Luciano Pulido
DNI:40137604

Desarrollo de una aplicación que prueba las mediciones del acelerómetro, el sensor de proximidad, la comunicación mediante la API REST y la persistencia de datos con SharedPreferences

Luciano Pulido

Universidad Nacional de La Matanza
Departamento de Ingeniería e Investigaciones Tecnológicas

Introducción:

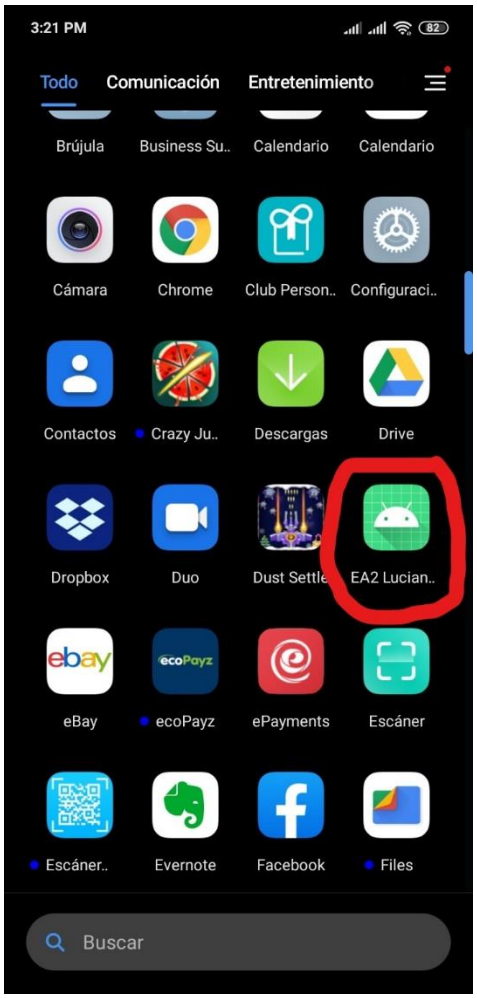
La aplicación construida con el IDE de desarrollo Android Studio utilizando el lenguaje de programación JAVA nos permite registrar un usuario en el servidor la API del servidor de la catedra, iniciar sesión la API del servidor de la catedra y además nos permite probar los sensores acelerómetro y el sensor de proximidad, mostrando en la interfaz gráfica de la aplicación los datos medidos por los sensores. Al ejecutar la aplicación se muestra en la pantalla la interfaz gráfica de la activity correspondiente al Login, la cual tiene un campo para ingresar el email previamente registrado y una contraseña previamente registrada y un botón para “iniciar sesión” y otro para crear una cuenta nueva(registrar un nuevo usuario). Si pulsamos el botón “iniciar sesión” y el email o contraseña es incorrecta se mostrara un mensaje de error donde se le indica claramente al usuario que ha ingresado el email o la contraseña incorrecta y que debe ingresarlos nuevamente de manera correcta. Si pulsamos el botón “iniciar sesión” y el email y la contraseña es correcta se hace la transición a la activity de sensores. Si pulsamos el botón “crear cuenta” se hace la transición a la activity de registrar nuevo usuario, donde tenemos los campos: nombre, apellido, DNI, email, contraseña y comisión y un botón para registrarse. Si pulsamos el botón “registrarse” y el DNI ya existe en la base de datos del servidor de la api de la catedra, mostrara por pantalla un mensaje de error diciendo que el DNI ya está registrado y que ingrese otro DNI o inicie sesión con el email y contraseña correspondiente. Si pulsamos el botón “registrarse” y el email ya existe en la base de datos del servidor de la api de la catedra, mostrara por pantalla un mensaje de error diciendo que el email ya está registrado y que ingrese otro email o inicie sesión con el email y contraseña correspondiente. Si pulsamos el botón “registrarse” y la contraseña tiene una longitud menor a 8 caracteres, mostrara por pantalla un mensaje de error diciendo que la contraseña ingresada debe tener un mínimo de 8 caracteres . Si pulsamos el botón “registrarse” y los campos están llenados correctamente se loguea automáticamente y se hace la transición a la activity de los sensores. Cuando estamos en la activity de sensores podemos registrar eventos que contengan informacion de las mediciones del sensor de proximidad en la API REST del servidor de la catedra acercando la mano al celular a una distancia menor a 5 centimetros y alejándola, luego de esto. Si pulsamos el botón “ver eventos sensor proximidad” se hace la transición a la activity donde se encuentra una lista con los eventos del sensor de proximidad registrados, si es que se registro alguno, y además

donde se guardan la información de estos eventos en un archivo SharedPreferenced de manera persistente, los cuales son cargados en la lista mencionada anteriormente la próxima vez que se abra la aplicación. En esta activity tenemos un botón “ver sensores” que si lo pulsamos nos lleva a la activity de sensores nuevamente. Además tanto en la activity de sensores como en la activity donde se encuentra la lista de eventos registrados del proximidad, al transcurrir 29 minutos desde que se inicio sesión en la activity login o se registro un nuevo usuario desde la activity registro, se solicita la renovación del token y se hace una transición a una activity encargada de la renovación del token, donde se nos pregunta si deseamos seguir usando la aplicación o cerrar la sesión. Si pulsamos el botón “cerrar la sesión” , se hace la transición a la activity login. Si pulsamos el botón “seguir usando la aplicación”, se refresca el token por uno nuevo y luego se hace la transición a la activity desde la cual se hizo la transición a la activity de renovación del token.

Luciano Pulido
DNI:40137604

Manual de usuario:

Lo primero que tenemos que hacer una vez instalada la aplicación en nuestro celular es presionar el icono de la aplicación en nuestro celular.



Una vez pulsado, la aplicación iniciara en la activity (pantalla) correspondiente al Loguin.

Si en la activity del Loguin llenamos los campos del email y contraseña correspondiente a una cuenta previamente creada y luego presionamos el botón “iniciar sesión”, accederemos a la activity donde se mostrara los datos que mide el sensor acelerómetro y el sensor de proximidad:

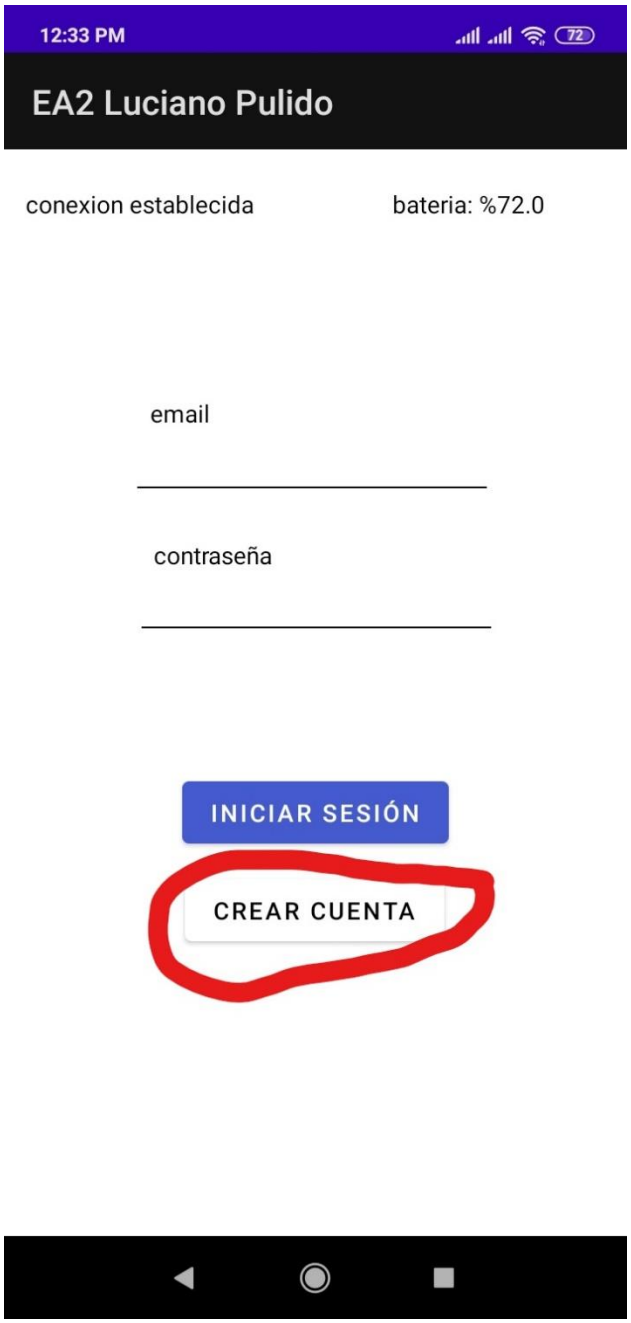
Activity de Login

Activity de Sensores

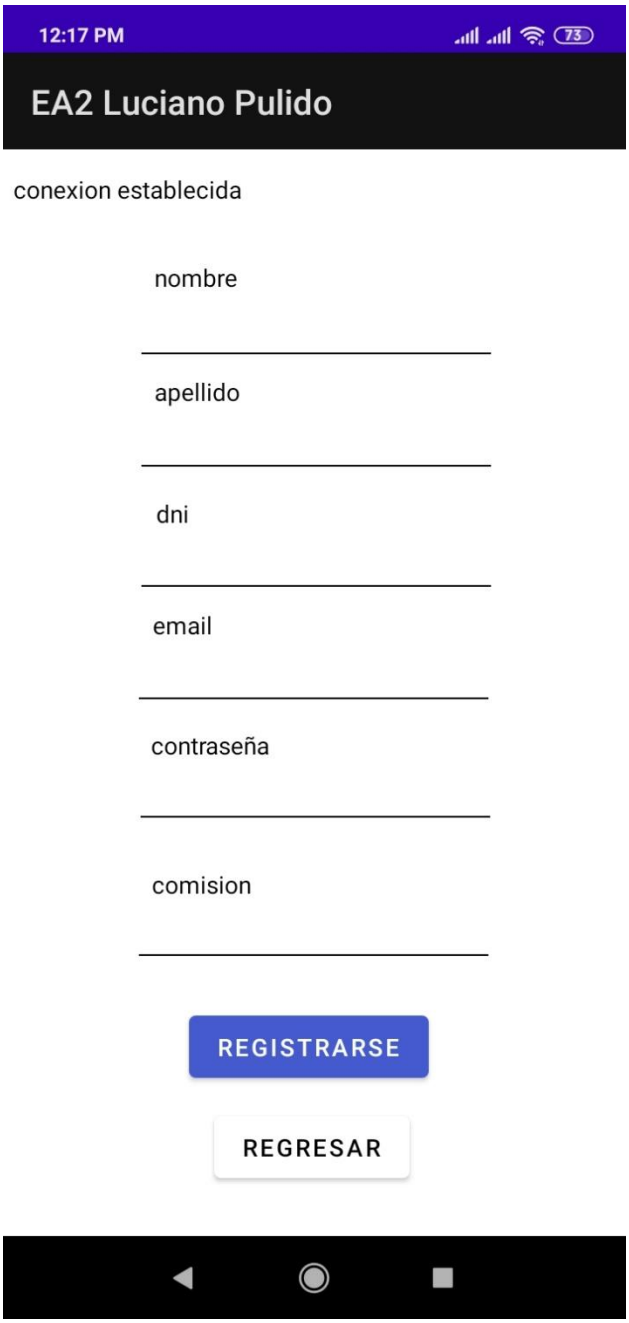
Luciano Pulido
DNI:40137604

Si en la activity del Loguin presionamos el botón “crear cuenta” porque no tenemos una cuenta previamente creada para iniciar sesión, accederemos a la activity de registro de usuarios:

Activity de Login



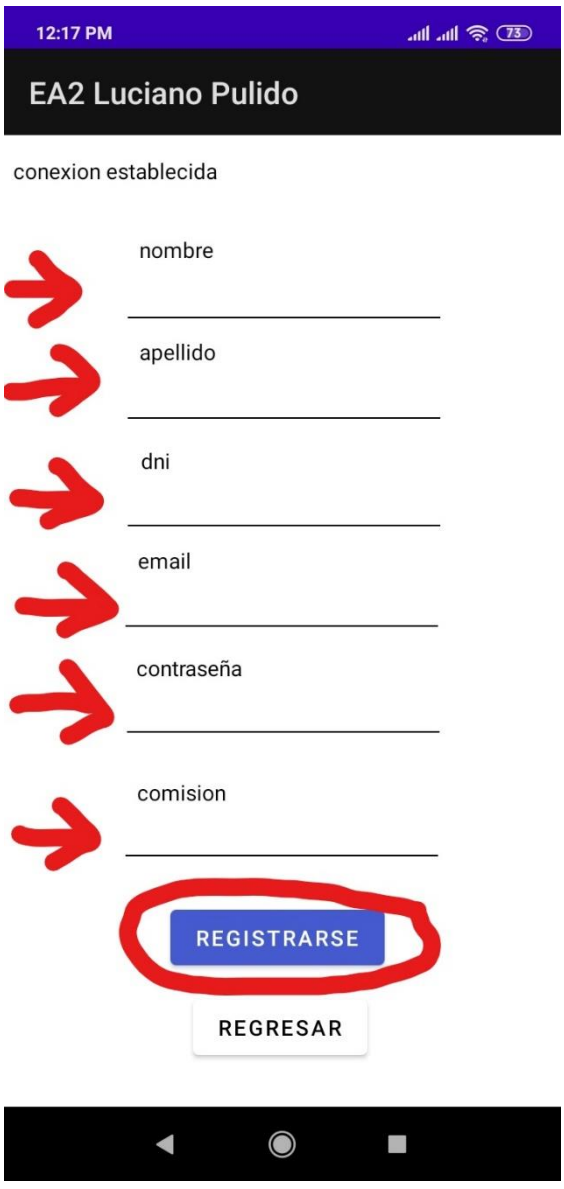
Activity de Sensores



Luciano Pulido
DNI:40137604

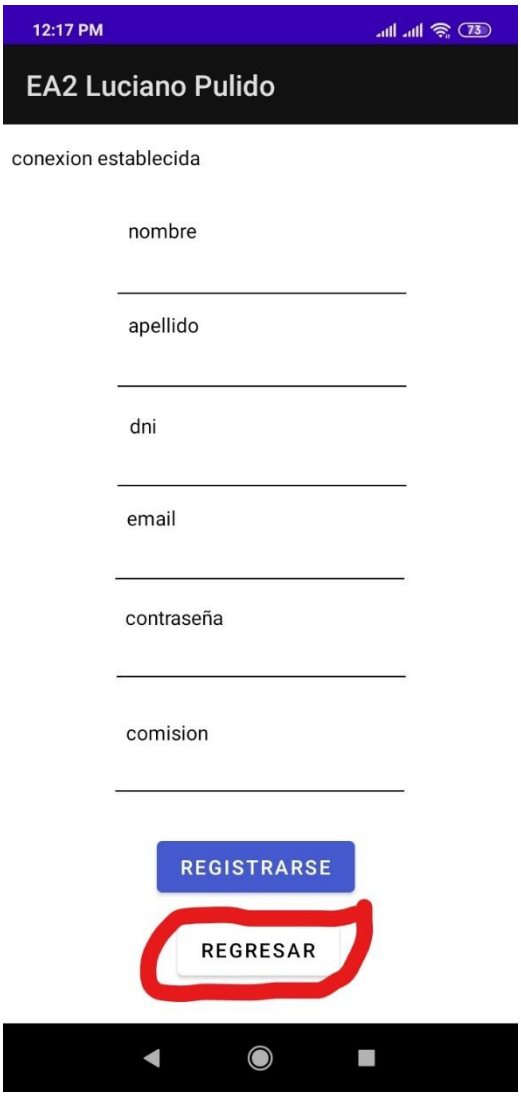
Si en la activity Registro llenamos los campos : nombre, apellido, DNI , email, contraseña y comisión correctamente y presionamos el botón “registrarse”, nos loguea automáticamente y accedemos a la activity de sensores.

Activity de registro → Activity de sensores



Si en la activity registro pulsamos el botón “regresar”, se hace la transición a la acitivity Login

Activity Registro



Activity Login

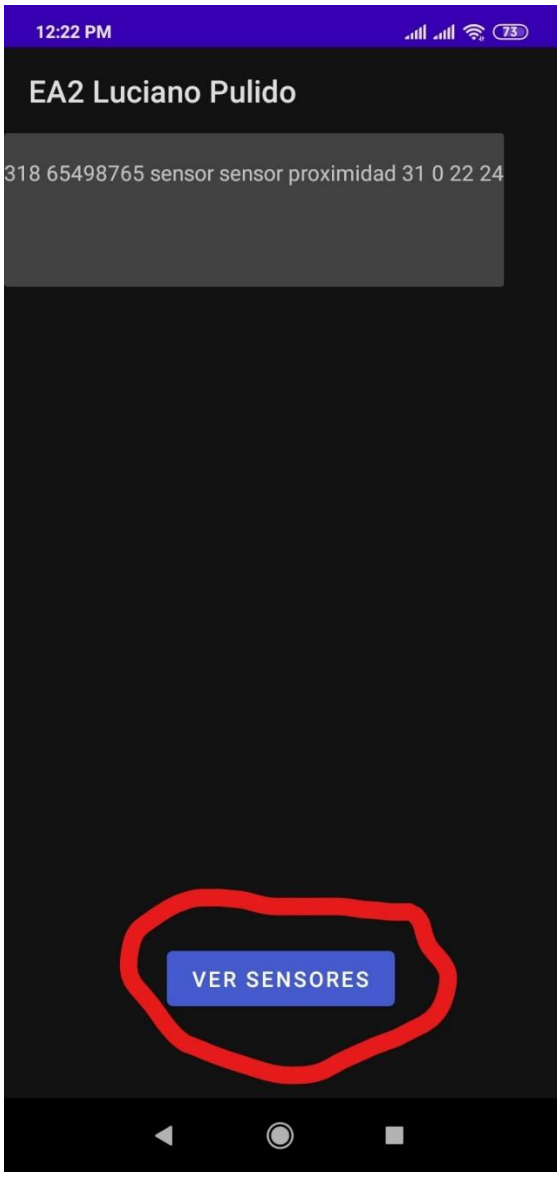
Si estamos en la Activity Sensores, podemos registrar eventos con la información medida por el sensor de proximidad en la API REST del servidor de la catedra, para hacer esto debemos acercar nuestra mano al Smartphone y alejarla (acercar la mano a una distancia menor a 5 centimetros y alejarla para registrar un evento con la informacion del sensor de proximidad). Además si pulsamos el botón “ver eventos sensor proximidad” se hará la transición a la activity “listaEventosSensorProximidad” donde hay una lista (RecyclerView) que contiene la información de los eventos registrados previamente , si es que se registró alguno previamente. Además en esta Activity la información de estos eventos es guardada de manera persistente en un archivo SharedPreferences para que al abrir la aplicación nuevamente recuperemos la información almacenada previamente y se cargue la misma en la lista que allí se encuentra.

Luciano Pulido
DNI:40137604

Activity Sensores → Activity ListaEventosSensorProximidad

Si estamos en la Activity listaEventosSensorProximidad y pulsamos el botón ver sensores, se hace la transición a la Activity Sensores

Activity listaEventosSensorProximidad

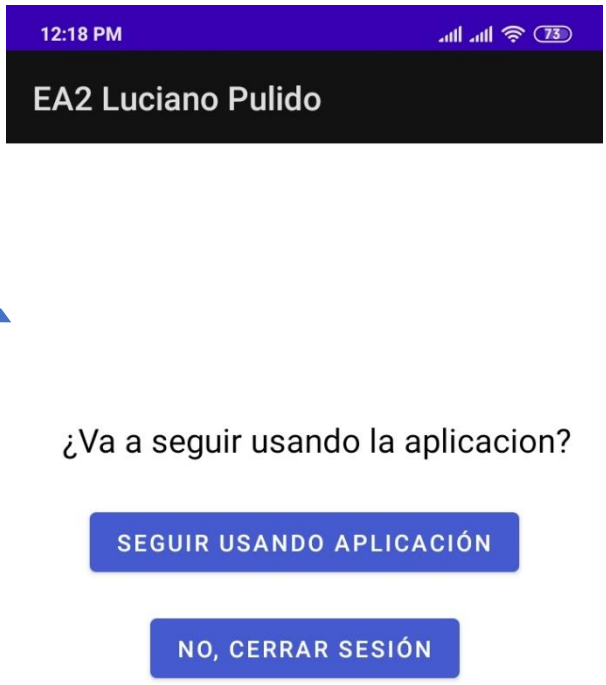


Activity Sensores

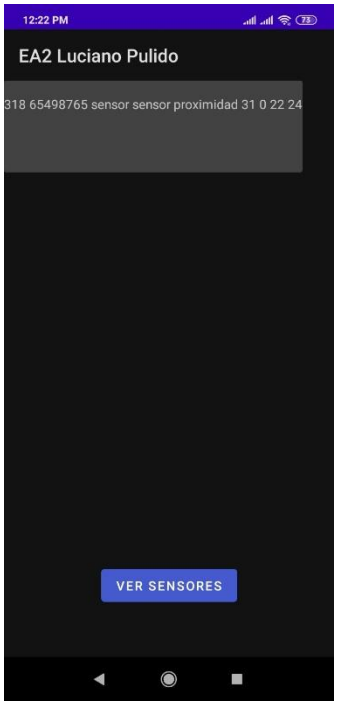
Además una vez que iniciamos sesión o registramos un nuevo usuario y pasaron 29 minutos estando en la Activity Sensores o en la Activity listaEventosSensorProximidad, se hará la transición automática hacia la activity “tokenRefresh”, donde podremos actualizar el token para seguir usando la aplicación o cerrar sesión si lo deseamos.

Activity Sensores

Activity tokenRefresh



Activity listaEventosSensorProximidad



En la Activity tokenRefresh si pulsamos el botón “cerrar sesión”, se hace la transición a la Activity Login .

Activity tokenRefresh

ActivityLogin

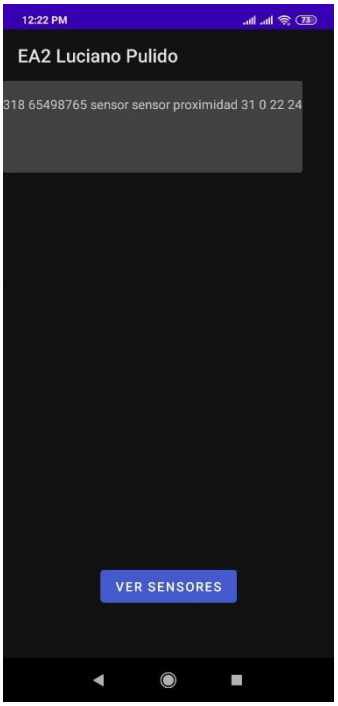
Luciano Pulido
DNI:40137604

En la Activity tokenRefresh si pulsamos el botón “seguir usando aplicación”, se hace una petición a la API REST se actualiza el token para seguir usando la aplicación por otros 29 minutos y luego se hace la transición a la Activity Correspondiente que puede ser la Activity Sensores o la Activity listaEventosSensorProximidad.

Activity tokenRefresh

Activity Sensores

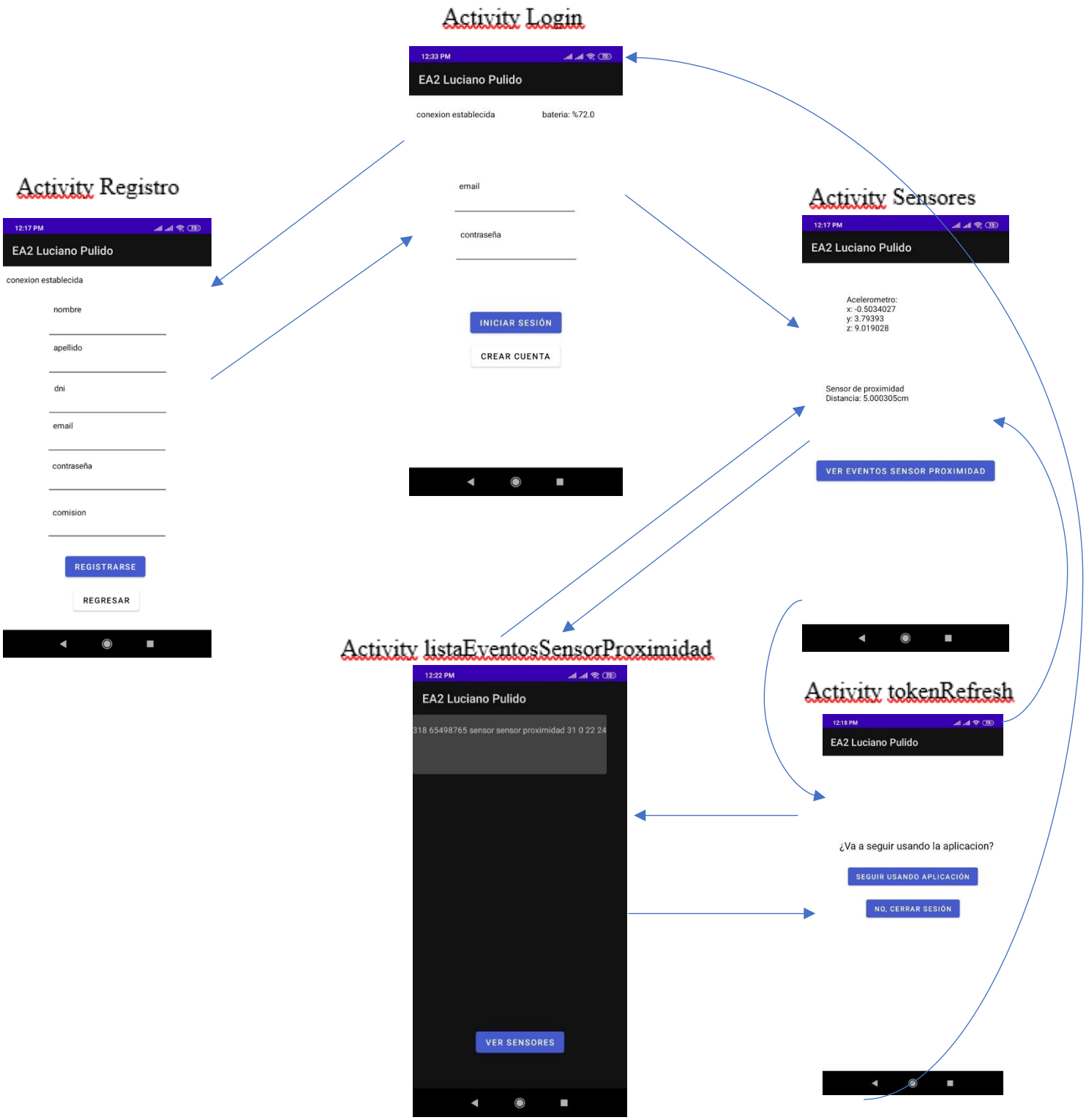
Activity listaEventosSensorProximidad



Repositorio de GitHub:

https://github.com/lucianopulido/EA2-Luciano-Pulido.git

Diagrama Funcional/Navegación activity:



Manera en que se realizó la Sincronización:

En mi aplicación en cada activity utilice el hilo principal para las tareas relacionadas con la interfaz gráfica (es el único hilo que puede modificar la UI) y un hilo secundario en background (Thread) que se encarga de realizar las operaciones bloqueantes y para la sincronización de los mismos en cada activity utilice un handler para sincronizar el hilo principal con el hilo secundario (Thread) para poder así desde este último enviarle datos a el hilo principal para mostrar algún mensaje de error que ayude al usuario a utilizar la aplicación o avisarle al hilo principal que la respuesta de la petición hecha a la API del servidor de la catedra ha sido exitosa y que por lo tanto el hilo principal tome la decisión de hacer la transición a la activity correspondiente. [1]

Manera en que se comunicaron los componentes:

En mi aplicación los componentes que comunique son las activitys utilizando intents para enviar los datos que una requiere de otra como por ejemplo desde la activity Login le envío el token a la activity sensores mediante un intent por el cual le envió un token para que la activity sensor pueda renovar el token si fuera necesario. [1]

Técnica utilizada para la comunicación con el servidor:

La técnica utilizada para la comunicación con el servidor de la catedra es HttpConnection.

Lo que hice fue dentro de un hilo secundario en Background (Thread) crear una instancia de la clase URL y una instancia de la clase HttpURLConnection. Luego abrí una conexión con el método openConnection de la instancia de clase URL para establecer la conexión con el servidor. Luego configure el verbo para la petición correspondiente, por ejemplo para el caso de querer registrar un nuevo usuario configure el verbo con el método setRequestMethod de instancia de la clase HttpURLConnection , pasándole como parámetro el verbo “POST”. Luego configure el header con el método setRequestProperty pasándole como parámetro por ejemplo para el registro de un nuevo usuario “content-type” y “application/json”. Luego para configurar el Body de la petición, empiezo creando una instancia de la clase DataOutputStream y le paso por parámetro el método getOutputStream de la instancia de clase HttpURLConnection para configurarla para enviar datos al servidor. Luego almaceno en la instancia de la clase DataOutputStream datos de la instancia de la clase JSONObject previamente creada, con el método WriteBytes de la instancia de la clase DataOutputStream. Luego envío la consulta con el método connect de la instancia de la clase HttpURLConnection. Luego espero la respuesta por parte del servidor con el método getResponseCode. [1]

Problemas durante el desarrollo:

Los problemas que tuve durante el desarrollo fueron los siguientes:

Problemas:

1. Desconocimiento del entorno de Android Studio.
2. Desconocimiento en el formato de texto Json
3. Desconocimiento en el funcionamiento de la api REST

Soluciones:

1. Investigar el entorno de desarrollo en internet y ver cursos.
2. Investigar cómo se envían datos utilizando el formato de texto Json en internet.
3. Investigar que era la api REST y como enviar peticiones utilizando la misma en internet y viendo cursos.

Motivo por el cual decidí usar Thread (Hilos en background):

Decidí usar hilos en background Thread en lugar de usar asynctask o servicios porque es un mecanismo de ejecución en background que ya conocía y por lo tanto decidí no improvisar ya que perdería tiempo investigando sobre los últimos 2 mecanismos de ejecución en background. Además de lo anteriormente mencionado tuve en cuenta en base a mi investigación en internet, donde encontré información que decía en que momentos es recomendable usar Thread, los cuales son :

- En tareas a largo plazo (operaciones de red que incluyen descargar datos)
- Funciones de alto procesador que deberían funcionar en segundo plano (operaciones bloqueantes como las consultas a la api del servidor de la catedra) [2]

Motivo por el cual decidí usar intent para comunicar activity:

Decidí usar intents como mecanismo de comunicación porque debía para comunicar activitys, por lo tanto decidí usar esto porque en base a mi investigación en internet , la información obtenida de las clases y la bibliografía de la catedra, obtuve como conclusión que cuando se quiere iniciar , comunicar y enviar información a otra activity la manera correcta era usando intents. [1]

Motivo por el cual decidí utilizar HttpConnection:

Decidí usar HttpConnection porque era un mecanismo de comunicación con el servidor muy similar a lo que es la comunicación normal de sockets que conocía de materias previas y por lo tanto me pareció más simple que usar “Retrofit” y además HttpConnection se debe ejecutar en un hilo secundario en Background y por lo tanto no debo tener el cuidado que debo tener si uso “Retrofit” ya que este tiene la posibilidad de que se produzca un bloqueo porque todo se ejecuta en el hilo principal. [1]

Mecanismo de persistencia de los datos en la aplicación:

En mi aplicación se guardan los datos de los eventos del sensor de proximidad de manera persistente utilizando un archivo SharedPreferences, el cual es usado tanto para escribir los eventos antes mencionados cuando pasamos por el método onPause de la activity llamada “ActivityListaEventosSensorProximidad y además cuando cerremos la aplicación y volvamos a abrirla, y accedamos al metodo onStarted de la activity mencionada anteriormente, se cargan en la RecyclerView que se encuentra allí, los eventos que fueron grabados previamente en el SharedPreferences. Técnicamente lo explicado anteriormente seria así:

- Cuando tengo que escribir la información de los eventos del sensor de proximidad de manera persistente en el archivo SharedPreferences cuando llamo al método guardarEventosSharePreferences(), desde el método onPause lo que hago es iniciar la creación del archivo SharePreferences asignándole a una instancia de la clase SharedPreferenced el método getSharedPreferenced, luego con una instancia de la clase SharePreferenced.Editor le asignamos instanciaSharePreferenced.edit() para poder empezar a escribir en el archivo SharedPreferenced, luego verficio si el archivo SharePreferenced tiene algo escrito en su contenido, si tiene algo lo borro y vuelvo a escribir porque el archivo SharedPreferenced es inmutable, es decir, no permite la escritura o actualización ya existente en el mismo. Luego de esto, seteo en la instancia de SharedPreferences.Editor un hashSet y finalmente hago el commit para que la informacion seteada se escriba la información de manera persistente en el archivo. [3]
- Cuando tengo que leer la información de los eventos del sensor de proximidad grabados previamente de manera persistente en el archivo SharedPreferences cuando llamo al método cargarEventosSharePreferences(), desde el método onResume lo que hago es asignarle el método getSharedPreferenced a una instancia de la clase SharedPreferences previamente creada, luego le asignamos a una instancia de HashSet el método getStringSet de la instancia de la clase SharedPreferences para recuperar los eventos previamente escritos en el archivo. [3]

Recaudos que tuve para que la aplicación sea tolerante a fallos:

Para pruebas fluida lo primero que tuve en cuenta es que la aplicación no lance errores durante la ejecución , no se cuelgue ni se bloque cuando la usamos como por ejemplo cuando hacemos una petición al servidor, me fije que la aplicación no se bloqueara.

Para la liberación de recursos tuve en cuenta que se liberen los sensores y finalicen los hilos correspondientes. [4]

Para cambios de estado tuve en cuenta que tengo que liberar los recursos en los estados adecuados por ejemplo si “desregistro” un sensor en el estado onPause debo volver a

registrarlo en el estado onResume porque si no la aplicación va a fallar porque el sensor estaría no estaría registrado.

Para errores de conexión tuve en cuenta que ante un error como por ejemplo que el usuario que intento registrar ya exista en el servidor, la aplicación muestre un mensaje amigable al usuario para que este último entiendo que es lo que sucede y tenga una buena experiencia de usuario.

Conclusiones:

Después de haber desarrollado la aplicación para mi EA2 llegue a la conclusión de que todo lo que aprendí fue muy útil porque reforcé conocimientos previos en especial en los temas de sincronización de hilos y comunicación de componentes . Además aplique el conocimiento en estos temas mencionados a casos prácticos útiles como la comunicación con la api del servidor de la catedra . Y como si fuera poco aprendí lo que era una api y su utilidad, y además a utilizar Json. Con esto finalizo mi conclusión diciendo que este EA2 me aporto el valor de dejarme una buena base para empezar a desarrollar aplicaciones y poder construir un negocio con las mismas.

Bibliografía

[1] E. Carnuccio, «Apunte Teorico Android,» 2020.

[2] M. Sure, «Mort Sure,» 2019. [En línea]. Available: https://es.mort-sure.com/blog/difference-between-thread-service-and-asynctask-in-android-ffe1fc/.

[3] Google, «Developer Android,» 2020. [En línea]. Available: https://developer.android.com/reference/android/content/SharedPreferences.

[4] E. Carnuccio, «Thread y Sincronización,» 2020.