Aplicación de tipo Juego llamado “Dime Quien Soy” para celulares con sistema operativo Android.

Jerónimo Fermín Deiros
Universidad Nacional de la Matanza,
Departamento de Ingeniería e Investigaciones Tecnológicas,
Florencio Varela 1903 – San Justo, Argentina.

jdeiros@alumno.unlam.edu.ar

Resumen: El objetivo de este documento es exponer los avances obtenidos en el desarrollo de una aplicación de entretenimiento, que consiste en un juego llamado “Dime quién soy” y consiste en adivinanzas para jugar entre 2 o más personas.

1 Introducción

En el caso de los seres humanos, un juego es toda aquella actividad que realiza uno o más personas (llamadas jugadores) que, independientemente de su edad, su profesión/ocupación o su estatus social, emplean su imaginación o herramientas para crear una situación con un número determinado de reglas, con el fin de obtener o proporcionar entretenimiento y diversión. Existen juegos competitivos, donde los jugadores tienen que lograr un objetivo, y juegos no competitivos, donde los jugadores buscan simplemente disfrutar del entretenimiento de la actividad y diversión. [1]
En este caso, se trata de un juego competitivo, que a su vez podrá ser colaborativo si se juega entre mas de 2 personas. No hay edad limite para jugar a “Dime quien Soy”, solo se requiere saber leer y tener la posibilidad de sostener el dispositivo y realizar movimientos con él, de manera que el o los oponentes puedan ver la información en pantalla, y la persona que sostiene el telefono no, para que pueda adivinar que ven los demás.

2 Desarrollo

2.1 Datos del Alumno

Nombre y Apellido: Jerónimo Fermín Deiros
DNI: 28095186
Comisión: 01-3900
Horario de cursada: miércoles (19 – 23) hs.

2.2 Breve manual de usuario

En principio, es necesario contar con conexión a internet en el dispositivo. Cuando el juego se inicia, la pantalla principal da la posibilidad de realizar el login, si el usuario ya tiene cuenta registrada, o crear una cuenta para un usuario nuevo.
Una vez ingresado el usuario y contraseña, el sistema valida dicha información o crea el usuario nuevo según la opción elegida y como siguiente paso, se mostrará la pantalla de opciones para jugar o cerrar sesión, donde si informará también el estado actual de la carga de la batería. Una vez logueado el usuario, el sistema comienza un proceso por detrás que permite mantener el token actualizado, ya que el Loguin se realiza contra un api desarrollada por la catedra de Sistemas Operativos Avanzados y esta especifica una duración del token de 30 minutos. El propósito del proceso es el de mantener la sesión activa renovando dicho token cada 25 minutos, hasta que el usuario cliquee el botón de cerrar sesión en la pantalla de opciones anteriormente mencionada. Entonces, una vez se presiona el botón “Jugar”, se cargarán una serie de personajes previamente cargados que se irán mostrando secuencialmente uno tras otro.
Se dispone de 1 minuto, 30 segundos para adivinar cada uno de los personajes que vayan apareciendo en la pantalla. Si bien se puede jugar de a dos personas, lo ideal es jugar en equipos y el juego se desarrolla como se muestra a continuación.
Como primer paso, ver **FIGURA 1**, el jugador que adivina se coloca el telefono en la frente con la pantalla horizontal, visible a los demás jugadores, pero no para sí mismo. Los jugadores de su mismo equipo pueden imitar, dar consejos, bailar, dibujar, cantar, lo que se les ocurra para dar pistas y ayudar a que el jugador que sostiene el telefono adivine el personaje que están viendo en pantalla. Si éste adivina la palabra, inclina el telefono hacia arriba (con la pantalla hacia arriba) como se muestra en la **FIGURA 2**, esto sumará un punto para el equipo, pero el tiempo sigue corriendo, por eso, si los jugadores deciden pasar al siguiente personaje sin adivinar, se debe inclinar el celular hacia abajo, como se ve en la **FIGURA 3**. Con este movimiento un nuevo personaje aparecerá en pantalla para ser adivinado, pero no se suma ningún punto, tampoco se pierde puntos al pasar de esta forma. Si en medio de la partida, por algún motivo es necesario pausar el juego, el jugador que esta adivinando puede tapar la pantalla como en la **FIGURA 4**, y así el juego quedará pausado, los sensores no detectaran movimiento y el tiempo dejará de correr hasta que se pulse el botón “continuar” que aparecerá en pantalla.


FIGURA 1


FIGURA 2


FIGURA 3


FIGURA 4

Finalmente, el equipo que adivine la mayor cantidad de personajes será el ganador, el resultado de la cantidad de personajes adivinados se muestra en la pantalla final, al llegar el contador de tiempo a cero.

2.3 Dirección web del repositorio GitHub

Repositorio: https://github.com/jdeiros/DimeQuienSoy

2.4 Diagrama funcional / navegación de Activities.

A continuación, en la **FIGURA 5**, se muestra el diagrama de Activities de la aplicación.

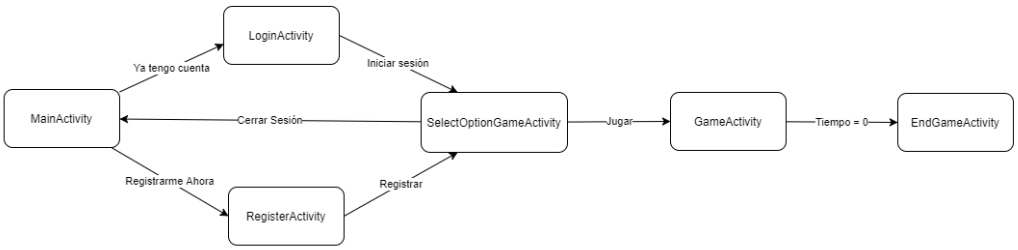

FIGURA 5

En el boceto mostrado en la **FIGURA 6**, puede verse de manera grafica las pantallas que involucra cada Activity.

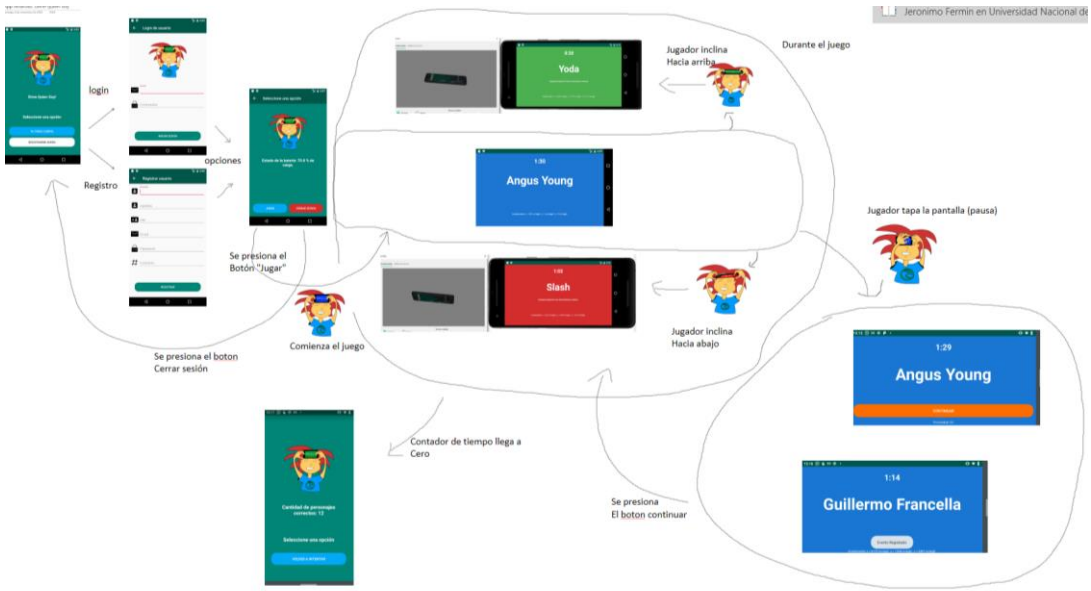

FIGURA 6

Justificación

Para mejorar el aspecto visual de las pantallas se utilizó Material Design [2] que es una normativa de diseño enfocado en la visualización del sistema operativo Android, además en la web y en cualquier plataforma desarrollado por Google [3].

El mecanismo de navegación entre pantallas utilizado en general es a través de Intents [4], instanciando un objeto Intent, pasando por parámetro la Activity actual y la de destino y llamando al método startActivity() que recibe como argumento el Intent creado. En las Activities LoginActivity, RegisterActivity y SelectOptionGameActivity también se utilizó una

Toolbar, que es un widget que permite mostrar un botón de regreso a la Activity padre definida en el tag de la Activity actual dentro de AndroidManifes.xml declarando así la actividad superior y permitiendo agregar la acción “up” o “hacia arriba” [5].

2.5 Sincronización de la ejecución concurrente del programa.

Se utilizó el mecanismo de sincronización de hilos a través del modificador “synchronized” visto en el apunte de la catedra [6].
La aplicación tiene concurrencia en la Activity del juego ActivityGame, dado que esta se suscribe a la escucha de los sensores de proximidad y de acelerómetro. También fue necesario sincronizar el acceso a los datos guardados por SharedPreference [7] para mantener la actualización del estado de login.

Justificación
Por un lado, el acceso concurrente que podríamos tener en los sensores para asegurarnos en caso de que el acceso sea simultaneo.
Por otro lado, para mantener el token actualizado, hemos creado un hilo implementando la interfaz Runnable el cual realiza una petición para refrescar el token al api desarrollada por la catedra cada 25 minutos, asegurando que el usuario no pierda la sesion. Este hilo actualiza el token actual y el token de refresco, como también la situación de usuario logueado. Como estos datos son accedidos por otros procesos, por ejemplo, para declarar el cierre de sesion, en la Activity SelectOptionGameActivity, se pasa el flag “userAlreadyLoggedIn” a false, para que el hilo de refresco de token detecte la intención de cerrar sesion y termine el ciclo, finalizando el thread. La razón por la cuál se utilizó este mecanismo de sincronización, es para asegurar que este dato guardado en sharedPreferences sea accedido para lectura o escritura de manera segura.

2.6 Comunicación entre componentes.

Como se mencionó en el apartado anterior 2.5, La comunicación entre el hilo de refresco de token y el activity que declara el cierre de sesión se realizó a través de un dato guardado en SharedPreferences.
Otro mecanismo utilizado entre componentes fue el de Intents, utilizando el metodo putExtra() – getExtras() [8] Por ejemplo entre la activity GameActivity y EndGameActivity se pasa un json con los datos de todos los personajes utilizados durante el juego para que en la pantalla final se pueda hacer el recuento de los personajes adivinados y mostrar el resultado final.

Justificación
El uso de SharedPreferences es para lograr guardar el dato (usuario logueado, o usuario sin loguear y el token de refresco) y que esto me permita recuperarlo independientemente de que la aplicación este cerrada o se vuelva a abrir. Se busco la forma de poder pasar por alto la pantalla de login si es que el usuario no necesita loguearse y este es el motivo por el cual se implementó ese mecanismo.
El uso de intent para el traspaso del array de personajes dentro del juego se utilizó con putExtra() porque consideramos que es un dato que no necesita ser persistido, es una variable que permitirá a la pantalla final hacer el recuento.

2.6 Comunicación con el servidor.

La comunicación con el servidor se usa para el registro de usuario, para el login de usuario, para la renovación automática del token y también para registrar eventos, registramos el evento “continue” que sucede al presionar el botón con el mismo nombre que aparece al pausar el juego cuando se detecta la mayor proximidad (cuando el usuario tapa la pantalla para pausar el juego). Se muestra en la **FIGURA 6** y **FIGURA 7**.

FIGURA 7

FIGURA 8

Se utilizó Retrofit. Para esto, primero se agregaron las dependencias en el archivo build gradle para retrofit y para gson que permite la transformación de las peticiones en objetos de java, también se agregaron los permisos de acceso a internet en el archivo AndroidManifiest. Luego, se crearon los objetos de transferencia para las peticiones de los registros de eventos, de login o registro de usuario y de token y token-refresh, que en este caso se compartieron en algunos casos con los de usuario ya que tenían datos similares en la respuesta (env, token, refresh-token, success). Estos son usados para preparar la petición antes de enviarlos al sevidor y para construir la respuesta del servidor, así tenemos por ejemplo los objetos EventRequest, EventResponse para los eventos y UserRequest, UserResponse para los usuarios y tokens que seran mapeados por retrofit automáticamente cuando recibe los datos. Luego se crearon las interfaces necesarias para cada servicio a configurar: registro de usuario, login de usuario, registro de evento y token-refresh.
Para hacer las peticiones se construye primero un objeto Retrofit que se configura con la url base y el mecanismo que vamos a usar para trabajar el objeto, en este caso gson. Luego creamos el servicio de retrofit basado en el servicio declarado en la interfaz. Con esta interfaz retrofit es capaz de crear el objeto de tipo Call para encolar las peticiones que necesitamos hacer. En esta aplicación se hizo un archivo de interfaz por cada servicio, si bien podría hacerse todo en el mismo archivo se tomo la decisión de separarlo para mayor legibilidad y entendimiento.
Entonces, con el objeto Call retrofit encola las peticiones y las sincroniza con el hilo principal. Lo hace de manera asincronica, pero dentro del hilo principal y envía las peticiones según tenga los recursos disponibles. Para encolar estas peticiones, se usa el metodo enqueue() del objeto Call pasando por parámetro el Callback del servicio requerido, así se recibe la respuesta en el método onResponse (respuesta del servidor 200->ok, 400->bad request, etc.) u onFailure (fallo la petición, no logro hacer la petición) según si fue exitosa o erronea.

2.6 Problemas durante el desarrollo.

Problemas no técnicos, relacionados con el desarrollo

En principio, el mayor problema con el que me encontré fue el de encarar un proyecto demasiado ambicioso que demandaría un tiempo de desarrollo y aprendizaje que excedía a lo requerido por la catedra. Quizás debido a la falta de experiencia con Android, me costó relacionar los requerimientos con una idea que se enfoque en los puntos a evaluar. Quise mencionar este inconveniente porque al darme cuenta de esto tomé la decisión de repensar el desarrollo y empezar desde cero con un nuevo proyecto, esto me costo tiempo de desarrollo, que si bien me sirvieron para aprender y poner en practica conceptos que pude aprovechar, me resto tiempo de desarrollo en la idea que finalmente desarrolle.

Problemas técnicos

Al agregar las dependencias de retrofit tuve un error relacionado con la version de java instalada. Lo solucioné siguiendo una respuesta encontrada en un post en el sitio de stackoverflow [9].

Problemas con la request al probar con un modelo de celular en particular: Motorola Moto G6, por ser http en vez de https la URL del servidor. Esto funcionaba bien en la maquina virtual de android studio pero no en el dispositivo físico. Tuve que agregar un permiso en el archivo AndroidManifest para solucionarlo: ACCESS_NETWORK_STATE. encontré la solución en un post de stackoberflow [10].

El mayor problema lo tuve al intentar, en un comienzo, utilizar HttpConnection para realizar las peticiones con un intent service. No logre que funcione el broadcast reciver para comunicarme con la activity principal y creo que es porque la acción que definí en el intent filter no la registre correctamente, entonces el broadcast reciver nunca era llamado o ejecutado. Por esto, no tenia ningún error y no pude encontrar la forma de resolverlo, es decir, no pude detectar exactamente la causa del problema. Esto me tuvo bloqueado por un tiempo considerable. Por eso decidí utilizar Retrofit, con el cual, si bien tuve problemas en un principio, como lo que explico mas arriba relacionado con la versión de java, finalmente logré solucionarlo y resolver las peticiones con esa otra metodología.

Referencias

[1] «Wikipedia,» 2017. [En línea]. Available: https://es.wikipedia.org/wiki/Juego.

[2] M. De, «Material Design,» 2020. [En línea]. Available: https://material.io/.

[3] Wikipedia, «Wikipedia,» 2019. [En línea]. Available: https://es.wikipedia.org/wiki/Material_Design.

[4] D. Android, «Iniciar otra actividad en android,» 2020. [En línea]. Available: https://developer.android.com/training/basics/firstapp/starting-activity?hl=es-419.

[5] a. developer, «Accion hacia arriba,» [En línea]. Available: https://developer.android.com/training/appbar/up-action.

[6] T. y. Sincronización, «Sodium,» 2020. [En línea]. Available: http://so-unlam.com.ar/material-clase/Android/Thread%20%20y%20Sincronizacion.pdf.

[7] d. android, «shared preferences,» 2020. [En línea]. Available: https://developer.android.com/reference/android/content/SharedPreferences.

[8] d. android, «Intent putExtra,» 2020. [En línea]. Available: https://developer.android.com/reference/android/content/Intent#putExtra(java.lang.String,%20android.os.Parcelable).

[9] s. overflow, «compativilidad version java,» 2020. [En línea]. Available: https://stackoverflow.com/questions/62151457/invoke-customs-are-only-supported-starting-with-android-o.

[10] S. Overflow, «access network state,» 2020. [En línea]. Available: https://stackoverflow.com/questions/19642032/whats-the-difference-between-access-network-state-and-internet.