UNIVERSIDAD NACIONAL DEL CENTRO DE LA PROVINCIA DE BUENOS AIRES

FACULTAD DE CIENCIAS EXACTAS
INGENIERÍA DE SISTEMAS

Introducción a la Programación de Dispositivos Móviles

Aplicación de seguimiento de actividades físicas

Alumnos: BONGIORNO, Emilio LU: 247666
Email: emiliobongiornov@gmail.com

SANTAMARINA, Esteban LU: 247762
Email: esteban.santamarina@hotmail.com

Docentes: Dr. Alejandro Zunino
Dr. Juan Manuel Rodríguez

Introducción

Este informe explica el desarrollo del trabajo final propuesto por la cátedra de Introducción a la Programación de Dispositivos Móviles correspondiente a la carrera de Ingeniería de Sistemas de la Universidad Nacional del Centro de la Provincia de Buenos Aires (UNICEN). El mismo consiste en una aplicación para el sistema operativo Android llamada “Coyote Run”, que permite mantener un registro de las actividades físicas realizadas (caminar, correr y andar en bicicleta) mediante la utilización del GPS, brindando al usuario información sobre los recorridos (tiempo, distancia recorrida, velocidad máxima, velocidad promedio) e ilustrando los mismos sobre un mapa.
Se destacan los aspectos técnicos sobre la implementación de la aplicación y la usabilidad de la misma, así como también se detallan los problemas encontrados durante el desarrollo, justificando cada una de las decisiones tomadas.

Desarrollo

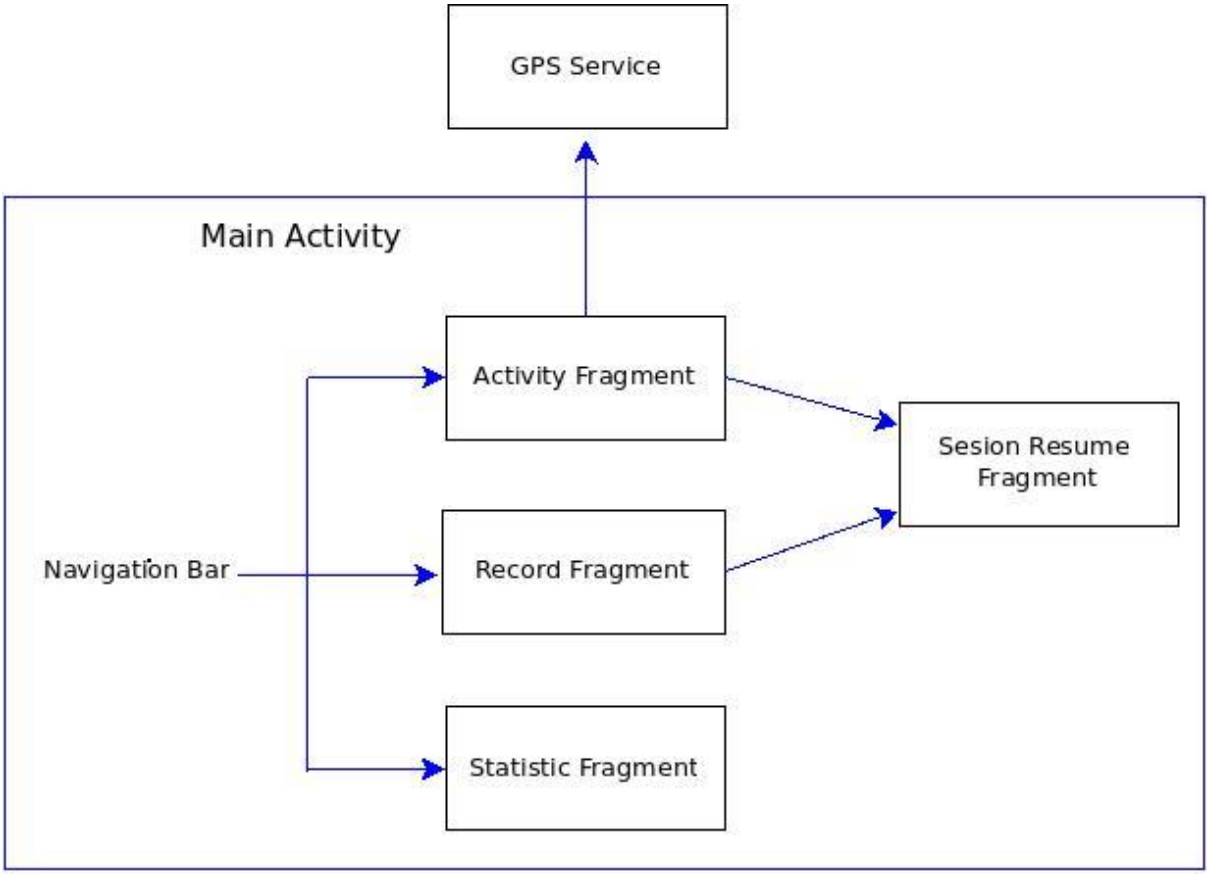
La aplicación fue desarrollada utilizando el entorno Eclipse con el SDK de Android. A continuación se detallarán los diferentes aspectos tenidos en cuenta para su desarrollo.

Diseño

Se cuenta con una barra de navegación que permite acceder a tres interfaces de la aplicación implementadas como “Fragment”. Estas consisten en lo siguiente:

- Actividad actual: permite iniciar una nueva sesión de seguimiento de la actividad, y pausar o parar la misma. Además muestra el tiempo transcurrido y la distancia recorrida durante la sesión actual.
- Registro de actividades: muestra una lista con todas las actividades realizadas con la aplicación, permitiendo acceder a los detalles de cada una de ellas.
- Estadísticas: muestra la información histórica registrada con la aplicación para cada tipo de actividad.

Tanto cuando se finaliza una sesión, como cuando se accede a una sesión ya realizada dentro del registro de actividades, se muestra un nuevo Fragment con los detalles de la actividad, permitiendo luego volver atrás en la navegación.
A su vez se cuenta con un servicio que corre en background cuando se está corriendo una sesión, que será analizado con mayor detalle más adelante.



Componentes de la aplicación (Figura 1)

Se puede observar los diferentes componentes de la aplicación y la navegación dentro de la misma en la imágen “Componentes de la aplicación” (Figura 1).
Nótese que la aplicación posee una única Activity encargada de administrar las diferentes interfaces, que es declarada en el manifiesto de la aplicación.
El servicio sólo puede ser iniciado y detenido desde el Fragment encargado de administrar la sesión actual (FragmentActivity). Cuando la aplicación se cierra y está corriendo una sesión (ya sea que esta se encuentre pausada o no), se muestra una notificación en la barra del dispositivo, indicando al usuario que todavía está registrando una actividad y le permite ingresar a la misma de forma directa.

Persistencia

Se implementó una clase “Sesión” en la que se almacena toda la información sobre una actividad cuando es finalizada: tipo de actividad, tiempo de inicio, duración, velocidad promedio, velocidad máxima, distancia y coordenadas recorridas.
Estas sesiones son agregadas a un vector, que se debe mantener de forma persistente para poder acceder a las sesiones registradas en cualquier momento, sin que se borren al terminar la ejecución o apagar el dispositivo móvil.
Para ello se optó por utilizar la interfaz “SharedPreferences” de Android, que permite guardar y obtener datos mediante el uso de una clave.

Hay que destacar que con la interfaz mencionada sólo es posible guardar datos de tipo primitivos en Java, por lo que fue necesario importar una librería llamada Gson, que brinda la posibilidad de convertir objetos de Java en representaciones JSON (String) y viceversa. De esta forma, mediante el uso de la clave “sesions” en la interfaz SharedPreferences, es posible obtener un String persistente con, por ejemplo, la representación JSON del vector, y con el método “fromJson” provisto por la librería obtener el vector en sí. De la misma manera, y luego de haber agregado la sesión, es posible almacenar la representación JSON del vector modificado tras haber hecho uso del método “toJson” que brinda la librería.
También se utilizó la interfaz SharedPreferences para guardar datos que persistan mientras corre una sesión. Por ejemplo, fue necesario almacenar el tiempo en que se inició una actividad y la distancia recorrida en cada tramo, para poder restablecer el cronómetro y mostrar la distancia que se realizó hasta el momento respectivamente en caso que se cierre la aplicación.

Servicio en Background

Con la necesidad de que la aplicación continúe registrando los eventos a pesar de que la interfaz gráfica no se encuentre activa, fue necesario implementar un servicio que se llamó “GPSService” y extiende de la clase “Service”. Al igual que con la Activity principal, fue necesario declarar esta clase en el manifiesto, en este caso con la etiqueta “service”, y agregar el permiso de “ACCESS_FINE_LOCATION” para el uso del GPS.
Cuando se inicia o continúa una sesión, se invoca al método “startService” de la actividad principal y se le pasa como parámetro un Intent con la clase del servicio. En este último se redefinió el método “onStartCommand” retornando la constante “START_STICKY”, permitiendo de esta forma mantener al servicio corriendo hasta que sea detenido explícitamente por la aplicación a partir de los botones de “Stop” o “Pause”.
Dentro del servicio se incluyó una instancia de la clase “LocationManager” capaz de detectar constantemente cuando se obtiene una nueva ubicación mediante el método “requestLocationUpdates”. Este método es invocado dentro de la redefinición del ya mencionado “onStartCommand” y recibe una instancia de “LocationListener” como parámetro, la cual implementa un método llamado “onLocationChanged” que es ejecutado cada vez que se actualiza la ubicación. En la implementación de este último, se registran todos los datos relevantes a la ubicación, como son la distancia recorrida, las coordenadas y la velocidad.
Cuando una sesión es pausada o finalizada, se invoca desde la actividad principal al método “stopService” con el Intent que había iniciado el servicio en un principio. De esta manera, cuando el usuario presiona los botones de pausar o parar, se deja de registrar la ubicación mediante el GPS.

Mapa

Con el fin de poder visualizar los recorridos realizados sobre un mapa, se utilizó la versión 2 de la API de Google Maps. Esta librería cuenta con una serie de funciones que facilitaron la tarea, como la inclusión de marcadores o el trazado de una línea entre dos puntos.
Para hacer uso de la librería en la aplicación se requiere obtener la clave de la API y agregarla al manifiesto. En consecuencia, fue necesario generar un certificado en codificación SHA1 y utilizarlo junto con el nombre de paquete para registrar la aplicación en la consola de Google Maps y obtener la clave necesaria.
Dentro del manifiesto, se debieron declarar los metadatos con los valores de la clave mencionada, y de la versión de Google Play Services (requerida para usar la versión 2 de la API de Google Maps). También se debió especificar el uso de la librería “com.google.android.maps”.
Como se mencionó anteriormente, cada sesión posee una lista con todas las ubicaciones que registró el servicio. Estas son utilizadas para dibujar el recorrido sobre el mapa a través de la clase “Polylines” de la API. Además, se agregaron marcadores tanto para el inicio como fin del recorrido, representados por la primer y última ubicación de la lista respectivamente.
Por otro lado se tuvo que especificar el zoom aplicado sobre el mapa y ajustar los límites del mismo, por lo cual fue necesario identificar las ubicaciones mínimas y máximas con respecto a su latitud y longitud dentro de la lista. De esta forma, el recorrido generado se visualiza de forma completa en la imágen inicial, a pesar de que luego el usuario puede moverse a través del mapa o bien, modificar el zoom a gusto.

Información durante la sesión

Con el fin de desarrollar una aplicación más interactiva con el usuario, se decidió que sea posible visualizar el tiempo transcurrido y la distancia recorrida mientras corre una sesión. Esto es posible a través del “Activity Fragment”.
El tiempo de la actividad es administrado por un cronómetro que fue incluido a partir de la clase “Chronometer” de Android, siendo capaz de iniciar, pausar o reanudar su conteo según los botones de la interfaz.
Es importante aclarar que este cronómetro no corre en background cuando se cierra la aplicación, sino que se almacena el instante en el que se dejó de visualizar y cuando es reanudado se calcula el valor actual en base a la hora actual del sistema y la previamente guardada.
Los problemas surgidos con esta implementación están ligados con el cálculo del valor del cronómetro cuando la sesión se encontraba activa y se cambiaba de pantalla o se cerraba la aplicación. Fue necesario discernir cuando la sesión estaba activa pero pausada, ya que si luego se cerraba la aplicación, no bastaba con calcular el intervalo de tiempo desde que se cerró, sino también que se precisa saber el tiempo transcurrido desde que se pausó. Por ello, surgió la necesidad de

almacenar otro valor con el intervalo de tiempo entre el cual se pausó la sesión y se cerró la interfaz, para poder considerar no sólo el tiempo desde que se dejó de visualizar el cronómetro, sino también el que transcurrió antes de que esto ocurra y haya estado pausada.
Para mostrar la distancia realizada mientras se corre una sesión fue necesario comunicar el “Activity Fragment” con el servicio de GPS, de forma que este último notifique la nueva distancia recorrida cada vez que es actualizado.
Cada vez que cambia la locación del GPS en el servicio, se envía un “Intent” con la distancia recorrida mediante la clase “LocalBroadcastManager” en broadcast, con el filtro “distance-change”. Esto permite instanciar una clase “BroadcastReceiver” en el Fragment que cumpla la función de receptor, y sobreescribir el método “onReceiver” para que pueda obtener el valor y actualizar al interfaz cada vez que se produzca un cambio. Cabe destacar que para recibir el Intent enviado mediante broadcast, fue necesario registrar el receptor con el filtro “distance-change” en el Fragment mediante el método “registerReceiver” cuando el Fragment es visible (dentro del método onResume), y anular el registro mediante el método “unregisterReceiver” cuando ya no lo es (dentro del método onPause).

Otros detalles de implementación

La aplicación fue desarrollada en el idioma inglés. Sin embargo, y con el fin de obtener un código más flexible, todos los textos fueron declarados en el archivo “strings.xml” dentro de la carpeta “values” e incluidos en el código a partir de este archivo. De esta forma, si se quisiera por ejemplo traducir la aplicación al idioma español, sólo será necesario crear otra carpeta de “values” con el sufijo correspondiente y un archivo XML igual al mencionado pero con los textos en español. En este caso, se utilizará por defecto el idioma inglés, y sólo el español en los dispositivos que posean dicho idioma como predeterminado.

Cuando se invoca al método “requestLocationUpdates” de “LocationManager”, además de la instancia de “LocationListener” y el proveedor de las locaciones (en este caso el GPS), se deben pasar dos parámetros adicionales que indican el intervalo de tiempo y la distancia desplazada mínima para que se actualice la locación. Luego de varias pruebas realizadas con diferentes valores, se llegó a la conclusión que actualizando la locación cada 3 segundos y 25 metros se obtienen los resultados esperados, disminuyendo considerablemente el consumo de batería. Cabe destacar que para que el cambio de ubicación tenga efecto, se deben cumplir de forma conjunta ambas condiciones mencionadas anteriormente.

Conclusiones

Problemas encontrados

Durante el desarrollo del trabajo surgieron una gran cantidad de problemas relacionados con errores en la aplicación que provocaron el cierre de la misma, resultados no esperados, disposiciones de los elementos en la pantalla, entre otros. De todas formas, una de las grandes ventajas que se encontraron a la hora de desarrollar aplicaciones para Android es la inmensa comunidad activa que existe en internet, donde se ofrece gran variedad de soluciones para los distintos problemas. A su vez, la plataforma brinda una extensa documentación donde se explica detalladamente cada componente. Estas características permitieron encontrar soluciones a todos los problemas que surgieron durante la implementación.

Los errores que provocaron el cierre de la aplicación estuvieron mayormente ligados a circunstancias donde se navegaba de un Fragment a otro o se cerraba y volvía a abrir la aplicación mientras se corría una sesión. Para resolver estos conflictos, fue necesario analizar cuidadosamente cómo funciona el ciclo de vida de una Activity y de un Fragment. En la actividad principal se tuvieron que redefinir métodos como “onStop”, “onResume” y “onBackPressed”, que facilitaron manipular los datos cuando se producen dichos eventos. Esto permitió lograr un mayor entendimiento en el funcionamiento general de una aplicación de Android, y cuestiones relacionadas con los posibles estados en los que se puede encontrar una Activity.

Otro inconveniente que se presentó fue el de establecer una interfaz gráfica que se adapte a los diferentes dispositivos con distintos tamaños de pantalla y resoluciones. Actualmente existe una gran diversidad de celulares y tablets con Android que poseen características muy diferentes. Esto exige al desarrollador considerar todas las alternativas para que su aplicación se adapte a todas ellas. Solucionar estos problemas implica crear diferentes disposiciones para los tipos de resolución existentes y utilizar imágenes de distintos tamaños. La aplicación desarrollada fue probada en dispositivos con tamaño de pantalla pequeño y mediano, pero no se diseñaron disposiciones de los elementos para tablets, ya que se concluyó que la misma será utilizada mayoritariamente en dispositivos pequeños. También se debe considerar la disposición de los elementos cuando el dispositivo se encuentra en modo vertical (“Portrait”) o en modo horizontal (“Landscape”). En el trabajo se definió la actividad principal en el manifiesto con la orientación en modo “Portrait”, por lo que no se tuvo que lidiar con las disposiciones de los elementos cuando se gira el dispositivo horizontalmente ya que sólo se puede utilizar de un sólo modo. Si bien se entiende esto puede disgustar a algunos usuarios que deseen usar su dispositivo en el modo “Landscape”, se consideró que por el tipo de aplicación tampoco se requiere añadir dicha funcionalidad.

También surgió la necesidad de probar la aplicación en diferentes versiones de Android. En el manifiesto se declaró como versión mínima necesaria la de Android 2.3 (API 9), lo que implicó importar librerías de soporte y utilizar los métodos que

estas brindan para implementar cuestiones como la barra de actividades o los Fragments.

A medida que se desarrolló la aplicación, esta fue probada en un dispositivo celular Motorola Moto G con Android 4.4. También se ejecutó la misma mediante el emulador que brinda Eclipse, pero no fue posible instalarla referenciando a la versión 2 de la API de Google Maps, impidiendo probarla en las máquinas virtuales.

Valoraciones

El trabajo realizado permitió afianzar los conceptos desarrollados durante la cursada de la materia, así como también ampliar los conocimientos sobre el desarrollo de aplicaciones para dispositivos móviles.
Llevar a la práctica el desarrollo de una aplicación más completa llevó al grupo a tener que investigar acerca de la plataforma de Android y entender su funcionamiento de manera más amplia.
Los contenidos de la materia resultaron útiles para llevar a cabo la aplicación, permitiendo realizar un desarrollo de manera más acertada. A su vez, lo aprendido brinda la posibilidad de llevar a cabo nuevas aplicaciones nativas en Android con mayor facilidad, al tener una base sobre los conceptos importantes que se deben considerar, y experiencia en el área luego de buscar soluciones para los problemas que se fueron presentando.

Referencias

http://developer.android.com/develop/index.html

https://developers.google.com/maps/documentation/android/?hl=es

http://stackoverflow.com/

http://sourceforge.net/projects/json-lib/files/latest/download