectos de software libre que estan alojados en el. También sirve como controlador de versiones para poder sincronizar el proyecto desde equipos diferentes.

Gracias a esta plataforma hemos podido salvar algunos escollos que se interponian en el objetivo, tanto para el desarrollo de la aplicación como para la elaboración de la memoria dado que contiene un gran número de proyectos de fin de carrera de otros compañeros ya acabados y han servido como guía cuando más perdidos estábamos.

Fake GPS free

Dado que en mi habitual lugar de pruebas el GPS no me funcionaba del todo bien y para poder realizar pruebas de una forma más cómoda, decidimos descargarnos alguna aplicación que pudiera falsear la posición del GPS.

6.2. Tiempo dedicado

En esta sección expondremos el número de horas que le hemos dedicado al proyecto. Vamos a desgranar este tiempo en varios conceptos para que quede todo lo más detallado posible.

Investigación: En investigación cubrimos los siguientes puntos:

1. Aprender a utilizar el SDK y el emulador de Android
2. Aprender a programar para dicha plataforma
3. Aprender de otras aplicaciones parecidas
4. Desarrollo de los requisitos funcionales y no funcionales que tendrá la aplicación

Implementación: En el proceso de implementación incluiremos todas aquellas horas que fueron invertidas en la creación de la aplicación. En este proceso se pueden distinguir varios subprocesos como la construcción de la estructura de clases, el desarrollo de la interfaz de usuario y resolución de problemas de última hora.

Tutorias: Horas que hemos estado en el despacho de nuestro tutor.

Documentación: Para la realización de esta documentación incluiremos las horas que hemos usado para recopilar todos aquellos elementos bibliograficos que hemos usado para el desarrollo de la aplicación y la redacción de esta.

Pruebas: Tiempo que hemos usado para comprobar que la aplicación funciona correctamente.

Concepto	Tiempo (horas)
Investigación	110
Implementación	210
Tutorías	6
Documentación	105
Pruebas	30
Total	461

6.3. Costes de desarrollo

En esta sección realizaremos un cálculo aproximado del coste que tendría desarrollar GeoAgenda. Para ello sumaremos los costes del material empleado y las horas que se han invertido para el desarrollo.

Hardware

- Teléfono Sony Ericsson XPERIA ARC S: 200€
- Ordenador portátil MacBook: 1200€

TOTAL HARDWARE: 1400€

Software

- Eclipse: 0€
- Android: 0€

TOTAL SOFTWARE: 0€

Horas invertidas por parte del programador

Suponemos que el sueldo medio de un desarrollador son 1100€ al mes. 1100€ al mes son Por otro lado tenemos que 461 horas son, aproximadamente 19'20 días. Por lo que obtenemos un montante final de:

$$\frac{1100}{30} = \frac{X}{19'20}$$

Donde X = 704€

Por lo que finalmente podemos obtener el coste total de la aplicación que serían:

$$1400 + 704 = 2104€$$

7. PRUEBAS

Las pruebas que le hemos realizado a GeoAgenda han sido llevadas a cabo en diferentes localizaciones y con diferentes dispositivos Android dado que hacía falta probar con qué exactitud nos geolocalizaba la aplicación.

A continuación llevaremos a cabo una serie de pruebas para verificar que se cumplen los siguientes requisitos:

- Geolocalización aceptable.
- Correcta funcionalidad del recordatorio de evento.
- Correcta funcionalidad de la notificación "llegas tarde".
- Validación del formulario de creación de un nuevo evento.

7.1. Geolocalización

Esta prueba va a ser dividida en dos, ya que tenemos dos vias posibles de geolocalización: Por GPS y mediante redes inalámbricas. Para ambas pruebas introduciremos como localización inicial mi domicilio en la Barriada San Diego y como destino la avenida Reina Mercedes en Sevilla.

Figura 27: Crear evento

7.1.1. Geolocalización mediante GPS

Como podemos ver en la imagen la localización mediante GPS es bastante efectiva, ya que la posición actual nos la marca a escasos 10 metros de la posición actual real.

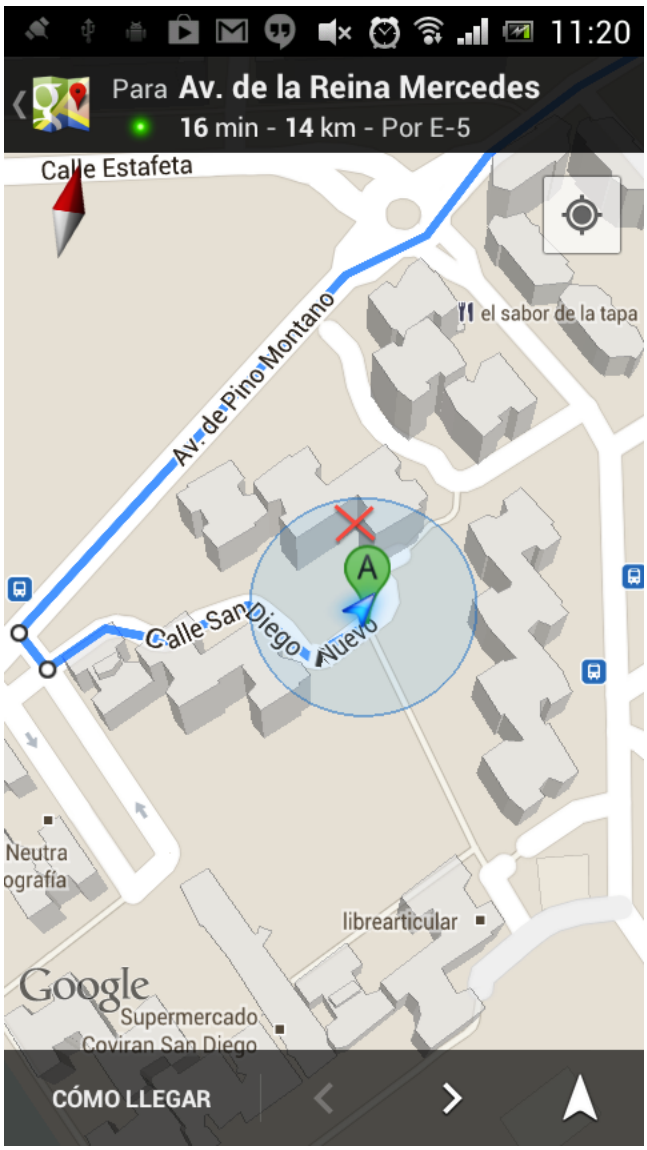

Figura 28: Geolocalización con GPS

7.1.2. Localización mediante redes inalámbricas

No podemos decir lo mismo, como era de imaginar, de la localización mediante redes inalámbricas dado que la posición actual nos la marca a unos 50 metros de nuestra posición actual real. Aún así no estaríamos a más de 5 minutos andando del lugar de partida.

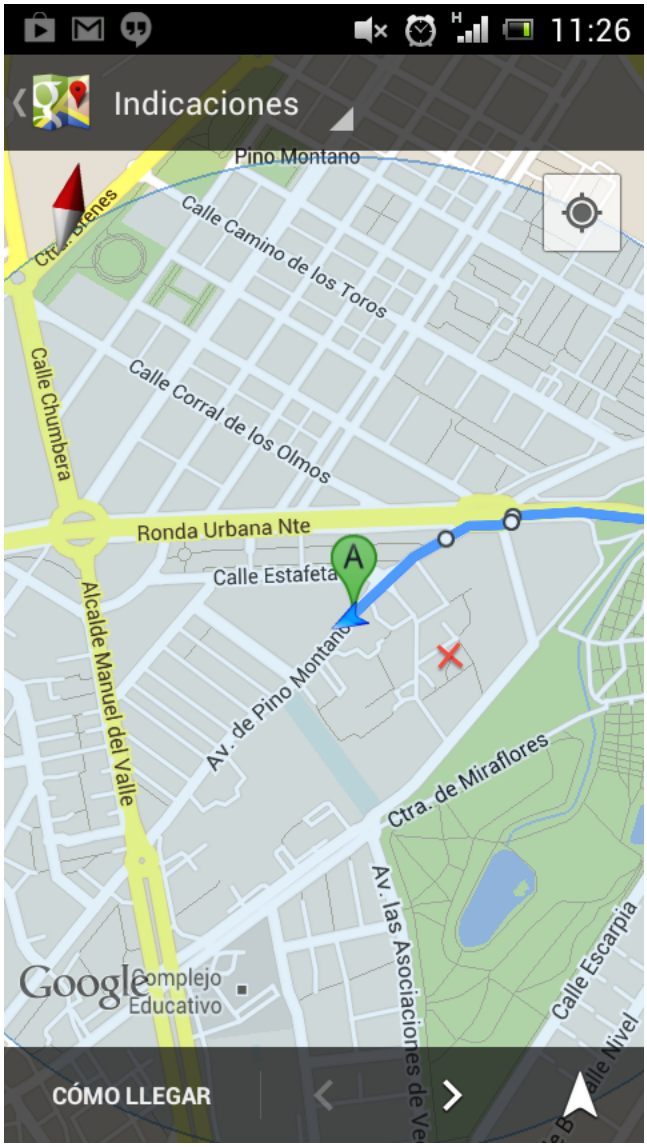

Figura 29: Geolocalización con redes

7.2. Recordatorio de evento

Cuando la hora del recordatorio ha llegado, GeoAgenda lo notifica al usuario mediante un todo de alerta que previamente fue seleccionado por este a la hora de crear el evento y, tambien, mediante una notificación en la barra de notificaciones del dispositivo. Cuando hacemos click en esta notificación, seremos llevados a la pantalla donde esta el listado de todos nuestros eventos guardados.

Figura 30: Recordar eventos

Como podemos observar se ha establecido un evento que se celebrará en la avenida Reina Mercedes en Sevilla a las 16:15 y le hemos pedido a la aplicación que nos avise a las 11:56, por lo que a dicha hora debe saltarnos una alerta en el teléfono en el que se nos indique que hay un evento próximo. Tal y como podemos ver en la siguiente imagen:

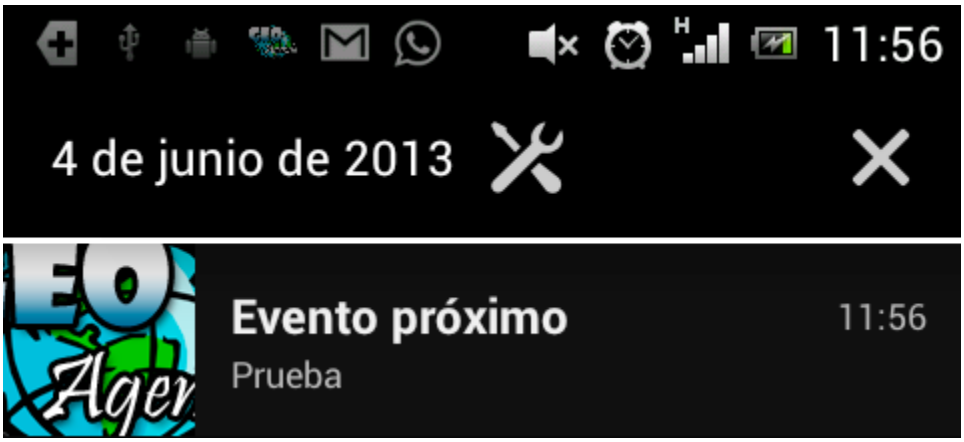

Figura 31: Notificación de evento

Hemos sido alertados, a la hora deseada, que hay un evento próximo a celebrarse.

7.3. ¡Llegas tarde!

Para llevar a cabo este supuesto vamos a poner un ejemplo bastante exagerado. Supongamos que estamos en Sevilla (posición actual) y dentro de tres horas tenemos una reunion en Madrid. Obviamente no vamos a llegar a tiempo, pero es un buen supuesto para que la alerta de que llegamos tarde sea lanzada.

Figura 32: Prueba de tardanza

Para generar este aviso, una hora antes del recordatorio, calculamos cuanto tiempo tardaríamos en llegar a nuestro destino. Si este tiempo es mayor del que tenemos entre el recordatorio hasta la fecha del evento, hacemos saltar la alerta de que llegamos tarde.

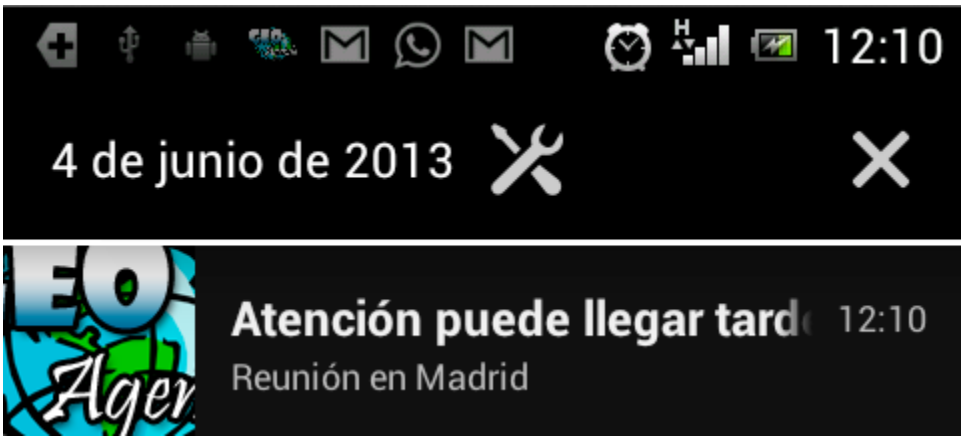

Figura 33: Notificación de tardanza

7.4. Validación del formulario

Para poder llevar a cabo el objetivo de GeoAgenda, no se puede dejar nada al azar y mucho menos a la hora de rellenar el formulario. Para ello se han dispuesto varias alertas cuando este no haya sido debidamente rellenado. Casos como establecer un evento o un recordatorio antes de la fecha actual, que la hora del recordatorio sea posterior a la del evento o que no se hayan completado los campos de texto son algunos de los casos.

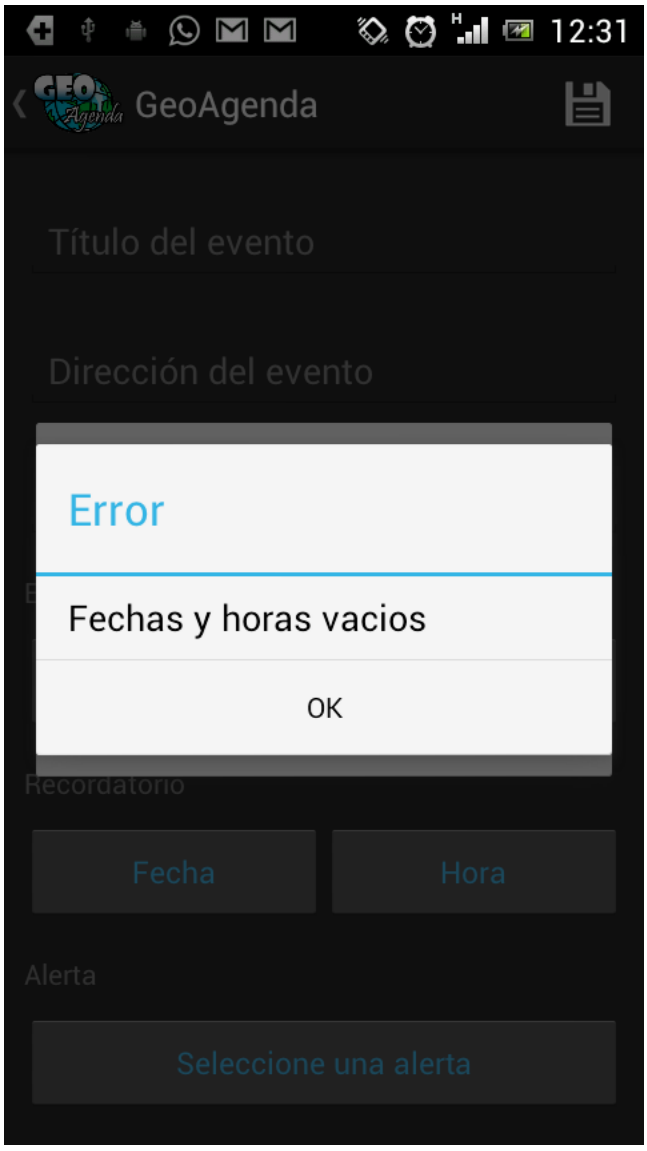

Figura 34: Validación 1

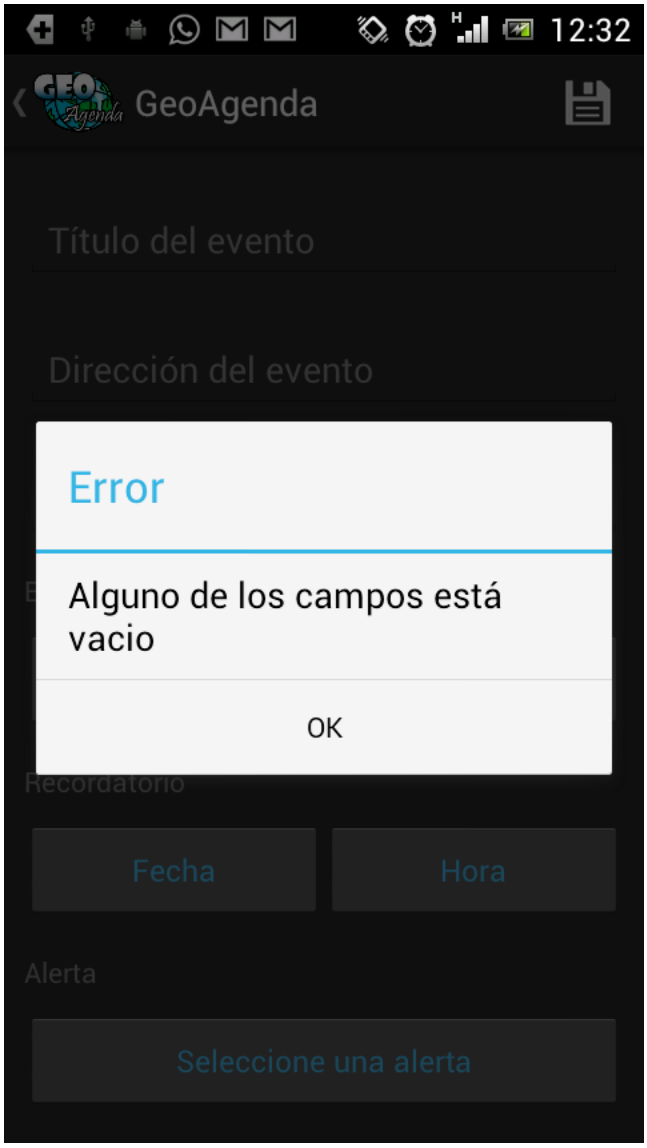

Figura 35: Validación 2

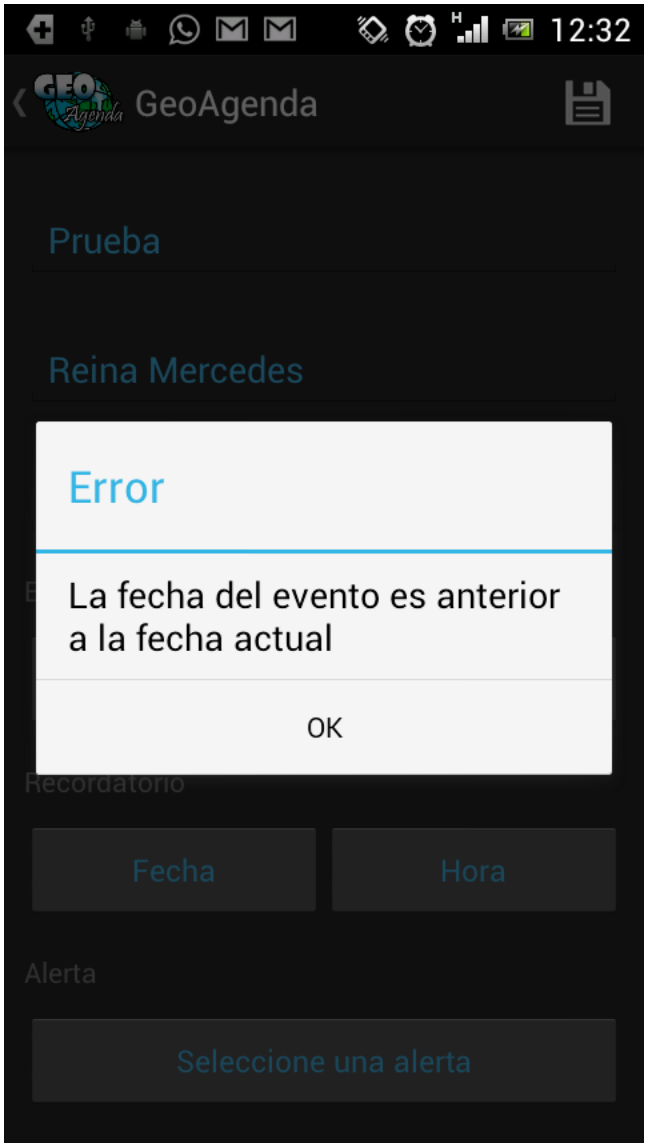

Figura 36: Validación 3

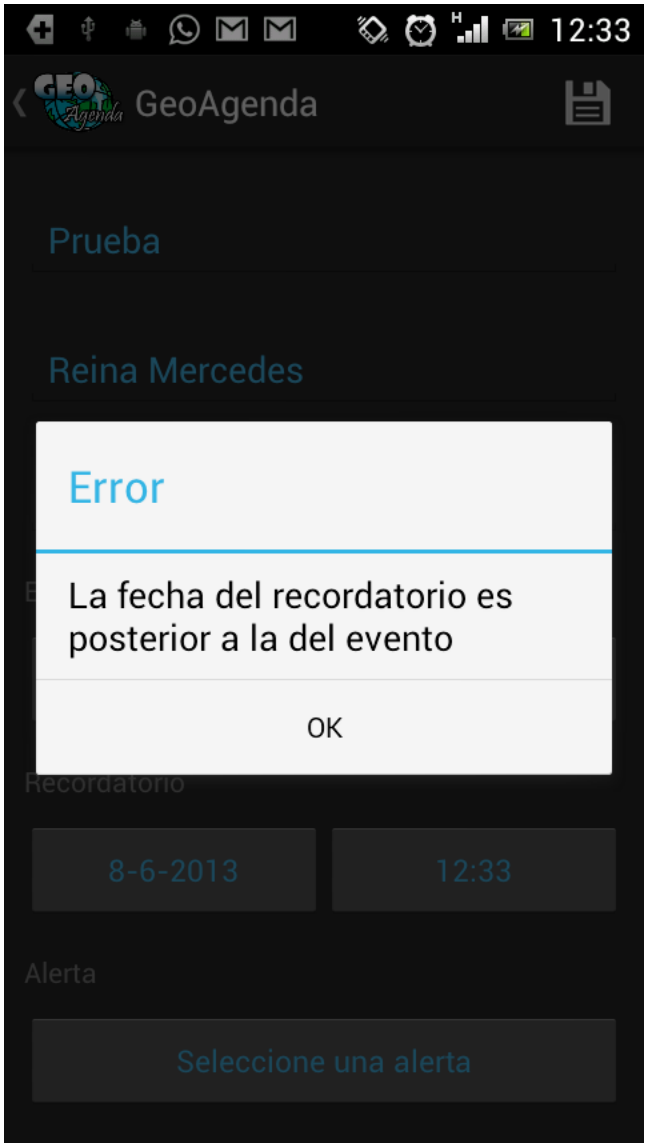

Figura 37: Validación 4

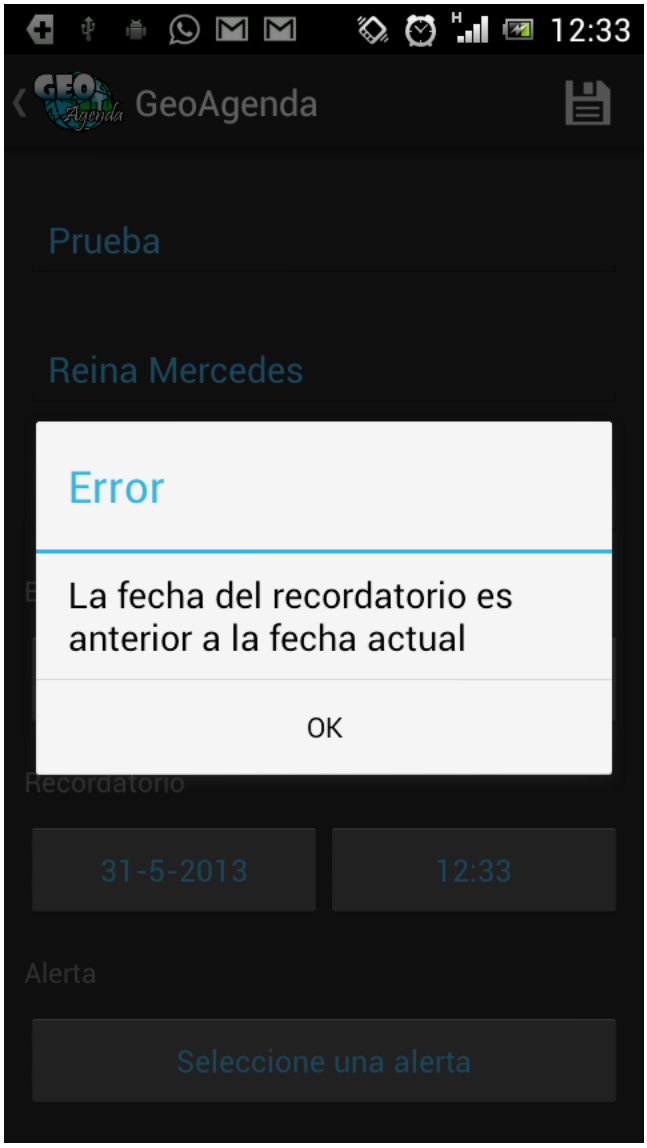

Figura 38: Validación 5

Cuando todo ha ido bien, la aplicación nos dirige a la pantalla principal mientras una notificación nos indica que se ha guardado el evento con éxito.

Figura 40: Evento guardado

8. MANUAL DE USUARIO

En este capítulo vamos a explicar como funciona GeoAgenda. Tal y como se explicó en los primeros capitulos, se ha querido hacer una aplicación sencilla de usar para que, sin dar muchas vueltas, podamos conocer como funciona la aplicación.

Figura 41: Menú principal

Este es nuestro panel de inicio. En él podemos encontrar 4 iconos:

1. Eventos: El icono de eventos nos lleva a una pantalla en la que se muestran todos los eventos que ya hayan sido guardados.

2. Nuevo evento: Accediendo a esta pantalla procederemos a la pantalla de creación de nuevo evento.

3. Ajustes: Dado que GeoAgenda es un prototipo esta opción aún no esta desarrollada. En futuros desarrollos al hacer click en este icono accederemos a una pantalla en la que podriamos encender o apagar los servicios de localización (GPS o redes inalámbricas), ajustar un tono de alerta por defecto, etc.

4. Información: Al hacer click accederemos a toda la información sobre el proyecto.

8.1. Nuevo evento

Esta pantalla esta destinada a la introducción de todos los datos necesarios para generar un evento. Para ello, obligatoriamente, deberemos completar todo el formulario que consiste en, nombre del evento, dirección del evento, ciudad del evento, hora del evento, hora de recordatorio y tono de alerta.

Una vez completado todo el formulario pulsaremos el botón de guardar, situado en la esquina superior izquierda (el típico icono del disquette). Si todo ha ido bien la aplicación nos llevará de vuelta al menú inicial, indicándonos que el evento ha sido guardado correctamente en la base de datos.

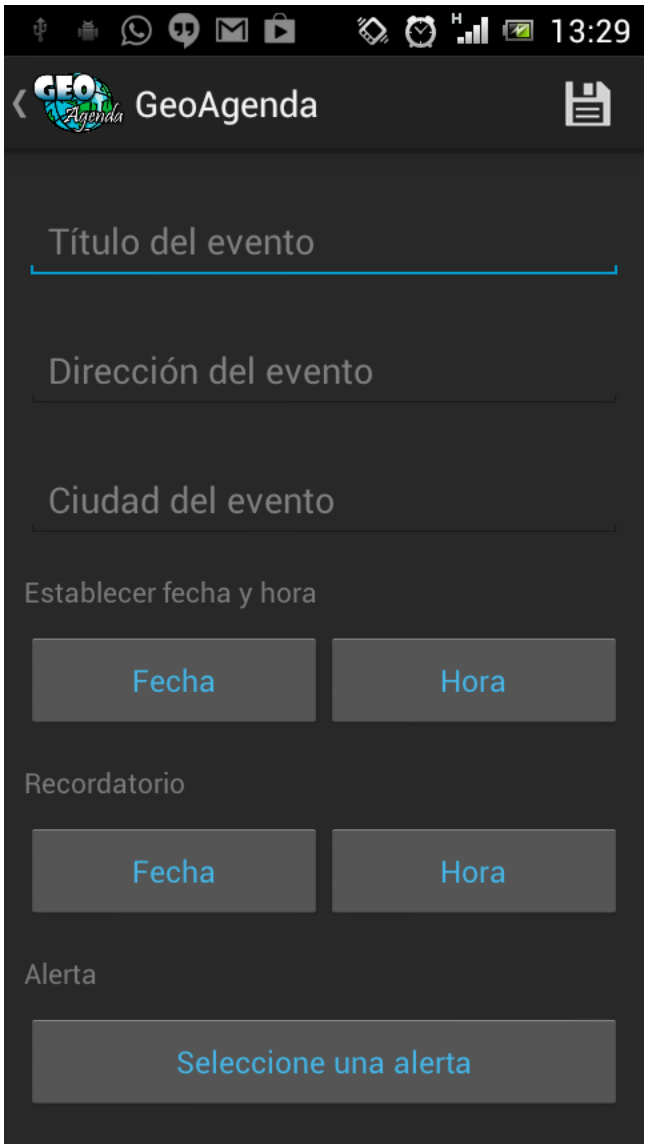

Figura 42: Crear evento

8.2. Eventos

Esta pantalla es una simple lista de todos los eventos que hemos ido guardando. Al hacer click en alguno de los elementos de la lista, podremos ver nuevamente los detalles del evento.

Figura 43: Lista de eventos

8.3. Detalle de eventos

Aquí a parte de poder ver nuevamente los detalles del evento seleccionado en la lista, podemos borrarlos y también podemos acceder al mapa en el que se nos muestra la ruta a seguir (por defecto en coche) para poder llegar al lugar de celebración del evento a partir de la posición actual

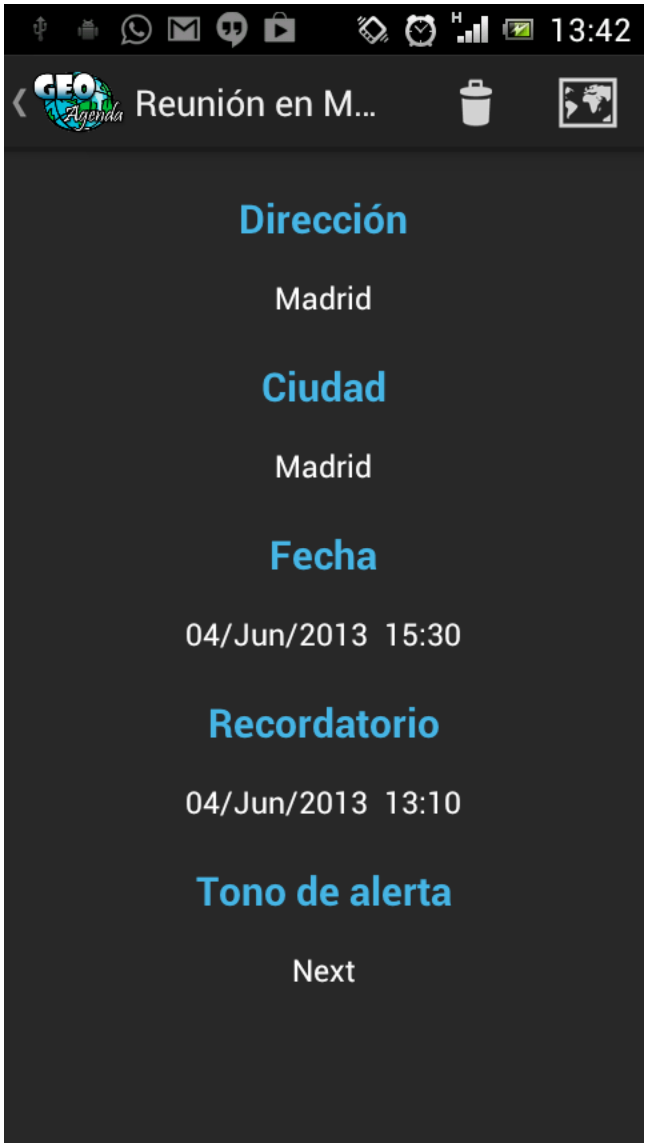

Figura 44: Detalle de evento

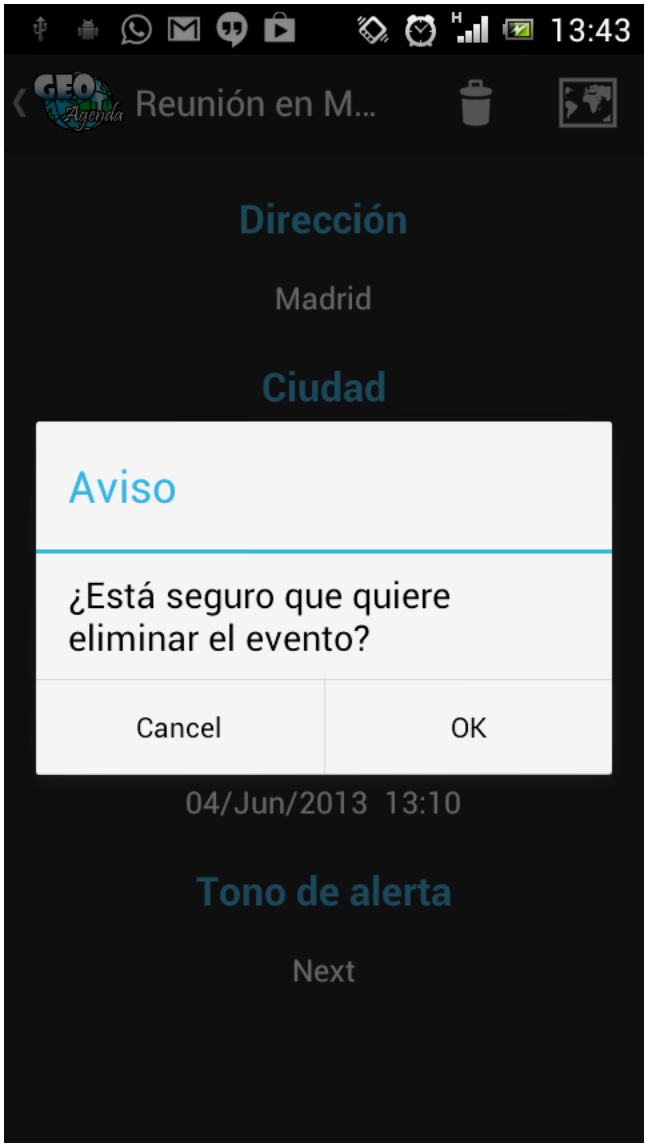

Figura 45: Eliminar evento

Figura 46: Ruta

8.4. Información

En información podemos encontrar quién ha realizado el proyecto y otro tipo de información tal y como se puede observar en la imagen

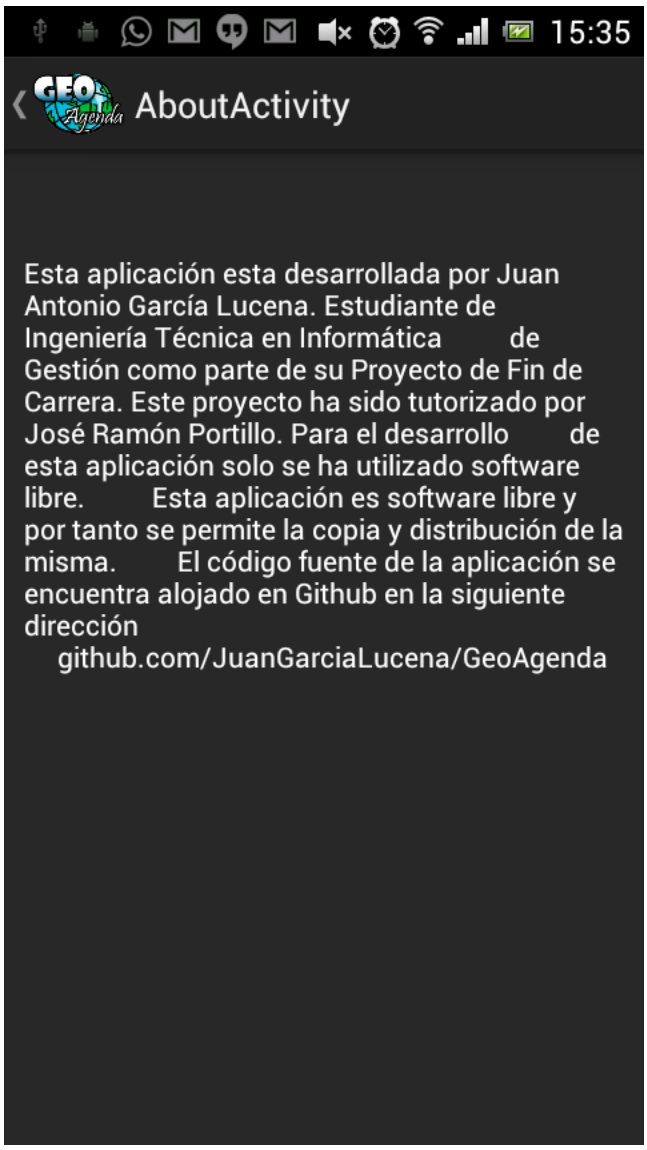

Figura 47: Acerca de

8.5. Alertas

Cuando la hora y fecha de un recordatorio ha llegado, GeoAgeda notificará al usuario mediante el tono de alerta que este seleccionó a la hora de crear el evento y también a través del sistema de notificaciones de Android.

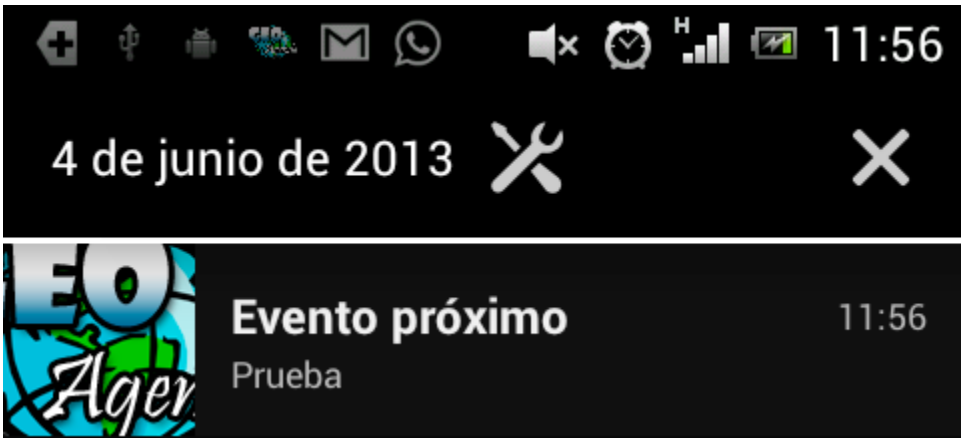

Figura 48: Alertas 1

Una hora antes del recordatorio GeoAgenda calcula si es posible llegar al lugar del evento a partir de la hora del recordatorio y el lugar en el que el usuario se encuentra en dicho momento. Si no es así GeoAgeda le avisará de que es posible que pueda llegar tarde a dicho evento usando los mismos mecanismos de notificación que se han descrito anteriormente.

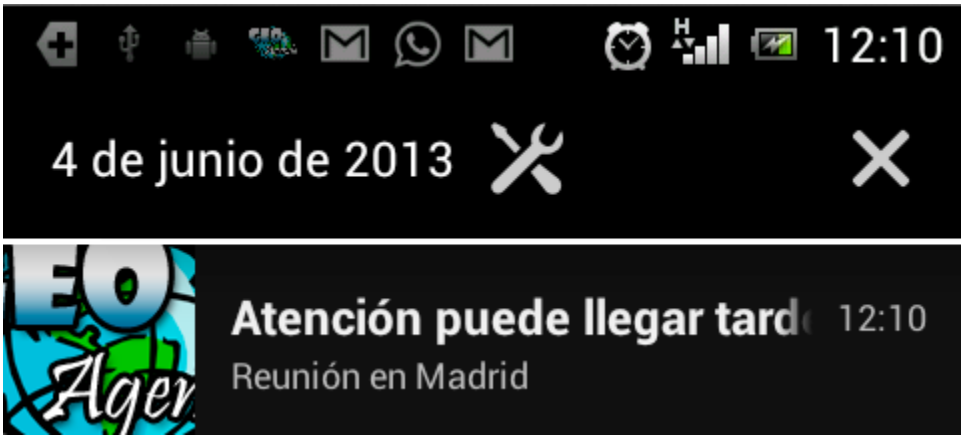

Figura 49: Alertas 2

9. COMPARACIÓN CON OTRAS ALTERNATIVAS

A continuación haremos una breve descripción de las alternativas a GeoAgenda que actualmente podemos encontrar para nuestros terminales Android.

Dado que GeoAgenda es un proyecto de Software Libre, la selección de alternativas que hemos realizado ha sido partiendo desde la premisa que la aplicación debe ser gratuita.

9.1. Google Calendar

Poco hay que decir de esta aplicación desarrollada por Google. Podemos crear, editar y eliminar eventos. Podemos ver todos los calendarios que tengamos a la vez incluso si no son de Google. También es posible enviar un correo electrónico a todas las personas que hayamos invitado a los eventos que hayamos creado. También nos muestra la localización del evento y una ruta desde nuestra posición actual.

Figura 50: Pantalla principal

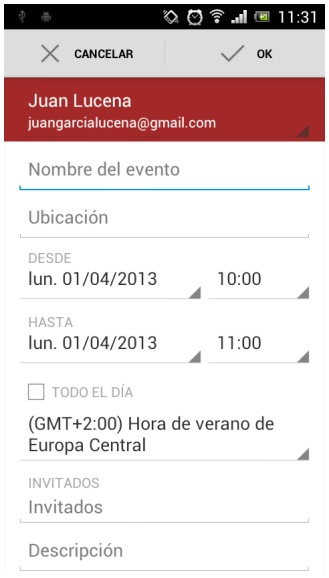

Figura 51: Creación de un evento

9.2. DigiCal

DigiCal es una aplicación que, tal y como Google Calendar, sincroniza los eventos con los calendarios de las cuentas de Google asociadas en el teléfono. Ofrece la posibilidad de cambiar la interfaz. La versión testeada es una version gratuita con publicidad, existiendo otra llamada DigiCal+ la cual no posee dicha publicidad.

Figura 52: Vista del calendario

Figura 53: Creación de un evento

9.3. Jorte Calendar

Jorte Calendar es otra aplicación típica de calendario cuyos eventos pueden ser sincronizados con nuestros calendarios de Google Calendar. Ofrece una interfaz similar, permitiéndonos eliminar o editar los eventos creados.

Aunque actúa como un calendario bastante completo, no ofrece como las dos anteriores, servicio de geolocalización.

Figura 54: Pantalla principal

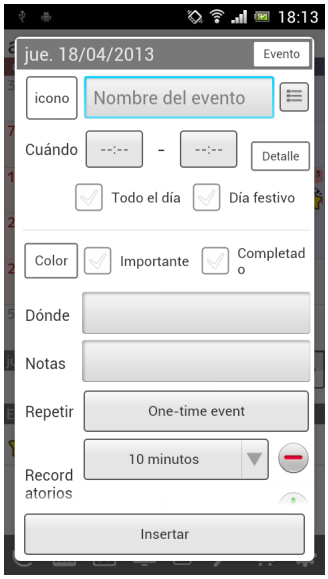

Figura 55: Creación de evento

10. CONCLUSIONES Y DESARROLLOS FUTUROS

10.1. Conclusiones

La sensación general tras terminar el proyecto de fin de carrera es la de que he aprendido mucho sobre un sistema operativo y en concreto un mundo, el de la movilidad, que para mi y muchos otros tiene muchísimo futuro y camino que recorrer aún.

Los primeros días de desarrollo fueron bastante duros. Ni tenía decidido si desarrollar el proyecto con algun framework o de forma nativa.

A resolver esta cuestión me ayudó mucho el año que estuve trabajando como becario en Viavansi. Estuve involucrado en dos proyectos de movilidad en el que estos eran desarrollados con Titanium Studio. Los resultados fueron buenos pero no tan buenos como cabría esperar si la aplicación hubiera sido nativa. Esto me hizo decidirme sobre el desarrollo nativo de mi PFC, el hecho de que ya conocía un buen número de frameworks de desarrollo para dispositivos móviles y que siempre me han gustado los retos (trabajar con una plataforma ya conocida no ampliaría mis conocimientos).

Durante el desarrollo de GeoAgenda ha habido momentos muy buenos y, también, muy críticos, tanto es así que la versión que se entrega es la cuarta versión en la que se ha trabajado. Pero me quedo con todo lo que he aprendido y he disfrutado desarrollando este PFC. Ahora solo queda trabajar duro para mejorar mis habilidades en el desarrollo para dispositivos móviles ya que me gustaría poder dedicarme profesionalmente a ello.

10.2. Desarrollos futuros

En el desarrollo de la aplicación se han quedado muchas cosas atrás, algunas las hemos comentado y otras no. Entre las que hemos comentado con anterioridad está el ofrecer un menú para seleccionar la localización correcta en caso de que la aplicación nos devuelva más de una y no coger la primera sin más.

También nos hubiera gustado desarrollar un poco el menú de ajustes que se ha sacrificado por querer dedicar más tiempo a poder desarrollar un prototipo de aplicación lo más cercano posible al producto final.

Por otro lado también nos hubiera gustado poder investigar y desarrollar un poco más la interfaz de usuario. La interfaz Holo con la que viene provista Android en sus últimas versiones, es muy potente y digna de ser investigada con mayor detenimiento ya que por lo menos en nuestro caso es una interfaz que nos tiene realmente maravillados.

A nivel de código también nos gustaría revisarlo en un futuro porque siempre hay cosas que, cuando la programas por primera vez parecen que estan realmente bien hechas, pero con el paso del tiempo puede que esa opción que tan válida era en el pasado ya no lo sea tanto.

Finalmente, a nivel estratégico, también nos gustaría desarrollar la posibilidad de sincronizar los calendarios con Google Calendar dado que es una opción que casi todas las aplicaciones de agenda poseen y no hemos tenido la posibilidad de desarrollarlo para GeoAgenda.

11. APÉNDICES

11.1. APENDICE I - Obtener una API Key de Google Maps

Para el desarrollo de esta aplicación, que hace uso de los recursos de geolocalización que tienen todos los smartphones de un tiempo a esta parte, hemos decidido que Google Maps es el servicio de mapas que mejor se adapta. Pero para hacer uso de este tipo de servicio necesitamos una API Key que nos suministrará Google.

¿Qué es y por qué una API Key?

Una API Key es un identificador único que Google suministra a todos aquellos que usen sus servicios y está asociada a la aplicación, como podemos ver en el archivo AndroidManifest de GeoAgenda en la figura ¿?, nunca a los usuarios. Esta API Key es la forma que tiene Google de conocer desde donde se le estan haciendo peticiones a sus servicios y, dependiendo del numero de peticiones y de servicio (normalmente por día), cobrar al desarrollador por el uso de su servicio (en la figura ¿? podemos ver los usos de Google Maps).

```
<meta-data android:name="com.google.android.maps.v2.API_KEY"
    android:value="AIzaSyCTHuerAX3IWRAjm_lBiueXTeF39VikLLc"/>
<activity
```

Figura 56: AndroidManfest

Features	Maps API	Maps API for Business
Street View	✓	✓
Geocoding Web Service	2500 requests per day	100 000 requests per day
Directions Web Service	2500 requests per day with 10 waypoints per request	100 000 requests per day with 23 waypoints per request
Distance Matrix Web Service	100 elements per query 100 elements per 10 seconds 2500 elements per day	625 elements per query 1000 elements per 10 seconds 100 000 elements per day
Elevation Web Service	2500 requests per day with 25 000 samples per day	100 000 requests per day with 1 000 000 samples per day
Static Maps API maximum resolution	640 x 640	2048 x 2048
Static Maps API maximum scale	2X	4X
Street View Image API maximum resolution	640 x 640	2048 x 2048
Analytics		✓
Demographics Layer		✓

Figura 57: Terminos de uso de GoogleMaps

La obtención de API Key's de Google Maps ha cambiado recientemente con la salida de la version 2.0 de este servicio. Los pasos a seguir se detallan a continuación:

1. Instalar el paquete Google Play Services en Eclipse desde el SDK Manager

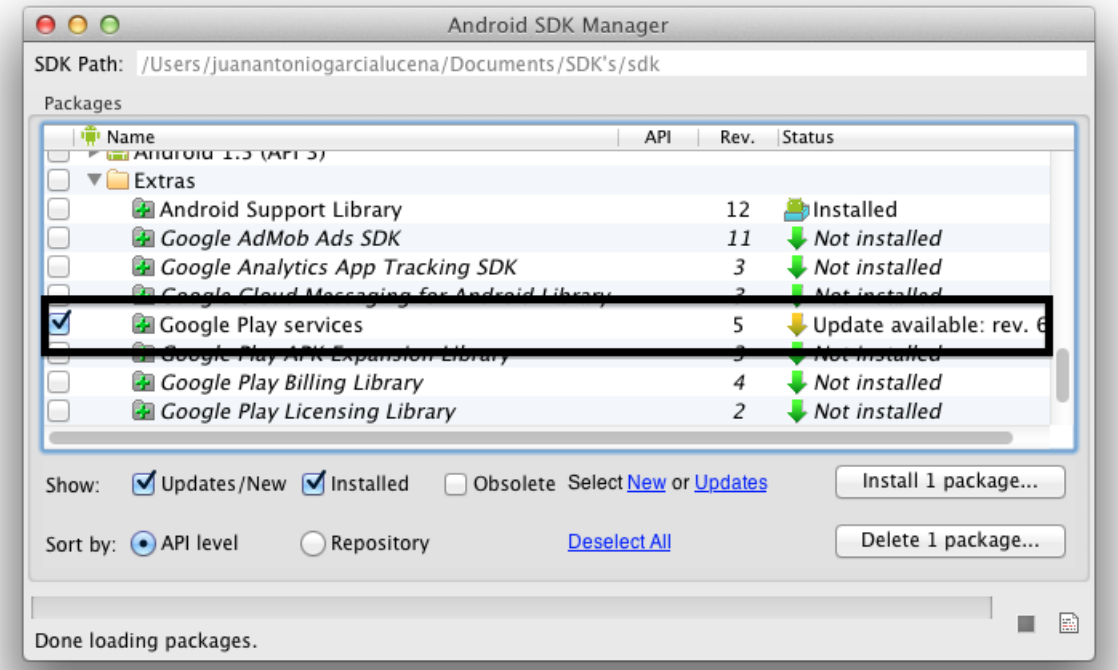

Figura 58: Android SDK Manager

2. El segundo paso a realizar es crear un proyecto en la pagina de Google Apis en la dirección https://code.google.com/apis/console/ y en el menú API Project hacer click en "create" e introducimos el nombre de nuestro proyecto

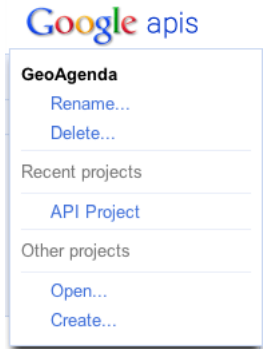

Figura 59: Google APIS

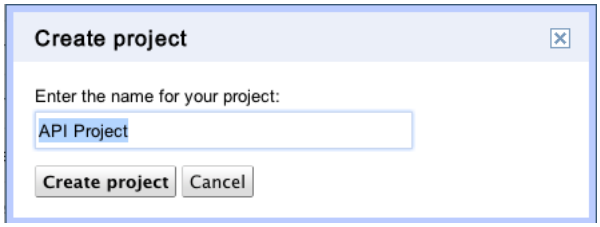

Figura 60: Create Project

3. Seguidamente nos aparecerá una lista de servicios de Google. Buscamos Google Maps Android API v2 y hacemos click en OFF para que cambie a ON.

Figura 61: GoogleMaps API

4. Una vez hecho esto, volvemos al menú lateral (donde creamos el proyecto) y seleccionamos la opcion API Services donde nos aparecerá, entre otras, la opción de crear una nueva Aandroid Key y hacemos click en esta opción (Nótese que en la figura ¿? ya hay creada una API Key. De momento obviemos este hecho).

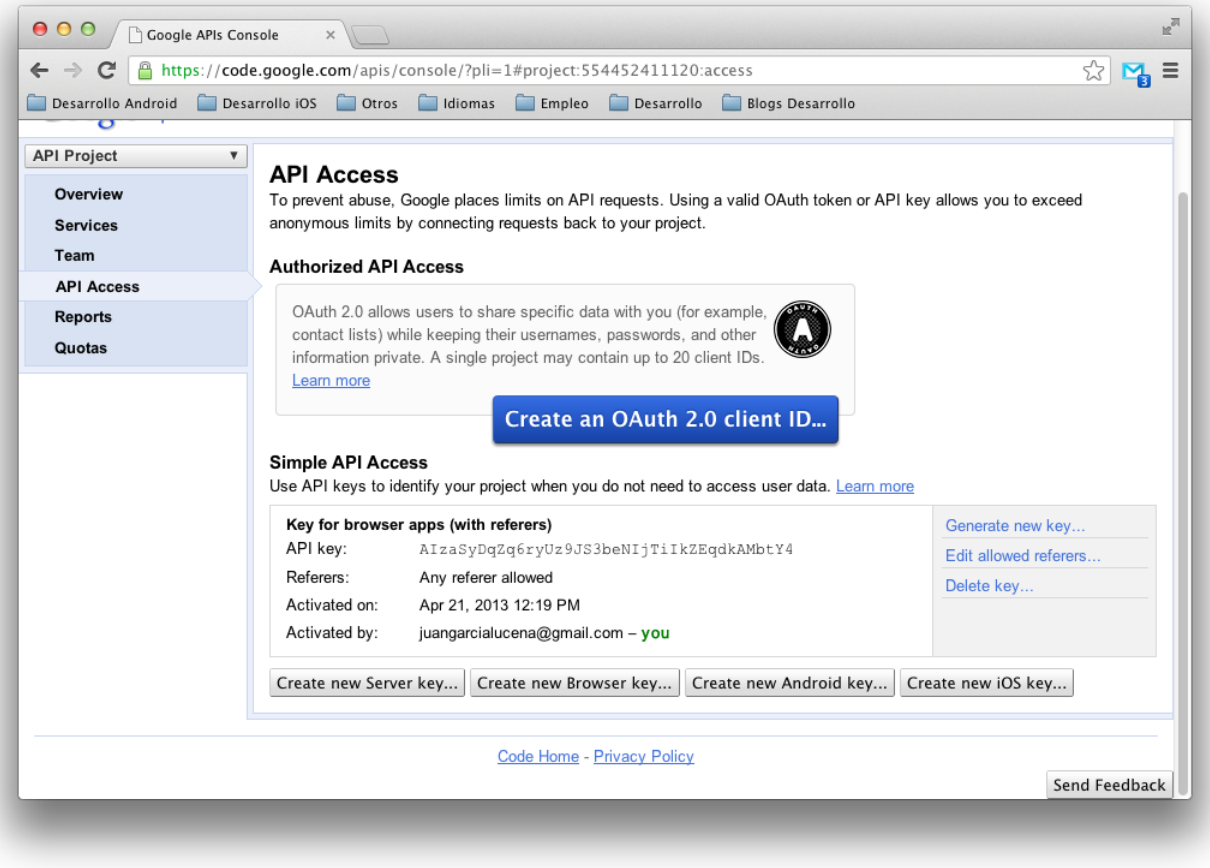

Figura 62: Obtención de API Key

5. Una vez que hemos entrado en el menú de creación de una nueva Key para nuestra aplicación, se nos informará que necesitamos una "huella digital SHA1" (Figura ¿?). Para obtenerla abrimos la consola e introducimos la orden que nos indica Google:

keytool -list -v -keystore mystore.keystore

Donde mystore.keystore es el paquete que hemos definido en la aplicación que en este caso será com.geoagenda.

Una vez introducida la orden obtendremos una clave como la que se puede observar en la figura ¿?

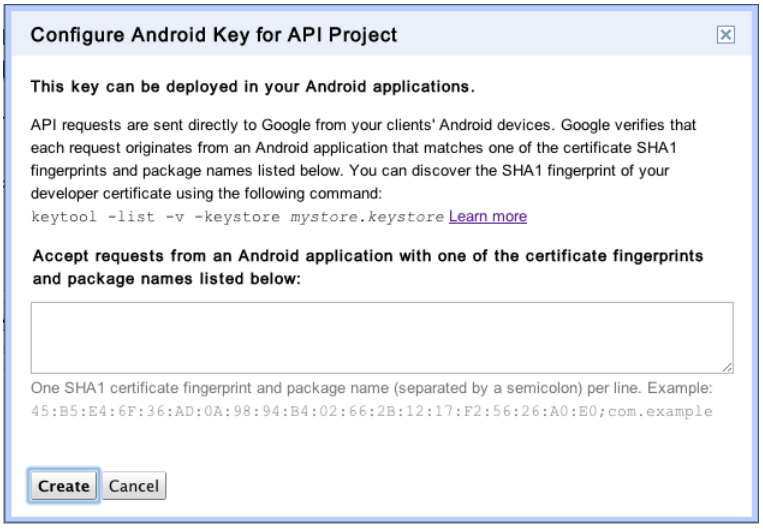

Figura 63: Configure API Key

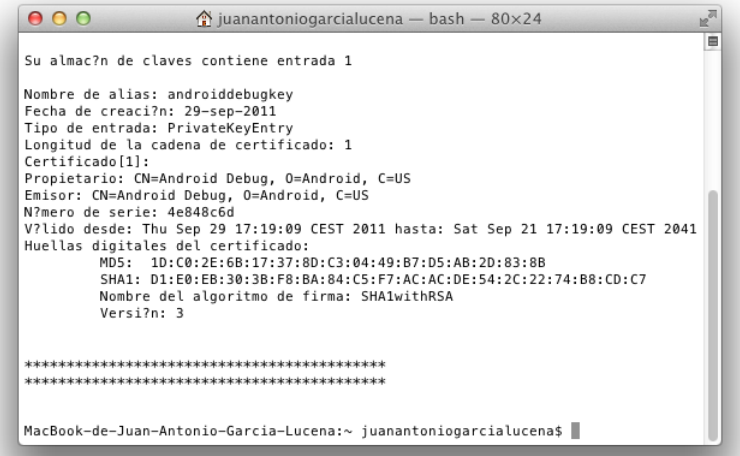

Figura 64: Fingerprint

6. Ya obtenida la huella digital, la introducimos en el menú de creación de la Key, hacemos click en "Create" y acto seguido en la pantalla principal de nuestro proyecto nos aparecerá el nombre del proyecto y su API Key asociada que usará nuestra aplicación.

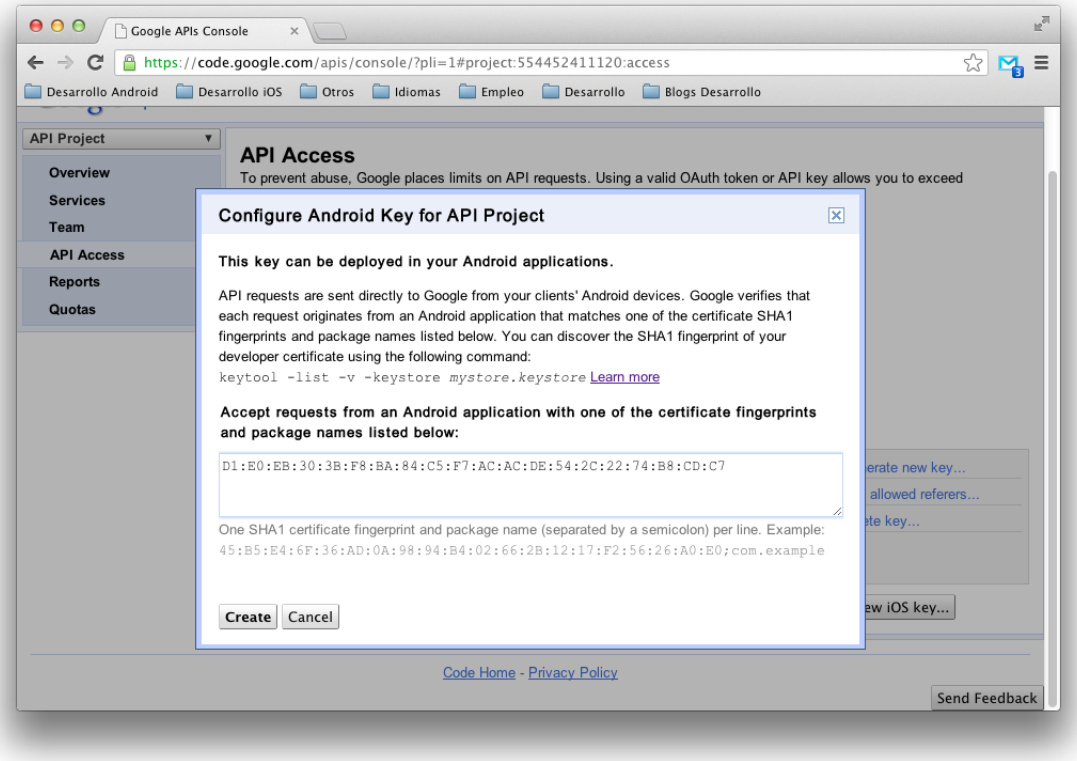

Figura 65: Introducción de API Key

Key for browser apps (with referers)

API key:	AIzaSyDqZq6ryUz9JS3beNIjTiIkZEqdkAMbtY4
Referers:	Any referer allowed
Activated on:	Apr 21, 2013 12:19 PM
Activated by:	juangarcialucena@gmail.com – **you**

Figura 66: Resumen

11.2. APENDICE II - Android Studio

Android Studio es el nuevo IDE presentado por Google en la Google I/O de 2013, en lo que supone el primer IDE oficial lanzado por Google para el desarrollo de aplicaciones para dispositivos Android. Android Studio esta basado en el IDE IntelliJ Idea desarrollado por JetBrains (cuya primera versión fue lanzada en enero de 2001) y que supone un importante rival para Eclipse para el desarrollo de aplicaciones. Con Intellij Idea podemos programar en numerosos lenguajes de programación tales como Java, PHP, Python, SQL, HTML, CSS y un largo etcétera. En este apéndice veremos como crear un "HelloWorld" para Android y así podremos comprobar que los cambios que se ofrecen respecto a Eclipse son mejorados en algunas facetas.

Para empezar cuando abrimos Android Studio, nos aparece esta ventana en la que se nos da a elegir entre importar o crear nuevos proyectos o, como se ve en la barra lateral izquierda, continuar con otros proyectos ya creados.

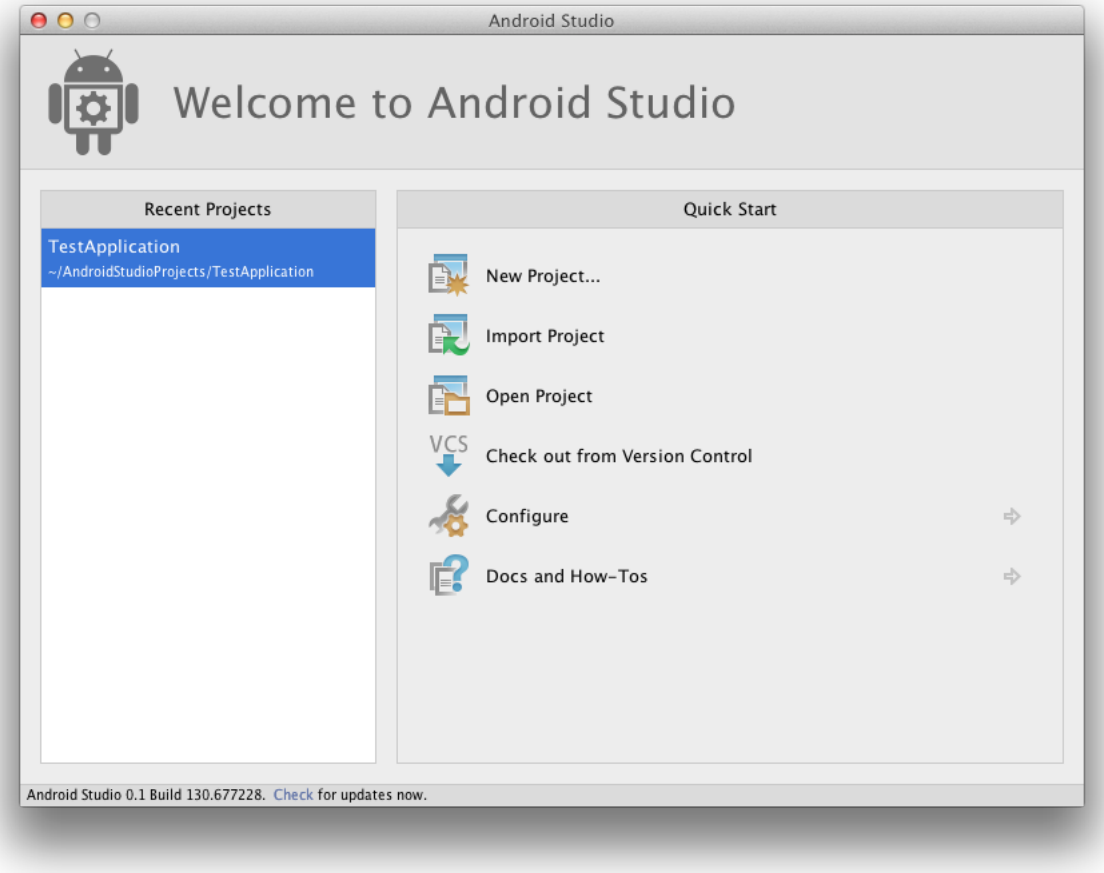

Figura 67: Android Studio

En este caso seleccionamos “New Project”. A continuación, tal y como ocurría en Eclipse se nos pide introducir cierta información para poder seguir adelante. Como El nombre del proyecto, para que versiones de Android se va a desarrollar el proyecto etc

Figura 68: Crear proyecto

Introducimos los datos y al pulsar next se nos dará la opción de configurar el icono de la aplicación y con qué tipo de activity vamos a empezar el proyecto, tal y como se pueden ver en las siguientes imagenes.

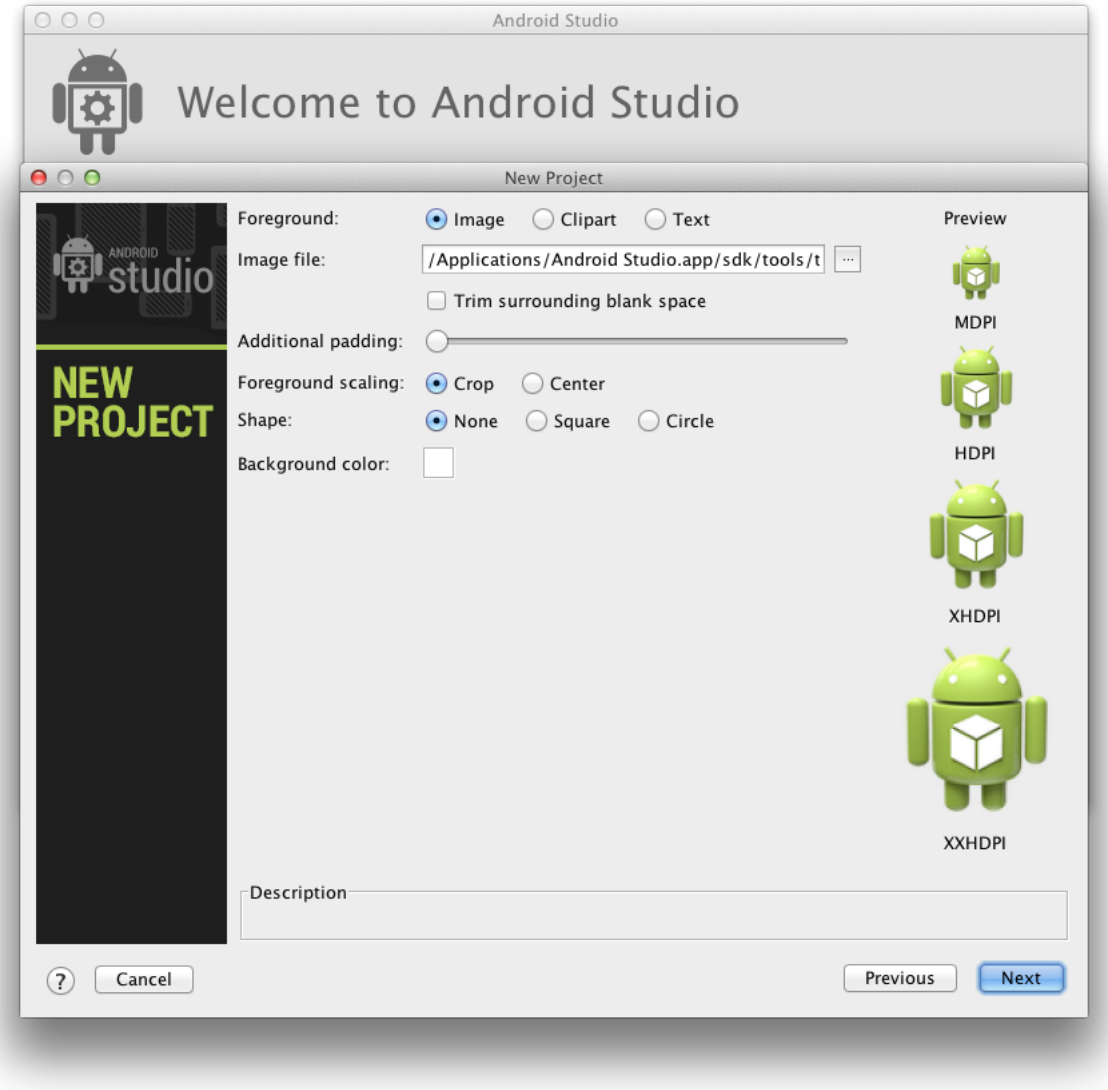

Figura 69: Configuración del proyecto

Figura 70: Configuración del proyecto 2

Una vez que hemos terminado de introducir los últimos parámetros iniciales del proyecto Android Studio procede a la creación del proyecto obteniendo la siguiente pantalla

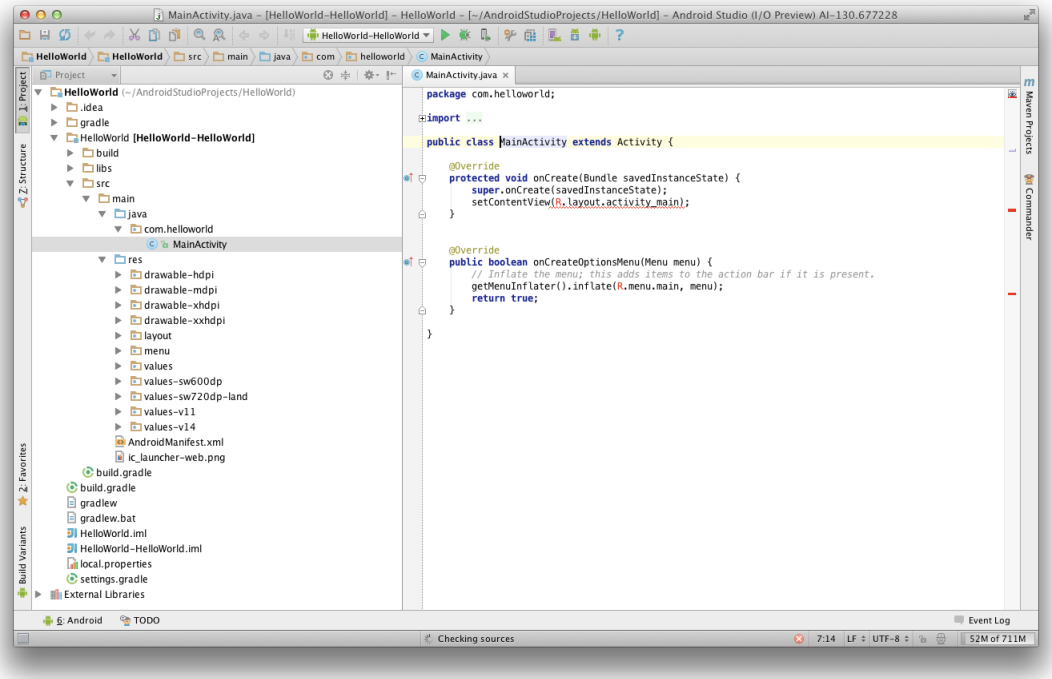

Figura 71: Vista general

A continuación viene, en nuestra opinión, una de las grandes mejoras que trae consigo Android Studio. Durante el desarrollo de GeoAgenda se nos hizo, en ocasiones, desesperante el diseñar la interfaz de usuario debido a que a medida que se iba escribiendo el código era imposible saber como iba quedando visualmente, teniendo que cambiar la vista de código a la vista gráfica para saber si los cambios realizados eran correctos o no. Esto no pasa con Android Studio dado que tal y como se puede ver en la imagen, a medida que vamos escribiendo código los cambios de la interfaz gráfica se generan en tiempo real, haciendo así más ameno el desarrollo de la interfaz de usuario. También tendremos la posibilidad de cambiar entre dispositivos nexus y otros dispositivos con todo tipo de resoluciones.

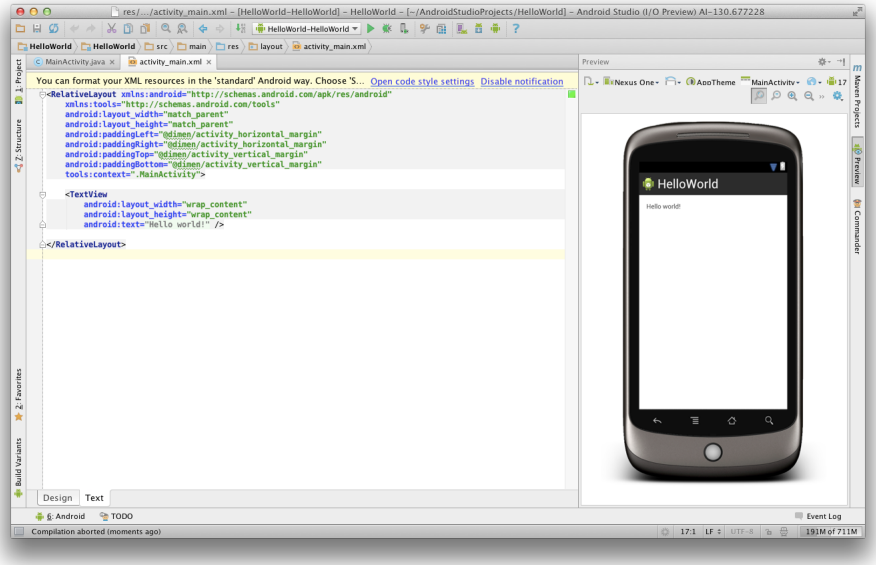

Figura 72: Vista de diseño

Como hemos podido comprobar Android Studio no revoluciona la forma de desarrollar aplicaciones para Android, es más podríamos decir que se asemeja en la mayoría de las cosas a Eclipse. Pero como también hemos podido comprobar, aunque pocas, vienen con algunas mejoras (desarrollo de interfaz de usuario) que, teniendo en cuenta que estamos ante una versión muy primitiva de este IDE, la 0.1 (aunque mientras se escriben estas lineas se ha lanzado un update a la version 0.12), nos hace disfrutar con el potencial que puede desarrollar esta herramienta en futuras versiones.

12. BIBLIOGRAFÍA

- Tomás Gironés, Jesús. El gran libro de Android. Editorial Marcombo (2011).
- Lee, Wei-Meng. Android 4. Desarrollos de aplicaciones. Ediciones Anaya Multimedia (2012).
- Ableson, W. Frank y otros. Android: Guía para desarrolladores. 2a Edición (2011).
- McCracken, Scott. Android: Curso de desarrollo de aplicaciones. Editorial INFORBOOK'S S.L. (2012).
- Martín Otero, Francisco Javier: Recursos Android basados en geolocalización. Proyecto de fin de carrera de la Escuela Técnica Superior de Ingeniería Informática de la Universidad de Sevilla. Dirigido por José Ramón Portillo Fernández, departamento de Matemática Aplicada I. Sevilla Junio 2012.
- León Padilla, Juan: Tool to announce arrival. Proyecto de fin de carrera de la Escuela Técnica Superior de Ingeniería Informática de la Universidad de Sevilla. Dirigido por José Ramón Portillo Fernández, departamento de Matemática Aplicada I
- Tutorial Android SGOliver `http://www.sgoliver.net/blog/?page_id=3011`
- Tutorial Android Ya `http://www.javaya.com.ar/androidya/`
- GitHub `https://github.com/`
- StackOverFlow `http://stackoverflow.com/`
- Android Developers `http://developer.android.com/intl/es/index.html`
- Wikipedia `http://www.wikipedia.org/`
- API GoogleMaps `http://code.google.com/intl/es-ES/android/add-ons/google-apis/reference/index.html`
- Play Store `https://play.google.com/store`