



GEOPOINT

INDICE

Introduccion	3.
Analisis	
- Estado del arte	3.
Fase de diseño	
- Lenguajes de programacion	4.
- Frameworks	6.
- Aplicación	7.
- Modelo Entidad Relacion y tablas	10.
Fase de Desarrollo	
- Historico de Desarrollo	12.
- Proceso de Desarrollo	12.
- Partes Principales de la App	16.
- Fase de pruebas	19.
- Manual del usuario	21.
- Presupuesto	24.
Fase de Mantenimiento	
- Posibles mejoras	25.
Conclusiones	26.

INTRODUCCION

Quien no ha vuelto de vacaciones o ha pasado por un sitio que ya visito y se ha acordado de ese restaurante que se comía tan bien y no se acuerda de donde estaba ni tampoco de cómo llegar, o de donde estaba esa playa tan bonita y escondida que un día encontró por error y que recuerda perfectamente pero el llegar ya no es tan fácil... Y ¿A quién no le interesa poder tener una agenda en el móvil en la que puedas tener almacenados los diferentes GeoPoint que hayas ido visitando o que quieras visitar como una ruta por la ciudad, diferentes monumentos, museos, restaurantes, parques, etc...? .

Además de disponer de la posibilidad de guardarlos también puedes compartir tus GeoPoint con tus amigos y agregar los suyos a tu agenda, así, si un amigo se marca una ruta de interés en una ciudad tú luego podrás repetirla e ir a los sitios más interesantes sin tener que estar buscándolos con engorrosos mapas y folletos.

GeoPoint también dispone de un servicio de GPS con el cual podrás dirigirte sin ningún problema al GeoPoint que desees ya que está conectado con el servicio de googleMaps.

ANALISIS

Estado del arte

Cuando se pensó en desarrollar GeoPoint, se vio una gran variedad de usos posibles para un almacenamiento de geolocalizaciones mediante avisos Push.

La posibilidad de poder acceder a ellos solo con el dispositivo móvil e añadirlos a una base de datos y recuperarlos en cualquier momento y que con un solo click te lleve al destino si tener que estar escribiendo direcciones en buscadores y en los tediosos GPS, es una gran ventaja para el usuario.

También se valoró el desarrollo de la interfaz y de su diseño, dándole un aspecto atractivo y de una usabilidad sencilla. Además de una interfaz muy limpia e intuitiva, se pretende dar al usuario una interactividad con la aplicación y con sus amigos o grupos de amigos pudiendo agregar GeoPoint a su cuenta de usuario y así tenerlos registrados e ir a visitarlos cuando estés por la zona, y si acaso te pasas del destino o no sabes exactamente por donde esta, GeoPoint, te notifica que has entrado o has salido del punto de interés.

GeoPoint también te da la posibilidad de modificar esos puntos de interés y borrarlos. Solo con pulsar en el mapa te da la opción de guardar un GeoPoint nuevo y te muestra la dirección de la calle, población, localidad... que nos devuelve el servicio Advance REST client de Google y tan simple como agregarle el título, descripción y el radio de aviso del

servicio “Geofencing” de Google ya tendríamos nuestro GeoPoint almacenado y listo para ubicarlo en el mapa cuando lo necesitemos.

FASE DE DISEÑO

Lenguajes de programacion

Para el desarrollo de GeoPoint será necesario conocer los principios básicos del desarrollo de aplicaciones Android, y conocer los diferentes lenguajes de programación como java, php y mysql.

Android: Es un sistema operativo basado en el núcleo Linux. Fue diseñado principalmente para dispositivos móviles con pantalla táctil, como teléfonos inteligentes, tablets y también para relojes inteligentes, televisores y automóviles.

Aunque la mayoría de las aplicaciones están escritas en Java, no hay una máquina virtual Java en la plataforma. El bytecode Java no es ejecutado, sino que primero se compila en un ejecutable Dalvik y se ejecuta en la Máquina Virtual Dalvik, Dalvik es una máquina virtual especializada, diseñada específicamente para Android y optimizada para dispositivos móviles que funcionan con batería y que tienen memoria y procesador limitados. A partir de la versión 5.0, se utiliza el Android Runtime (ART). El soporte para J2ME puede ser agregado mediante aplicaciones de terceros como el J2ME MIDP Runner.

Incluye un emulador de dispositivos, herramientas para depuración de memoria y análisis del rendimiento del software. Inicialmente el entorno de desarrollo integrado (IDE) utilizado era Eclipse con el plugin de Herramientas de Desarrollo de Android (ADT). Ahora se considera como entorno oficial Android Studio.

JAVA: Es un lenguaje de programación de propósito general, concurrente, orientado a objetos que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo, lo que quiere decir que el código que es ejecutado en una plataforma no tiene que ser recompilado para correr en otra. Java es, a partir de 2012, uno de los lenguajes de programación más populares en uso, particularmente para aplicaciones de cliente-servidor de web, con unos 10 millones de usuarios reportados.

Los objetivos al desarrollar java eran implementar una máquina virtual y un lenguaje con una estructura y sintaxis similar a C++.

PHP: Es un lenguaje de programación de uso general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico.

Fue uno de los primeros lenguajes de programación del lado del servidor que se podían incorporar directamente en el documento HTML en lugar de llamar a un archivo externo que procese los datos.

El código es interpretado por un servidor web con un módulo de procesador de PHP que genera la página web resultante. PHP ha evolucionado por lo que ahora incluye también una interfaz de línea de comandos que puede ser usada en aplicaciones gráficas independientes. Puede ser usado en la mayoría de los servidores web al igual que en casi todos los sistemas operativos y plataformas sin ningún costo.

PHP puede ser desplegado en la mayoría de los servidores web y en casi todos los sistemas operativos y plataformas sin costo alguno.

El lenguaje PHP se encuentra instalado en más de 20 millones de sitios web y en un millón de servidores. El número de sitios basados en PHP se ha visto reducido progresivamente en los últimos años, con la aparición de nuevas tecnologías como Node.JS, Golang, ASP.NET, etc. PHP es también el módulo Apache más popular entre las computadoras que utilizan Apache como servidor web.

MySQL: Es un sistema de gestión de bases de datos relacional y está considerada como la base datos open source más popular del mundo, y una de las más populares en general junto a Oracle y Microsoft SQL Server, sobre todo para entornos de desarrollo web.

Existen varias interfaces de programación de aplicaciones que permiten, a aplicaciones escritas en diversos lenguajes de programación, acceder a las bases de datos MySQL, incluyendo C, C++, C#, Pascal, Delphi (vía dbExpress), Eiffel, Smalltalk, Java (con una implementación nativa del driver de Java), Lisp, Perl, PHP, Python, Ruby, Gambas, REALbasic (Mac y Linux), (x)Harbour (Eagle1), FreeBASIC, y Tcl; cada uno de estos utiliza una interfaz de programación de aplicaciones específica. También existe una interfaz ODBC, llamado MyODBC que permite a cualquier lenguaje de programación que soporte ODBC comunicarse con las bases de datos MySQL. También se puede acceder desde el sistema SAP, lenguaje ABAP.

FRAMEWORKS

Para un mejor desarrollo y una correcta implementación de GeoPoint se ha investigado varias librerías y diferentes frameworks y de los cuales se han implementado los que más nos beneficiaban y mejor utilidad nos ofrecían.

El uso de **ListViews**, pudiendo clicar en cada registro gracias a las implementaciones de `View.OnClickListener`. Y con `GoogleMap.OnMarkerClickListener` podremos ver la localización en el mapa, el título y descripción del geoPoint.

El uso de **fragmentMaps** para incorporar un mapa de google en el cual se puede interactuar con el gracias a implementar `OnMapReadyCallback`, `GoogleMap.OnInfoWindowClickListener` y `GoogleMap.OnMapClickListener` y recibir la localización mediante `LocationListener` y añadirle marcadores mediante `GoogleMap.OnMapClickListener` y `GoogleMap.OnMarkerClickListener`.

Geoposicionamiento: Para el servicio de geoposicionamiento se ha usado los servicios de localización GPS del dispositivo móvil con la geolocalización de Google y sus servicios de geofencing y el Advance REST client que te devuelve los parámetros de localización tales como calles, provincias etc... y la implementación de las librerías GoogleMaps, GoogleApiClient, JSON.

PendingIntent. Este es un servicio que se le envía una petición a una aplicación externa y que permite a la solicitud utilizar los permisos de la aplicación para ejecutar una parte predefinida de código. Esto lo usamos para pasarle a la Api "geofencing" de Google la petición de que nos avise cuando estemos en el área marcado de nuestro GeoPoint, esto se ejecuta al inicio de la aplicación y los pendingIntent se quedan esperando en los servicios de Google hasta que entras en el área y este entonces te devuelve la llamada que nosotros recogemos y mostramos con los servicios NotificationManager de Android.

GoogleMaps: Google Maps es un servidor de aplicaciones de mapas en la web. Ofrece imágenes de mapas desplazables, así como fotografías por satélite del mundo e incluso la ruta entre diferentes ubicaciones o imágenes a pie de calle con Google Street View.

Google Maps ofrece la capacidad de realizar acercamientos y alejamientos para mostrar el mapa. El usuario puede controlar el mapa con el mouse o las teclas de dirección para moverse a la ubicación que se desee. Para permitir un movimiento más rápido, las teclas «+» y «-» pueden ser usadas para controlar el nivel de zoom.

Conexión a Base de Datos: Volley es una librería desarrollada por Google para optimizar el envío de peticiones Http desde las aplicaciones Android hacia servidores externos.

Es un cliente Http creado para facilitar la comunicación de red en las aplicaciones Android. A diferencia de la interfaz HttpURLConnection, Volley está totalmente enfocado en las peticiones, evitando la creación de código repetitivo para manejar tareas asíncronas por cada petición o incluso para parsear los datos que vienen del flujo externo.

DialogInterface es una librería que te permite mostrar diálogos emergentes en los que muestras e interactúas con el usuario y a la vez te permite seguir a la aplicación trabajando por detrás en otra venta o acción.

APLICACIÓN

La utilidad de GoePoint se basa en ofrecer un servicio de almacenamiento con localización para las personas que posean un dispositivo con Android, conexión a internet y GPS incorporado en el dispositivo mostrando así los puntos y avisando cuando entres en ellos.

GeoPoint está dirigido a todos los públicos, conteniendo una interfaz bonita, limpia e intuitiva y capacitando a que cualquier usuario pueda utilizarla sin necesidad de que tenga conocimientos previos, siendo así, una aplicación apta para cualquier persona con acceso a internet y un terminal móvil con localización GPS.

Ademas la interacción con amigos o grupos de amigos es una ventaja que hace que la gente esté interesada en ella y que aproveche su usabilidad al máximo sacándole partido a las numerosas funcionalidades que tiene y que se irán implementando más adelante en futuras actualizaciones, como la posibilidad de crear rutas y poder avisar a los contactos del grupo por las diferentes fases que pasa el recorrido planeado, la programación de un juego (Gincana) que se base en llegar a un punto del mapa y cuando estes en él, se te notifiquen unas instrucciones y así completar el punto de control y saltar a la siguiente prueba y llevar una agenda de localizaciones personales interesantes.

GeoPoint es una aplicación que no usa apenas memoria del móvil ya que los registros de los geoPints se guardan en una base de datos de un servidor externo, esto es una gran ventaja para el usuario porque cada vez la gente es más reacia a instalar aplicaciones que consuman mucha memoria y recursos del teléfono.

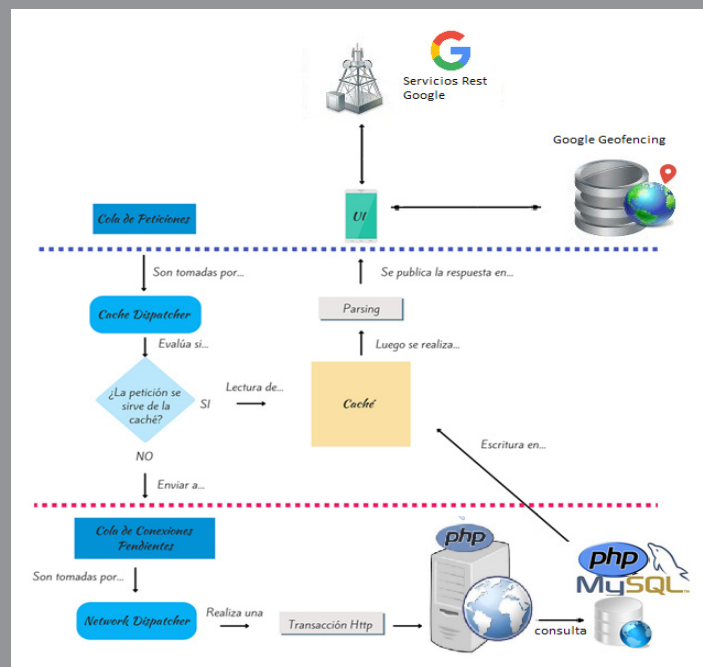
La aplicación esta compuesta:

- > De un acceso sencillo en el que se precisa estar registrado.
- > De una pantalla en la que veras un mapa con tus marcadores, un listado de GeoPoints personales, y una lista de usuarios amigos y los marcadores de sus GeoPoints con el titulo y la descripción de los mismos y con la posibilidad de guardartelos en tu lista.
- > Y una pantalla donde rellenaremos o modificaremos los campos de nuestro GeoPoint y lo añadiremos a la BBDD.
- > De una seccion en la que añadiremos a los usuarios amigos.

Esquema General de Funcionamiento:

La aplicación recoge la localización del dispositivo, al pulsar el mapa, estas se le mandan a los servicios REST de Google para que nos devuelva la localización con todos sus parametros de Poblacion, calle, numero, localidad etc...

Si los parametros de vuelta son correctos se los pasamos a la activity y rellenamos los EditText con esos valores, si no se le mostraran vacios para que el usuario los rellene.



Una vez que el usuario rellene los campos y haga una petición por Http este conectará con Volley, que procesará la petición y hará ciertas comprobaciones antes de enviar los datos al servidor para su procesamiento.



El servidor externo que es quien se encargara de procesar la peticion y enviar la consulta a la BBDD, una vez procesada la consulta en la BBDD, estos datos son devueltos por el servidor en formato JSON, y seran recibidos en la aplicacion para procesarlos.

Una vez recibidos los datos y procesados se llama a los servicios "Geofencing" de Google para cargar las alertas de los GeoPoint y asi puedan ser notificadas a los usuarios.

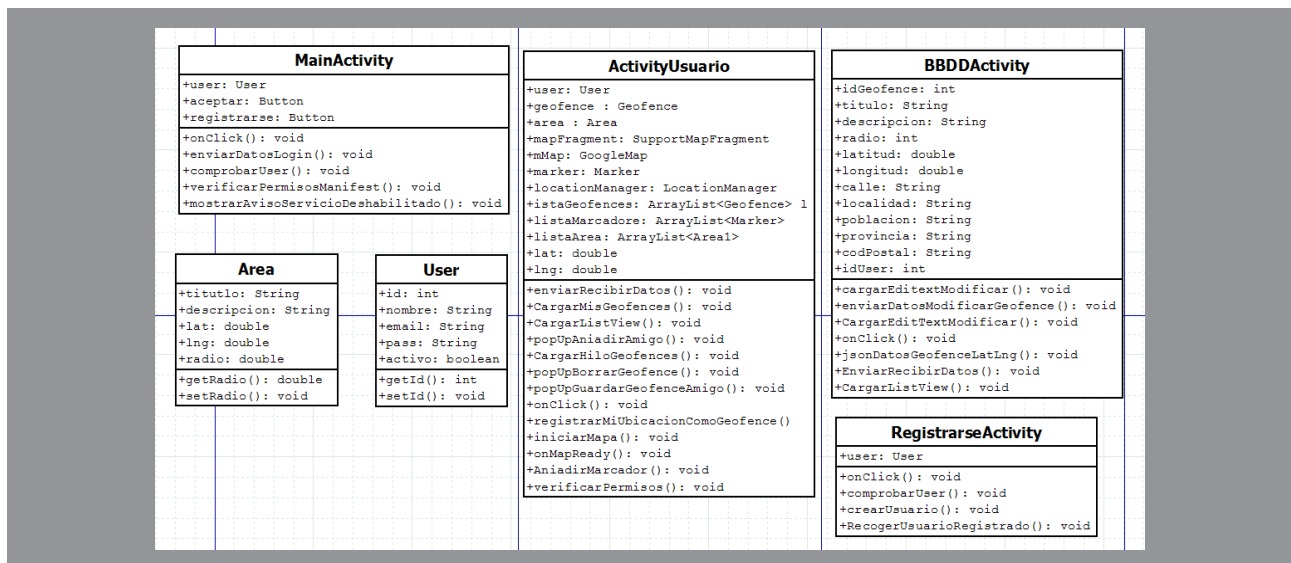
Diagrama de clases:

Creamos un objeto Geofences al que le vamos asignando todos los parámetros que recogemos a la hora de guardar el GeoPoint.

Creamos un Usuario al que le pasamos los parámetros de registro o de login y que luego recogemos en el contexto de Application y asi le podemos llamar a en todas las activities.

Creamos un Area al que le vamos asignando los valores que le queremos pasar a la Api "Geofences" de Google para que nos muestre los avisos.

Creamos MainActivity que es la clase donde arranca la aplicacion y es procesado el registro del usuario.



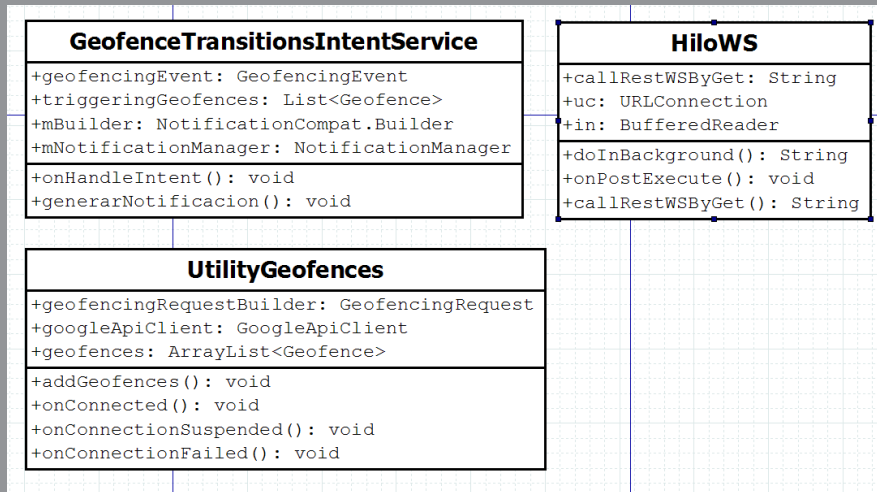
Creamos ActivityUsuario que es la encargada de mostrar todos los datos y de recoger y procesar los avisos y peticiones del mapa, a su vez tambien se encarga de mostrar los diferentes GeoPoint y Amigos del usuario.

La clase BBDDActivity es la encargada de almacenar y guardar los geofences en la BBDD y a su vez de procesar los datos devueltos por la clase HiloWS.

HiloWS esta encargada de enviar y recibir peticiones a Google y procesar los datos de la localizacion que le hemos pasado.

UtilityGeofences se encarga de recoger los geofences pasados, procesarlos y enviárselos a la API "Geofences" de Google.

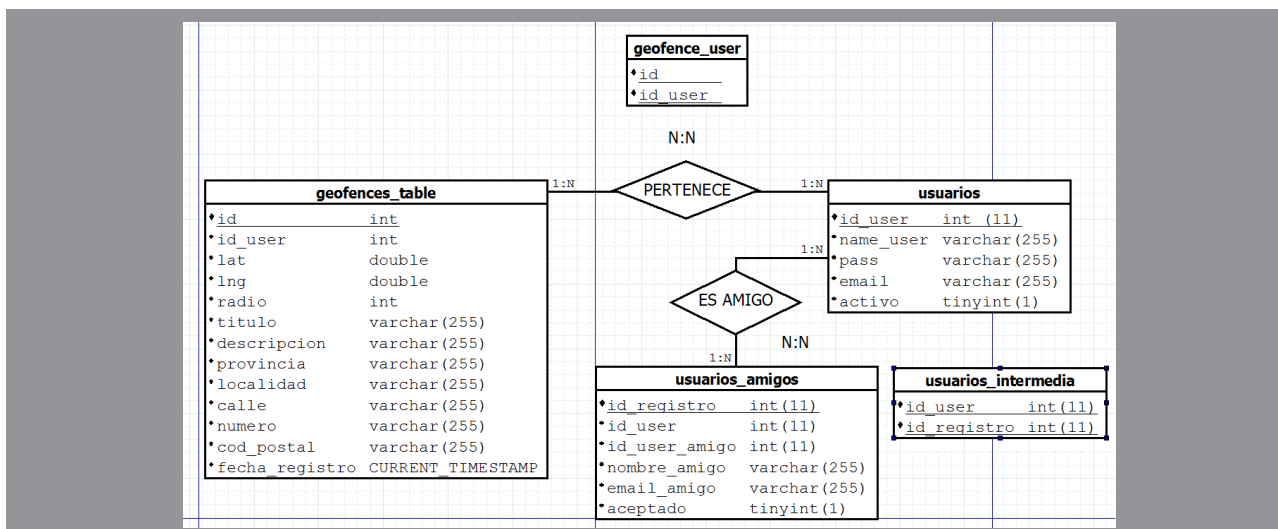
GeofenceTransitionsIntentService es el encargado de procesar los Intent que le llegan de "Geofencing" de Google y transformarlos en una notificación Push.



MODELO ENTIDAD RELACION Y SUS TABLAS:

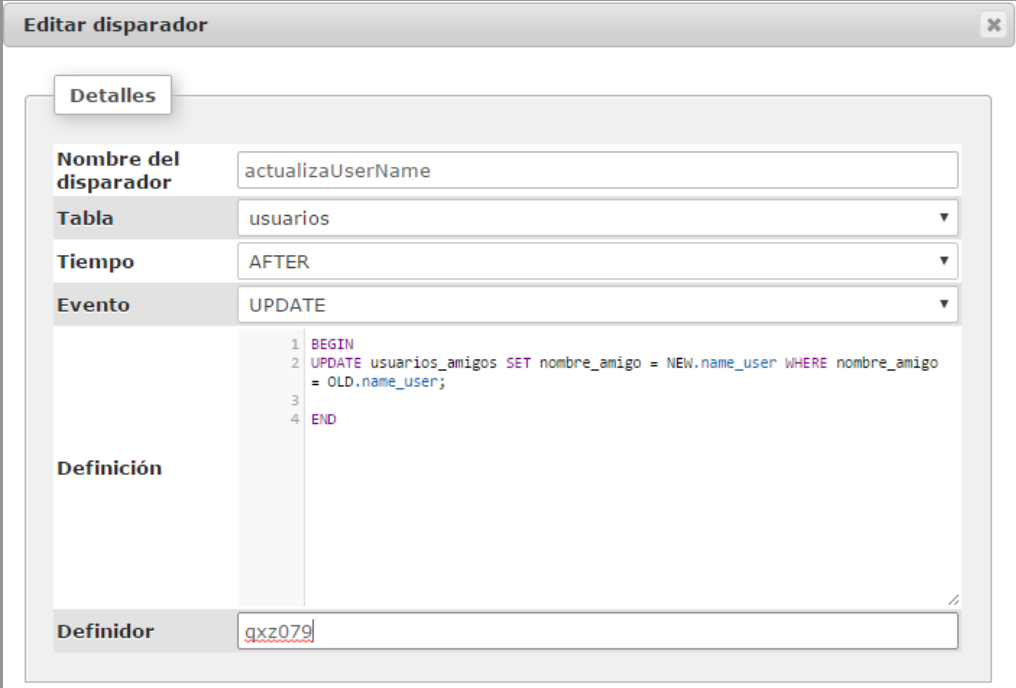
En la tabla geofences_table recogemos los campos del id, el id_usuario y los datos de las latitudes y longitudes radio y otros datos de interés que queremos almacenar para que el usuario identifique sus puntos de interés fácilmente.

En la tabla de usuarios recogemos el id y los datos del usuario además de un campo que nos dice si el usuario está activo o dado de baja del servicio.



Y en la tabla de usuarios amigos recogemos los id de los usuarios que se han agregado como amigos, el campo nombre que se ira actualizando mediante un TRIGGER AFTER UPDATE y un campo aceptado que tendrá que estar aceptado para que se muestre como amigo.

Creamos un trigger de actualización para después de que un usuario modifique sus datos en su perfil a los usuarios que tenga agregados como amigos se les mostrara el nombre nuevo en el listView de amigos.



Editar disparador

Detalles

Nombre del disparador	actualizaUserName
Tabla	usuarios
Tiempo	AFTER
Evento	UPDATE

Definición

```
1 BEGIN
2 UPDATE usuarios_amigos SET nombre_amigo = NEW.name_user WHERE nombre_amigo
  = OLD.name_user;
3
4 END
```

Definidor qxz079

FASE DE DESARROLLO

HISTORICO DE DESARROLLO:

1. Se finalizo el diseño de la BBDDD.
2. Se finalizo el diseño de la aplicacion.
3. Se finalizo la incorporacion de GoogleMaps y las implementaciones de los metodos.
4. Se finalizo la conexion con la BBDD.
5. Se finalizo el proceso de registro/login de usuarios.
6. Se finalizo la recogida de datos del servicio Advance REST Client de Google.
7. Se finalizo la insercion de los datos de los GeoPoint.
8. Se finalizo el proceso de conexion con Google "Geofencing".
9. Se finalizo la recogida de datos devueltos por Google "Geofencing" y se proceso los avisos Push.
10. Se finalizo la recogida y mostrado de los GeoPoint del usuario.
11. Se finalizo el guardado de usuarios amigos.
12. Se finalizo la recogida de los usuarios amigos.
13. Se finalizo el muestado de los GeoPoint de los amigos.
14. Se finalizo la agregacion de los GeoPoint del amigo al usuario.

PROCESO DE DESARROLLO:

1. Lo primero que se hizo fue empezar diseñando lo que iba a ser la interfaz del usuario y que opciones tendría para el usuario y su disposición en la pantalla.
2. Se creó el proyecto, se le crearon las clases para almacenar los usuarios y los GeoPoint.
3. Se diseñó la Base de Datos con las tablas y los campos necesarios para almacenar los diferentes registros.
4. Luego se diseñó un trigger para la actualización de algunos campos de la tabla.
5. Se le implemento una BBDD interna.

6. Esta opción se descartó en el momento que se valoró el recurso de memoria y la ralentización del dispositivo.
7. Se implementó una BBDD en un servidor externo mediante mysql.
8. Se creó la conexión de la base de datos mediante lenguaje PHP.
9. Se conectó la aplicación a la BBDD mediante la librería Volley.
10. Se recogieron los datos correctamente.
11. Se creó la Activity de Login y de registro de usuarios.
12. Se hicieron las comprobaciones correspondientes de login/registro con los servicios Http y procesándolas con PHP.
13. Se crea un Objeto Usuario.
14. Se dieron los permisos en el manifest.xml.
15. Se recoge y se comprueba que los servicios de Localización del dispositivo están conectados y están activos.
16. Se comprobó que los servicios de datos están activados para poder procesar las peticiones.
17. Se crea un MapFragment.
18. Se implementa el mapa con los eventos de pulsaciones.
19. Se recogieron esos eventos.
20. Se procesaron y se guardaron los datos de Localización devueltos por el evento OnClickMap del mapa de Google.
21. Se le muestra un aviso de que si quiere añadir la localización a la BBDD.
22. La localización recogida se la pasamos a la Api de Google: Advance REST Client.
23. Esta te devuelve los detalles de la localización (calle, número, provincia etc...) en formato JSON.
24. Procesamos estos datos y los recogemos.
25. Creamos la activity para agregar/modificar GeoPoint y cargamos al usuario los datos devueltos por la Api ARC de Google.
26. Recogemos todos los datos que el usuario quiere guardar y se los pasamos a la BBDD mediante Http.
27. Recogemos esos datos en el servidor, los procesamos y le devolvemos la confirmación del proceso.

28. Se crea un Objeto GeoPoint.
29. Se añade a una lista de GeoPoint.
30. Se crea un botón registrar mi ubicación.
31. Se recoge el evento del botón y se procesa la petición de agregar mi posición actual.
32. Se recoge la respuesta.
33. Se crea un Objeto GeoPoint.
34. Se añade a una lista de GeoPoint.
35. Se crea un listView en el que cargaremos los GeoPoints del usuario.
36. Se implementan los métodos para ver que elemento de la lista a sido pulsado y asi poder recoger sus valores y mostrarlos en el mapa posteriormente.
37. Se crea la petición para recoger los GeoPoints según el id del usuario guardado en la base de datos.
38. Se procesa la petición y se devuelven los GeoPoints del usuario
39. Se recogen los GeoPoint.
40. Se añade un marcador con un titulo al mapa según las coordenadas de cada GeoPoint.
41. Se recoge el evento de pulsación del marcador y se le implementa la opcion de activar los mapas de google del móvil y mostrarte la ruta desde donde te encuentras actualmente hasta tu GeoPoint pulsado.
42. Se le muestran al usuario en la lista.
43. Se recoge el GeoPoint pulsado y se le enfoca en el mapa la posición del mismo.
44. Se recoge el evento de pulsación larga en el listView y se le muestra al usuario un dialogo de que si quiere modificar o borrar ese GeoPoint.
45. Se recoge la opcion de modificar cargándole al usuario en la activity de añadir/modificar los valores de su geofence para que pueda modificarlos.
46. Se recoge los campos de la modificación.
47. Se envían y se procesan.
48. Se recoge la respuesta de la acción modificar.
49. Se recoge la opcion de borrar el GeoPoint.
50. Se envía la petición de borrado del GeoPoint.

51. Se Procesa la petición en el servidor
52. Se recoge la respuesta de la acción borrar.
53. Se implementa el botón de mis amigos.
54. Se recoge el evento de pulsación.
55. Se envía la petición para que nos devuelva los amigos del usuario.
56. Se procesa la petición y se devuelve un JSON con los diferentes amigos.
57. Se recogen los amigos.
58. Se muestran en el listView recogiendo el evento de pulsación de la posición en la lista.
59. Se realiza la petición de los GeoPoint del usuario amigo pulsado.
60. Se procesa la petición.
61. Se recoge el JSON devuelto con los GeoPoint del amigo.
62. Se le cargan los GeoPoint del usuario amigo.
63. Se añaden al mapa los marcadores de los GeoPoint.
64. Se recoge el evento de pulsación corta para enfocar en el mapa el GeoPoint y la posibilidad de iniciar la ruta GPS.
65. Se recoge el evento de pulsación larga y se le muestra al usuario si quiere guardar el GeoPoint de su amigo.
66. Si se confirma que si, se le envía la petición de añadir con el id del GeoPoint seleccionado.
67. Se procesa la petición y se devuelve la confirmación.
68. Se recoge el evento de pulsación larga en el listView de amigos y se le muestra al usuario que si desea borrar ese amigo.
69. Se envía la petición de borrado y se procesa para proceder al borrado.
70. Se recoge la confirmación de borrado y se le notifica.
71. Se crea una ventana para añadir amigo.
72. Se recoge el evento de añadir amigo.
73. Se comprueba que no esta añadido ya como amigo.
74. Se añade en la BBDD.
75. Se recoge la respuesta de la petición.

- 76. Se procesa y se le notifica al usuario
- 77. Se carga el listView de los usuarios amigos.

PARTES PRINCIPALES EN LAS QUE SE BASA LA APP:

➤ Envío de peticiones con volley

Este componente actúa como una interfaz de alto nivel, liberando al programador de la administración de hilos y procesos tediosos de parsing, para permitir publicar fácilmente resultados en el hilo principal.

Entre sus características más potenciadoras podemos encontrar:

- > Procesamiento concurrente de peticiones.
- > Priorización de las peticiones, lo que permite definir la preponderancia de cada petición.
- > Cancelación de peticiones, evitando la presentación de resultados no deseados en el hilo principal.
- > Gestión automática de trabajos en segundo plano, dejando de lado la implementación manual de un framework de hilos.
- > Implementación de caché en disco y memoria.
- > Capacidad de personalización de las peticiones.
- > Provee información detallada del estado y flujo de trabajo de las peticiones en la consola de depuración.

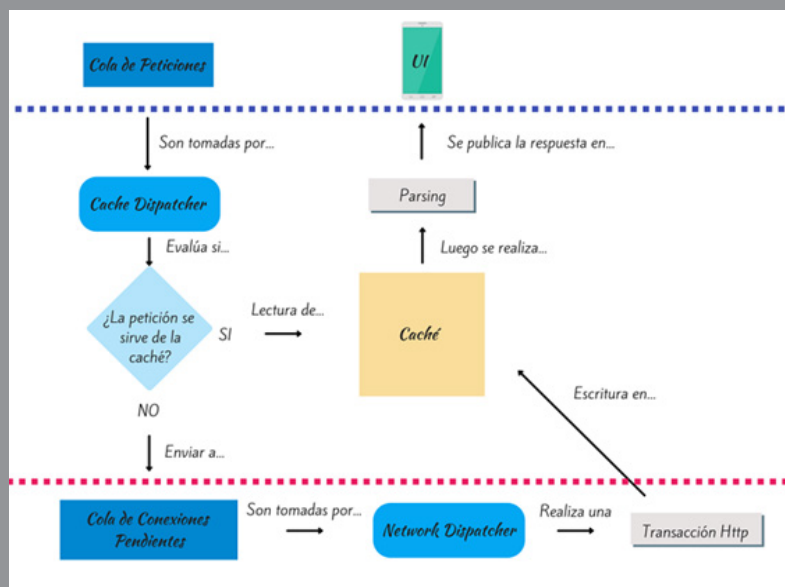
Volley posee varios componentes que optimizan la administración de las peticiones generadas desde las aplicaciones Android. La gestión comienza en una Cola de Peticiones que recibe cada una de las peticiones generadas, donde son previamente priorizadas para su realización.

Luego son seleccionadas por un elemento llamado Cache Dispatcher, cuya función es comprobar si la respuesta de la petición actual puede ser obtenida de resultados previos guardados en caché. Si es así, entonces se pasa a parsear la respuesta almacenada y luego se presenta al hilo principal. En caso negativo, se envía la petición a la Cola de Conexiones Pendientes, donde reposan todas aquellas peticiones que están por ejecutarse.

Luego entra en juego un componente llamado Network Dispatcher, el cual se encarga de seleccionar las peticiones pendientes de la cola, para realizar las respectivas transacciones Http hacia el servidor. Si es necesario, las respuesta de estas peticiones se guardan en caché, luego se parsean y finalmente se publican en

el hilo principal.

Aunque esta es una definición básica, es muy concisa y significativa para comprender el proceso que realiza Volley al surgir peticiones. A continuación se muestra una ilustración que resume el flujo:



La implementación de Volley elimina la implementación de hilos manualmente con Tareas asíncronas como era el caso del uso del cliente `HttpURLConnection`.

Aunque Volley posee ventajas enormes para el procesamiento de peticiones, no significa que se debe usar en cada petición que hagamos. Esta librería tiene limitaciones con la descarga de información demasiado extensa, ya que su operación se basa en salvaguardas en cache, lo que haría lento el proceso con datos voluminosos.

➤ Conexión con los servicios Geofencing.

Para conectar con los servicios de Geofencing necesitamos pasarle una latitud, una longitud y un radio.

Al método le pasamos una lista de Areas que es de donde recogemos la latitud, longitud y radio del punto que queremos activar los avisos, luego de campo de la lista construimos un área y se lo pasamos al servicio de geofencing y creamos las notificaciones de esos GeoPoint.

Luego una vez construido y enviados los geofences, en el método `@Override` de los servicios: `onConnected(@Nullable Bundle bundle)`; Se le añade a la clase `GeofenceTransitionsIntentService` un `PendingIntent` que se recoge en la clase `UtilityGeofences` y se procesa para mostrarle el mensaje de entrada al GeoPoint, salida del mismo o que se ha desactivado el GPS.

```
public GeofenceTransitionsIntentService() { super("GeofenceTransitionsIntentService"); }

@Override
protected void onHandleIntent(@Nullable Intent intent) {
    GeofencingEvent geofencingEvent = GeofencingEvent.fromIntent(intent);
    if (geofencingEvent != null && geofencingEvent.hasError()) {
        //generarNotificacion(App.getAppContext(), "ERROR de geofence: "+geofencingEvent.getErrorCode());
        generarNotificacion(App.getAppContext(), "Se ha desactivado el GPS, la aplicacion no le avisara de los puntos de Interes");
        return;
    }

    // Get the transition type.
    int geofenceTransition = geofencingEvent.getGeofenceTransition();
    String geo = "";
    // Test that the reported transition was of interest.
    if (geofenceTransition == Geofence.GEOFENCE_TRANSITION_ENTER) {
        final List<Geofence> triggeringGeofences = geofencingEvent.getTriggeringGeofences();

        for (Geofence geofence: triggeringGeofences) {
            geo += geofence.getRequestId() + " ";
        }
        // Send notification and log the transition details.
        generarNotificacion(App.getAppContext(), "Has entrado en: "+geo);
    } else if (geofenceTransition == Geofence.GEOFENCE_TRANSITION_EXIT) {
        final List<Geofence> triggeringGeofences = geofencingEvent.getTriggeringGeofences();
        for (Geofence geofence: triggeringGeofences) {
            geo += geofence.getRequestId() + " ";
        }
        generarNotificacion(App.getAppContext(), "Has salido de: "+geo);
    } else {
        generarNotificacion(App.getAppContext(), "Transición no contemplada");
    }
}

public static void generarNotificacion(Context context, String mensaje){
    NotificationCompat.Builder mBuilder = new NotificationCompat.Builder(context)
        .setSmallIcon(R.mipmap.favicon)
        .setContentTitle("GeoPoint")
        .setContentText(mensaje)
        .setAutoCancel(true);

    mBuilder.setVibrate(new long[]{250, 500, 250, 500});
    mBuilder.setSound(RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION));

    NotificationManager mNotificationManager = (NotificationManager) context.getSystemService(Context.NOTIFICATION_SERVICE);
    mNotificationManager.notify(randInt(), mBuilder.build());
}
```

>Notificaciones Push de los servicios Geofencing.

Las notificaciones son creadas en el momento que los servicios de Google "Geofencing" retornan el PendingIntent. en ese momento se comprueba que valor devuelve y se le muestra la notificacion correspondiente.

```
private GeofencingRequest.Builder geofencingRequestBuilder;
private GoogleApiClient googleApiClient;

public void addGeofences(final List<Areal> puntosControl) {
    if (puntosControl != null && puntosControl.size() > 0) {
        final ArrayList<Geofence> geofences = new ArrayList<>(puntosControl.size());

        for (int i = 0; i < puntosControl.size(); i++) {
            final Areal puntoControl = puntosControl.get(i);

            geofences.add(new Geofence.Builder()
                .setRequestId(String.valueOf(puntoControl.titulo))
                .setCircularRegion(puntoControl.lat, puntoControl.lng, puntoControl.radio)
                .setExpirationDuration(Geofence.NEVER_EXPIRE)
                .setTransitionTypes(Geofence.GEOFENCE_TRANSITION_ENTER | Geofence.GEOFENCE_TRANSITION_EXIT)
                .build());
        }

        geofencingRequestBuilder = new GeofencingRequest.Builder();
        geofencingRequestBuilder.addGeofences(geofences);

        GoogleApiClient.Builder b = new GoogleApiClient.Builder(App.getAppContext());
        b.addApi(LocationServices.API);
        b.addConnectionCallbacks(this);
        b.addOnConnectionFailedListener(this);

        googleApiClient = b.build();
        googleApiClient.connect();
    }
}
```

➤ Rercogida de datos de los servicios Rest de Google.

Para pasarle los datos a los servicios Advanced Rest Client de Google creamos un metodo en el que recogemos una latitud y una longitud y se la pasamos al HiloWS, luego capturamos el JSONObject que nos devuelve Google, lo recorremos y vamos añadiendo a los EditText los valores.

➤ Actualizacion automatica de campos en la BBDD.

El nombre de usuario en la base de datos puede variar si el usuario decide cambiarle, de esta forma el campo nombre_amigo que tenemos en la tabla usuarios_amigos se quedaria con el nombre del usuario recogido en el momento que se agrega como amigo.

Para esto necesitamos de la creacion de un Trigger After Update para actualizar el nombre y asi cuando le mostremos al usuario el nombre de su amigo le aparezca el nombre actual.

FASE DE PRUEBAS:

Comprobaciones de registro correcto.

Se realiza la comprobacion de los registros de usuario comprobando que el email no este

registrado en la BBDD, una vez realizadas estas comprobaciones el funcionamiento de la aplicacion es el correcto.

Comprobaciones de localización correcta.

Se comprueba que la localizacion se hace correctamente viendo en el propio mapa de la aplicacion donde esta apuntando el Marker y verificando que nos encontremos en ese punto.

Comprobaciones de localización correcta en sitios de difícil cobertura.

Se verifica que con alguna dificultad la localizacion es correcta en sitios de difícil cobertura y de difícil acceso, aun asi en un breve tiempo conseguimos que el sistema de localizacion del dispositivo nos ubique correctamente y nos notifique que hemos entrado en el radio de nuestro GeoPoint.

Comprobacion del limite de radio de la aplicacion.

Se hizo una comprobacion con un GeoPoint que se le asigno un radio de 50 metros, se realizo un test de 5 pruebas para ver a que distancia nos notificaria la aplicacion que ya estabamos dentro del radio y nos dio un valor medio de menos de 2 metros de error.

Luego se realizo otra comprobacion con un GeoPoint de radio 100 metros y el resultado del test fue muy similar al anterior, el margen de error medio que nos entregaron las pruebas fueron un resultado de 1,5 metros.

Comprobaciones de comunicación y guardado en la BBDD correctas.

Se comprueba el resultado de las peticiones http con buena y escasa cobertura y los resultados son satisfactorios para el usuario ya que los resultados devueltos son en formato String y no necesita gran cantidad de trafico de datos.

Comprobaciones de peticiones a servicios Google correctas.

Se comprueba que cuando un usuario pulsa el mapa y quiere agregar un GeoPoint, los valores que se le envian y que devuelve Google son correctos.

Comprobación de avisos Geofencing correctos.

Se comprueba repetidamente que los avisos son bien recibidos y en infinidad de situaciones diferentes la aplicacion responde correctamente.

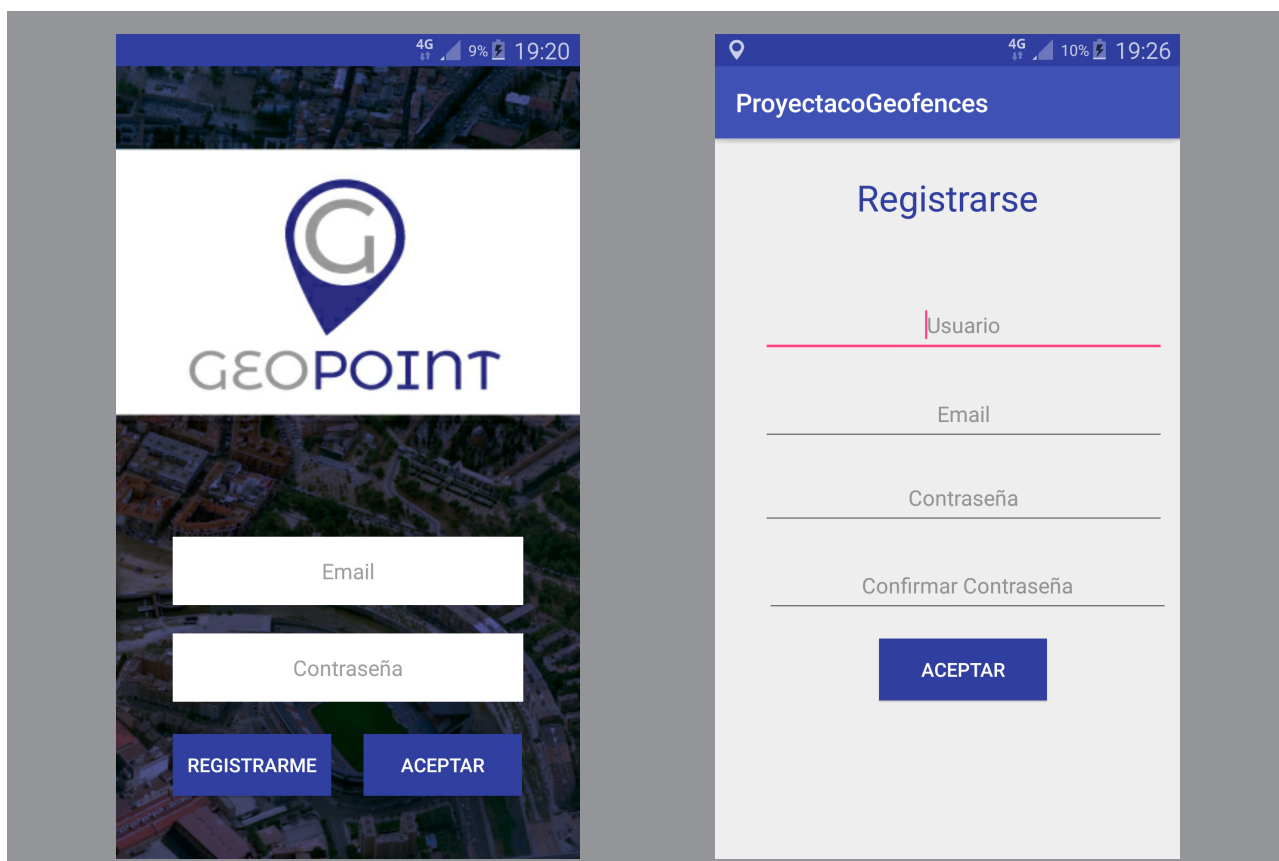
MANUAL DEL USUARIO

Registro y login:

La opción de registro te permite registrarte facilmente y sin demoras de tiempo absurdas, con solo 3 campos de un formulario estaras registrado u listo para usar GeoPoint.

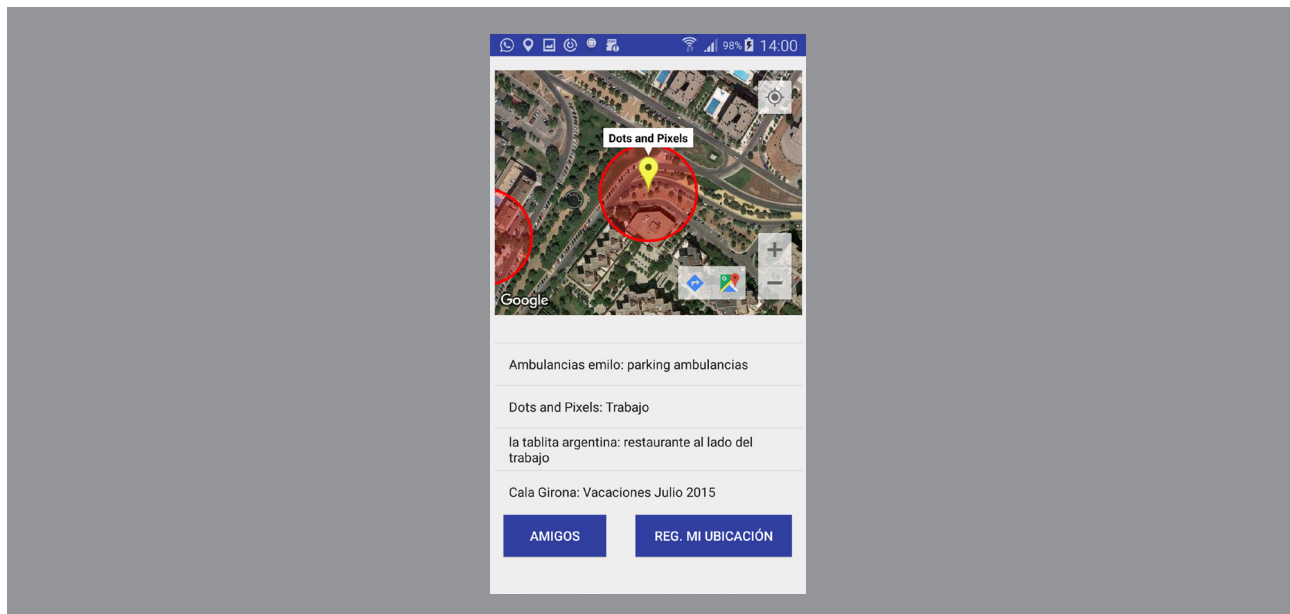
- Email, nombre usuario y contraseña.

Y en caso de estar ya registrado, accederás mediante el email y una contraseña.

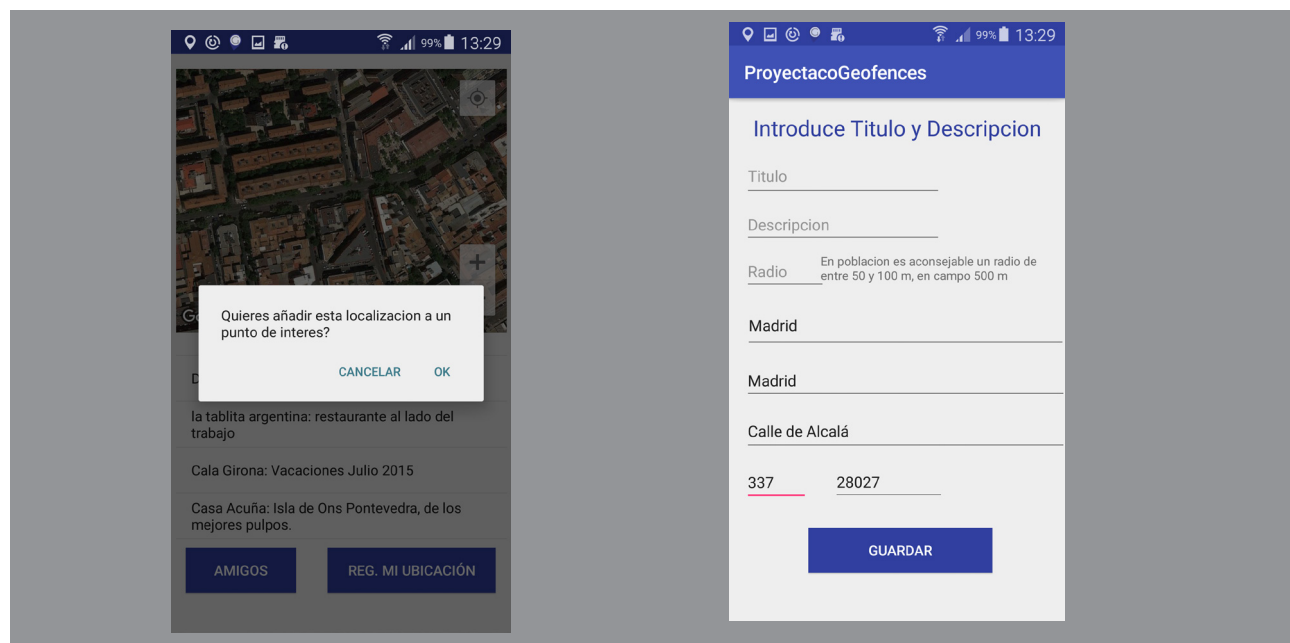


Activity principal:

Una vez entrado en GeoPoint se mostrara un mapa con los diferentes GeoPoint guardados, el usuario podrá visualizarlos haciendo click sobre ellos y tendrá la opción de ser guiado mediante GoogleMaps.

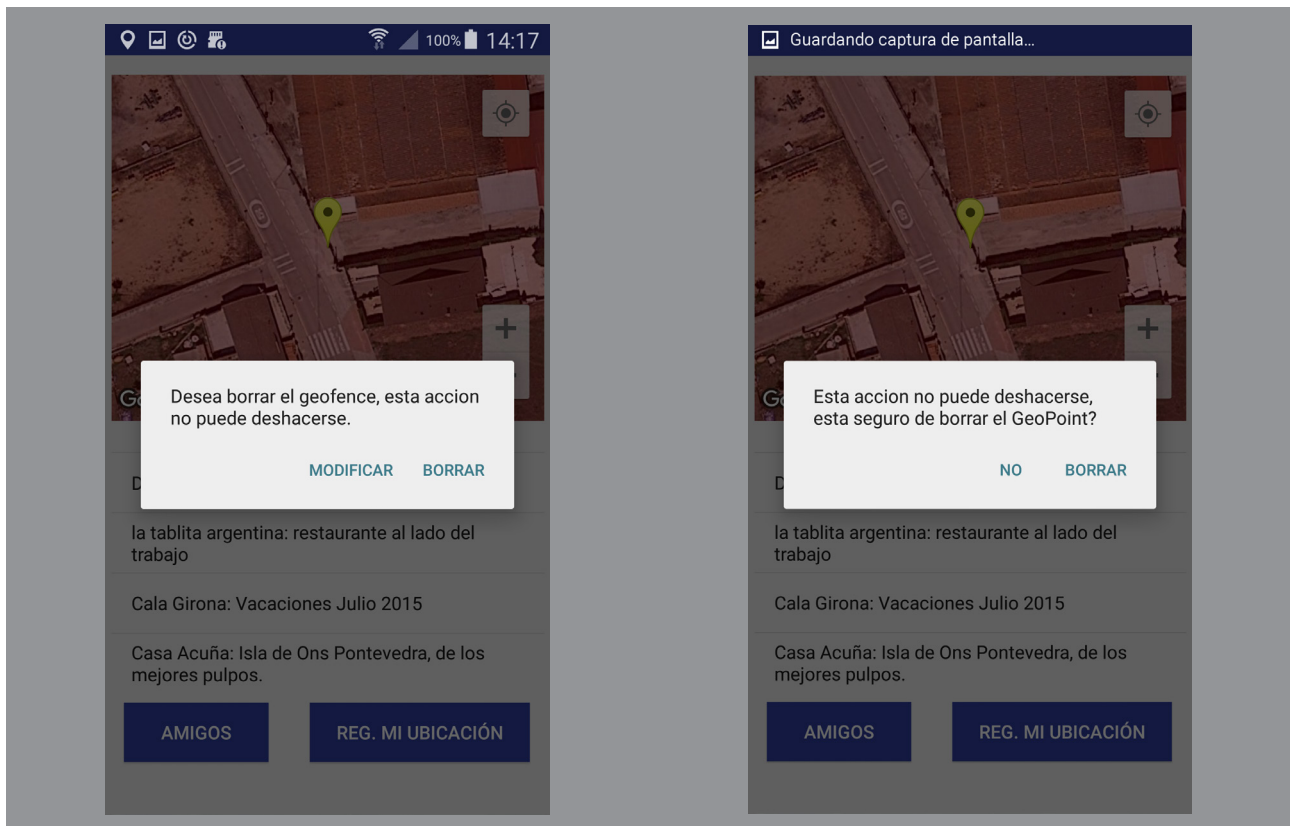


Un mapa en el cual el usuario podrá ubicarse y podrá seleccionar la localizacion que desea guardar, en caso de aceptar el guardado se le enviara al usuario a la activity de agregar GeoPoint.



Si en la lista de GeoPoint personales dejamos pulsado cualquiera de ellos durante un breve periodo de tiempo al usuario le saltara un dialogo que le dira si quiere modificar o borrar el geofence.

Si se acepta el borrado, le volvera a salir un dialogo que le dira si realmente quiere borrar el GeoPoint, y le avisara de que esta opcion no puede deshacerse.



Activity Guardado/modificado GeoPoint:

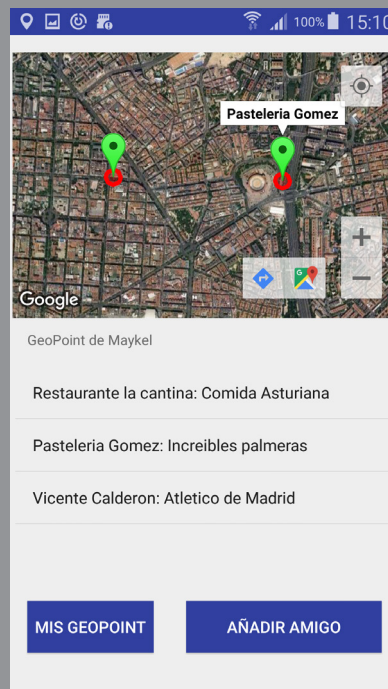
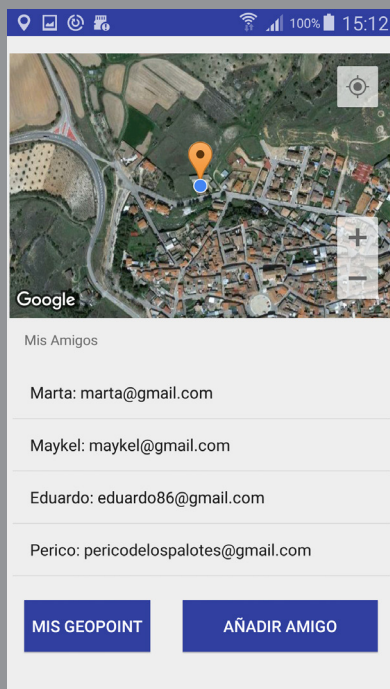
La activity de Guardado se compone de distintos campos de texto con la información necesaria que se quiere añadir al GeoPoint, la localización de la calle, población, número etc... se la devolvemos con los servicios de Google Advance REST Client, de este modo el usuario solo tendrá que añadirle un título, una descripción y un radio y podrá guardar su GeoPoint.

La activity de modificado se le cargan todos los valores del GeoPoint y se le muestran en las cajas de texto, así el usuario puede visualizar todos los campos y modificar el que desee, luego con darle al botón de MODIFICAR tendría su GeoPoint actualizado.

Activity Amigos:

En la activity principal, cuando se selecciona el botón de ir a mis amigos, se carga un ListView donde se muestran los amigos agregados dando la opción a ver los puntos de cualquiera de ellos.

Una vez seleccionado el amigo que desas ver se cargara un ListView en el que se mostraran los GeoPoint del amigo seleccionado.



Una vez cargado los GeoPoint del amigo, al pulsar sobre cualquiera de ellos se cargara en el mapa la localizacion y un marcador con el titulo, si se deja pulsado el GeoPoint durante un breve tiempo, al usuario le saldra un aviso de que si desea agregar ese GeoPoint a su lista.

Si se acepta el guardado, la aplicacion le mostrara un aviso de que se ha añadido correctamente, en caso contrario tambien se le notificara que no se pudo agregar el GeoPoint seleccionado.

Tambien en la activity de amigos hay un boton que es para agregar amigos, este boton te muestra un dialogo en el que introduces el email de tu amigo y se te añadira a la lista siempre y cuando tu amigo este registrado y el tambien te añada a ti como amigo.

En la lista de usuarios amigos, si dejamos pulsado durante un breve tiempo al usuario se le mostrara un dialogo que le avisara de que si desea borrar a su amigo y de que esta accion no podra deshacerse.

PRESUPUESTO

En un principio el coste total de la aplicación será sufragado por los desarrolladores con el objetivo de que el usuario tenga una licencia gratuita y disfrute de GoePoint sin tener que abonar ninguna cantidad inicial. y en caso de que la aplicación sea descargada y se necesiten introducir mejoras se buscará la financiación de terceros, registrando

establecimientos y locales como puntos de interés generales, además de áreas de servicios como gasolineras, parques y otros puntos.

El coste total previsto del desarrollo de la aplicación GeoPoint ha sido:

- > Analista: $22\text{€/hora} \times 17 = 374\text{€}$
- > Programador: $17\text{€/hora} \times 55 = 935\text{€}$
- > Diseñador: $10\text{€/hora} \times 25 = 250\text{€}$
- > Coste total previsto: 1559€

El coste total real del desarrollo de la aplicación GeoPoint ha sido:

- > Analista: $25\text{€/hora} \times 20 = 500\text{€}$
- > Programador: $20\text{€/hora} \times 60 = 1200\text{€}$
- > Diseñador: $10\text{€/hora} \times 30 = 300\text{€}$
- > Coste total real: 2000€

FASE DE MANTENIMIENTO

POSIBLES MEJORAS

El equipo de Desarrolladores de GeoPoint tiene en mente varias mejoras disponibles siempre y cuando se aumenten los recursos y la inversión en tiempo y dinero sea mayo, además algunas propuestas tienen como propósito generar dinero para financiar un buen futuro a GeoPoint.

Alguna de las mejoras son:

Valoraciones de los GeoPoints de tus amigos.

Implementación para recoger los amigos que tienes agregados en Facebook.

Confirmación de registro en el email.

Posibilidad para crear rutas y agregar a usuarios a ellas con una clave de registro para la ruta.

Creación de categorías y subcategorías para los diferentes GeoPoint.

Sugerencias de gasolineras, talleres, lavadero de vehículos, centros comerciales, peluquerías o paradas de autobuses o taxis.

PIDs de Ocio: Llegar a acuerdos con hosteleros y dueños de negocios para promocionar su comida o restaurante.



GEOPOINT

Memoria Proyecto

CONCLUSIONES

Para finalizar quiero decir que GeoPoint tiene un uso muy versatil y de diferentes aplicacion, es una herramienta de mucha ayuda y de pocos recursos del dispositivo y la gran ventaja es que tienes todo guardado en tu cuenta y podras acceder desde cualquier dispositivo en cualquier sitio y asi siempre tener tus GeoPoint disponibles y al alcance de tu mano y de la de tus usuarios amigos que podran ver tus puntos y viceversa.

