Sin Aliento

Del Rio Nicolás Ezequiel, Sólimo Martín Leandro, Vitale Luciano Nahuel
DNI 32478117, DNI 36088705, DNI 40389087
Dia de cursada: Miercoles, Numero de Grupo: 6

[1]Universidad Nacional de La Matanza,
Departamento de Ingeniería e Investigaciones Tecnológicas,
Florencio Varela 1903 - San Justo, Argentina

Resumen. En este paper se explica como funciona la app Sin Aliento, cuya funcionalidad es controlar que una persona acostada boca arriba tenga respiración al apoyar en celular en el pecho, y en el caso de no detectarla, emite una señal sonora como alarma y envía un mensaje de texto a otro celular registrado, como el de un familiar. Su utilización esta pensado en personas que tengan sintomas o confirmación de Covid positivo, y estén en aislamiento en sus hogares (no internadas).

Palabras claves: Covid, Android, App, Respiración, Aislamiento.

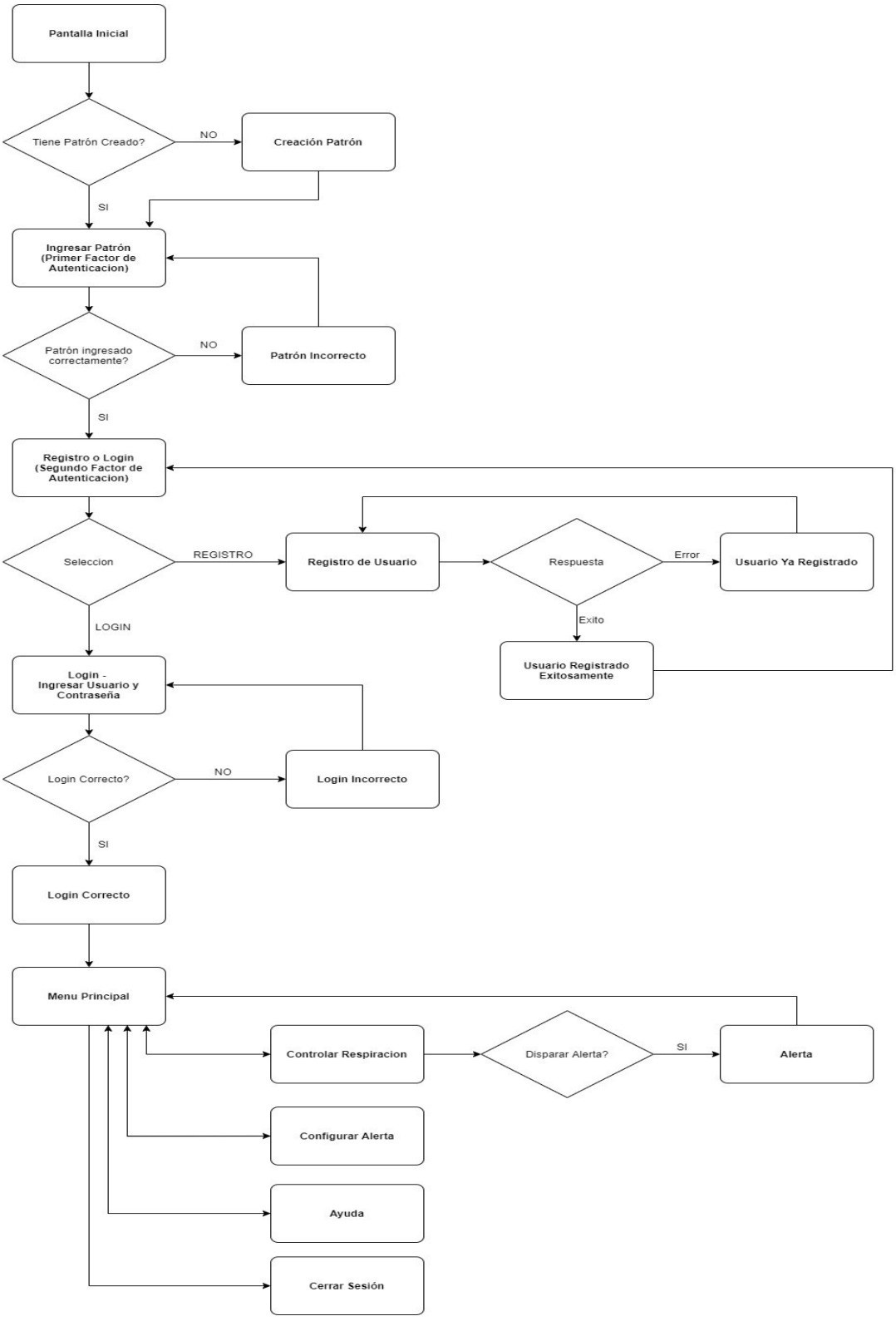
1 Introducción

- Entre los síntomas de un cuadro grave de la COVID-19 se incluyen la Disnea (dificultad respiratoria) [3].
  La funcionalidad de la aplicación es controlar que una persona acostada boca arriba tenga respiración al apoyar en celular en el pecho. Para ello, utliza los sensores de proximidad [1] [9] para verificar que el celular este apoyado sobre el pecho, y el sensor giroscopio [2] [8] para detectar la repiración por el movimiento del pecho al inhalar y exhalar .
  En reposo, en un adulto medio, tienen lugar alrededor de 15 respiraciones por minuto [4]. Por este motivo, al pasar más de 10 segundos sin detectar movimiento se dispara la alarma.

2 Desarrollo

- **Dirección web del repositorio GitHub**:
  https://github.com/SOA2021C1MieG6/TrabajosPracticosSOA

- **Diagrama funcional/navegación de las Activities.**



- **Implementación de la ejecución concurrente del programa y mecanismo de sincronización**
  Los hilos son fundamentales para una interacción fluida con el usuario. Si una aplicación realiza una operación lenta en el mismo hilo de ejecución de la interfaz gráfica, el lapso de tiempo que dure la conexión, la interfaz gráfica dejará de responder. Este efecto es indeseable ya que el usuario no lo va a comprender, ni aunque la operación dure sólo un segundo. Es más, si la congelación dura más de dos segundos, es muy probable que el sistema operativo muestre el diálogo ANR, "Application not responding", invitando al usuario a matar la aplicaicón. Para evitar esto hay que crear otro hilo (Thread) de ejecución que realice la operación lenta. [7]
  Cuando varios hilos pueden acceder simultáneamente a algún dato, hay que asegurarse de que no lo hagan simultáneamente en el caso de estar modificándolo. Para ello utilizaremos el modificador synchronized sobre métodos. De esta manera si un hilo empieza a ejecutar este método, todos los demás hilos que intenten ejecutarlo tendrán que esperar a que el primero que entró salga del método. [5] [6]

- **Comunicación entre los componentes** [10]
  Los componentes de la aplicación son bloques de creación esenciales de una aplicación para Android. Cada componente es un punto de entrada por el que el sistema o un usuario ingresan a tu aplicación. Algunos componentes dependen de otros.
  Las aplicaciones tienen cuatro tipos de componentes diferentes:
  1. Actividades
  2. Servicios
  3. Receptores de emisiones
  4. Proveedores de contenido

  Cada tipo tiene un fin específico y un ciclo de vida característico que define cómo se crea y se destruye el componente.

  1. Una actividad es el punto de entrada de interacción con el usuario. Representa una pantalla individual con una interfaz de usuario. Una actividad posibilita las siguientes interacciones clave entre el sistema y la aplicación:
     - Realizar un seguimiento de lo que realmente le interesa al usuario (lo que está en pantalla) para garantizar que el sistema siga ejecutando el proceso que aloja la actividad.
     - Saber que los procesos usados con anterioridad contienen elementos a los que el usuario puede regresar (actividades detenidas) y, en consecuencia, priorizar más esos procesos que otros.
     - Ayudar a la aplicación a controlar la finalización de su proceso para que el usuario pueda regresar a las actividades con el estado anterior restaurado.
     - Permitir que las aplicaciones implementen flujos de usuarios entre sí y que el sistema los coordine (el ejemplo más común es compartir).

     Las actividades se implementan como subclases de la clase **Activity**.
  2. Un servicio es un punto de entrada general que permite mantener la ejecución de una aplicación en segundo plano por diversos motivos. Un servicio no proporciona una interfaz de usuario. Existen dos semánticas muy diferentes que los servicios usan para indicarle al sistema cómo administrar una aplicación: Los servicios iniciados le indican al sistema que los siga ejecutando hasta que finalicen su trabajo. Los servicios enlazados se ejecutan porque otra aplicación (o el sistema) indicó que quiere usarlos. Básicamente, lo que sucede es que un servicio le brinda

una API a otro proceso. Por lo tanto, el sistema sabe que hay una dependencia entre estos procesos. En consecuencia, si el proceso A está enlazado a un servicio en el proceso B, sabe que debe mantener funcionando el proceso B (y el servicio correspondiente) para el proceso A. Además, si el proceso A es de interés para el usuario, también sabe que debe tratar el proceso B teniendo esto en cuenta. Gracias a su flexibilidad (o a pesar de ella), los servicios se han convertido en un componente muy útil para todos los tipos de conceptos generales del sistema.

Un servicio se implementa como una subclase de **Service**.

3. Un receptor de emisión es un componente que posibilita que el sistema entregue eventos a la aplicación fuera de un flujo de usuarios habitual, lo que permite que la aplicación responda a los anuncios de emisión de todo el sistema. El sistema puede entregar emisiones incluso a las aplicaciones que no estén en ejecución. Si bien los receptores de emisión no exhiben una interfaz de usuario, pueden crear una notificación de la barra de estado para alertar al usuario cuando se produzca un evento de emisión. Sin embargo, por lo general, un receptor de emisión es simplemente una puerta de enlace a otros componentes y está destinado a realizar una cantidad mínima de trabajo. Por ejemplo, podría programar un servicio **JobService** para que realice algunas tareas en función del evento con **JobScheduler**. Un receptor de emisión se implementa como una subclase de **BroadcastReceiver** y cada receptor de emisión se entrega como un objeto **Intent**. Para obtener más información, consulta la clase **BroadcastReceiver**.
4. Un proveedor de contenido administra un conjunto compartido de datos de la aplicación que puedes almacenar en el sistema de archivos, en una base de datos SQLite, en la Web o en cualquier otra ubicación de almacenamiento persistente a la que tenga acceso tu aplicación. A través del proveedor de contenido, otras aplicaciones pueden consultar o modificar los datos si el proveedor de contenido lo permite. Para el sistema, un proveedor de contenido es un punto de entrada a una aplicación para publicar elementos de datos con nombre y se identifica mediante un esquema de URI. Así, una aplicación puede decidir cómo quiere asignar los datos que contiene a un espacio de nombres de URI y entregar esos URI a otras entidades que, a su vez, pueden usarlos para acceder a los datos.

   Los proveedores de contenido también son útiles para leer y escribir datos privados de tu aplicación y que no se comparten.

   Un proveedor de contenido se implementa como una subclase de **ContentProvider** y debe implementar un conjunto estándar de API que permitan a otras aplicaciones realizar transacciones.

   Activación de componentes

   De los cuatro tipos de componentes, tres (actividades, servicios y receptores de emisión) se activan mediante un mensaje asíncrono denominado intent. Las intents vinculan componentes entre sí durante el tiempo de ejecución. Imagina que son los mensajeros que solicitan acciones de otros componentes, ya sea que estos pertenezcan a tu aplicación o a alguna otra.

   Una intent se crea con un objeto **Intent**, que define un mensaje para activar un componente específico (intent explícita) o un tipo específico de componente (intent implícita). Para actividades y servicios, una intent define la acción que se realizará (por ejemplo, ver o enviar algo) y puede especificar el URI de los datos en los que debe actuar, entre otras cosas que el componente que se está iniciando podría necesitar saber. En el caso de los receptores de emisión, la intent tan solo define el anuncio que se

emite. A diferencia de las actividades, los servicios y los receptores de emisión, los proveedores de contenido no se activan con intents.
Existen métodos independientes para activar cada tipo de componente:

Puedes iniciar una actividad o asignarle una tarea nueva al pasar un **Intent** a **startActivity()** o **startActivityForResult()** (cuando quieres que la actividad devuelva un resultado).
Con Android 5.0 (nivel de API 21) y las versiones posteriores, puedes usar la clase **JobScheduler** para programar acciones. En versiones anteriores de Android, puedes iniciar un servicio (o darle instrucciones nuevas a un servicio en curso) al pasar un **Intent** a **startService()**. Puedes establecer un enlace con el servicio al pasar un **Intent** a **bindService()**.
Puedes iniciar una emisión al pasar un **Intent** a métodos como **sendBroadcast()**, **sendOrderedBroadcast()** o **sendStickyBroadcast().**
Puedes realizar una consulta a un proveedor de contenido si llamas a **query()** en un **ContentResolver**.

- **Técnica que utilizó para la comunicación con el servidor**
  Retrofit es una librería para hacer llamadas red y obtener el resultado estructurado de una vez. La labor manual de utilizar un HttpClient y luego usar json.get para construir nuestros objetos es cosa del pasado con Retrofit.
  Retrofit requiere la creación de dos archivos especiales, una interfaz y un cliente. [11] [12]

- **Implementación de la persistencia de los datos en la aplicación**
  La persistencia de los datos se realiza a través de peticiones API con protocolo REST con los mensajes “Request / Response”. [13]

- **Manual de Usuario de la Aplicación Android**
  La iniciación de sesión se realiza a con doble factor de autentización.
  La primera vez que ejecuta la aplicación, debe agregar un patrón y memorizarlo. El mismo será requerido cada vez que abra la aplicación. En el caso de olvidarlo, debera desinstalar la app y reinstalarla.

  Ingresado el patrón, tiene la opción de registrar un usuario o de loguearse.
  Para registrar un usuario debe ingresar:
  - Nombre.
  - Apellido.
  - DNI.
  - Email.
  - Password.
  - Número de Comisión.
  - Número de Grupo.

  Para loguear un usuario debe ingresar:
  - Email.
  - Password.

  Una vez logueado el usuario, se muestra el Menú Principal con las siguientes opciones:
  - Controlar Respiración: Seleccionada esta opción, tiene unos 5 segundos para apoyar el celular en el pecho (emitiendo una señal sonora como conteo).

Si el sensor de proximidad no detecta que está apoyado y/o no detecta la respiración al principio, emita una señal indicando que no se empezó a ejecutar correctamente.
Si detecto la proximidad y la respiración, si luego de 10 segundos no detecta movimiento (respiración) pero si proximidad, se interpreta que dejo de respirar y envía el mensaje de texto al número de celular registrado.
Si deja de detectar proximidad, deja de controlar la respiración, volviendo al Menú Principal

- Configurar Alerta: Se configura la señal sonora en caso de no detectar respiración, y la opción de ingresar un número de celular para enviar un mensaje de texto.
- Ayuda: Explicacion de las funcionalidades “Controlar Respiración” y “Configurar Alerta”.
- Cerrar Sesión.

3 Conclusiones

- Se incorporaron validaciones en los campos tanto de Login como del Formulario de Registro, con el fin de evitar, que se envíen caracteres inválidos en las solicitudes al servidor. Evitando llamados al servidor que pueden venir con error y para mejorar la facilidad de uso por parte de los usuarios, al indicar con mensajes aclarativos que campos presentan errores.
- Se ha presentado el problema a la hora de hacer solicitudes al servidor, por ser peticiones HTTP, el error:
  “*java.net.UnknownServiceException: CLEARTEXT communication to http://so-unlam.net.ar not permitted by network security policy*”
  Diciendo cleartext o texto claro, es decir, que no está encriptado como las comunicaciones por HTTPS.
  La solución ha sido en el AndroidManifest.xml, dentro de <application> agrega la siguiente línea: android:usesCleartextTraffic="true".
- El trabajo nos ha permitido profundizar en el uso tanto de sensores como de métodos en segundo plano utilizados en Android, así como también problemas respiratorios antes no conocidos.

4 Referencias

[1] «Sensores de movimiento | Desarrolladores de Android,» Desarrolladores de Android, [En línea]. Available: https://developer.android.com/guide/topics/sensors/sensors_motion.

[2] «Sensores de posición | Desarrolladores de Android,» Desarrolladores de Android, [En línea]. Available: https://developer.android.com/guide/topics/sensors/sensors_position.

[3] «Respiracion,» Wikipedia, [En línea]. Available: https://es.wikipedia.org/wiki/Respiraci%C3%B3n.

[4] «Informacion Básica sobre la COVID-19,» Organizacion Mundial de la Salud, [En línea]. Available: https://www.who.int/es/emergencies/diseases/novel-coronavirus-2019/question-and-answers-hub/q-a-detail/coronavirus-disease-covid-19.

[5] «Métodos sincronizados - Java,» [En línea]. Available: https://docs.oracle.com/javase/tutorial/essential/concurrency/syncmeth.html.

[6] «Sincronización de hilos,» Universidad de Alicante, [En línea]. Available: http://www.jtech.ua.es/dadm/2011-2012/restringido/android-av/sesion01-apuntes.html#Sincronizaci%C3%B3n+de+hilos.

[7] «Hilos de Ejecución,» Universidad de Alicante, [En línea]. Available: http://www.jtech.ua.es/dadm/2011-2012/restringido/android-av/sesion01-apuntes.html#Hilos+de+ejecuci%C3%B3n.

[8] «¿Qué es el Giroscopio del Móvil y Cómo Funciona?,» Youtube, [En línea]. Available: https://www.youtube.com/watch?v=mWy6p5WRDjM.

[9] «Sensor de proximidad en Android - Bytes,» Youtube, [En línea]. Available: https://www.youtube.com/watch?v=Di7WubMHvak.

[10] «Componentes de la Arquitectura de Android,» Developer Android, [En línea]. Available: https://developer.android.com/guide/components/fundamentals.

[11] «Retrofit,» [En línea]. Available: https://square.github.io/retrofit/.

[12] «Retrofit para Android desde 0,» [En línea]. Available: https://medium.com/contraslashsas/retrofit-para-android-desde-0-1c8be830a1af.

[13] «JSON - SERVER,» [En línea]. Available: https://www.npmjs.com/package/json-server.