Universidad Nacional de La Matanza

Departamento de Ingeniería

Sistemas Operativos Avanzados

Reentrega de Evaluación de Aprendizaje N°2

Fecha: 13/11/2020

Profesores:

Graciela de Luca

Waldo A. Valiente

Sebastián Barillaro

Mariano Volker

Carnuccio Esteban Andrés

Gerardo García

Alumnos:

Pablo Bermudez - DNI: 35337444 - pablobermudez2@gmail.com

Curso:

Martes – Noche 19hs a 23hs

Comisión:

02-2900

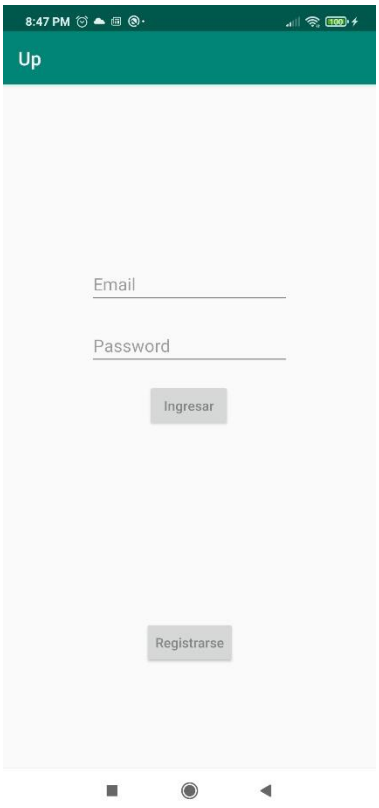
Introducción

Al momento de esta entrega, se ha llegado a solamente a realizar funcionalidades solicitadas por la cátedra en los requisitos propuesto de la Evaluación de Aprendizaje N°2. Dicho esto, esta aplicación utilizará la API propuesta por la cátedra para registrarse, ingresar y guardar eventos.

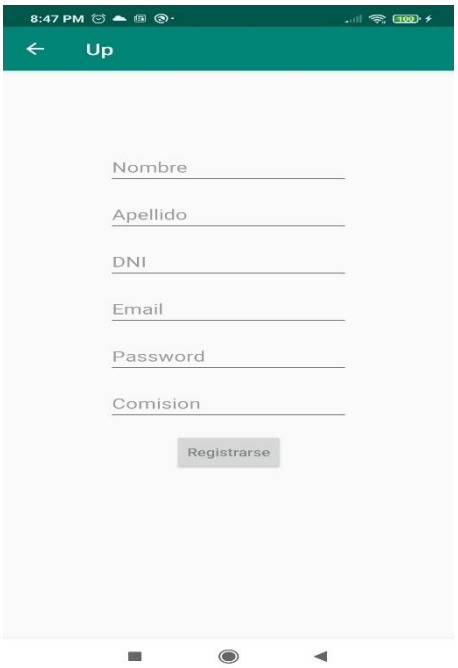
Además, se leerán los sensores Acelerómetro y el sensor vectorial de rotación para mostrar sus datos por pantalla. Adicionalmente, en el menú de lectura del sensor vectorial se ha implementado la posibilidad de guardar la información brindada por el sensor mediante SharedPreferences.

Adicionalmente, la aplicación puede recibir mensajes desde el servicio brindado por Firebase, el token para enviar estos mensajes se mostrará en el terminal de errores al iniciar la aplicación.

Manual de Usuario



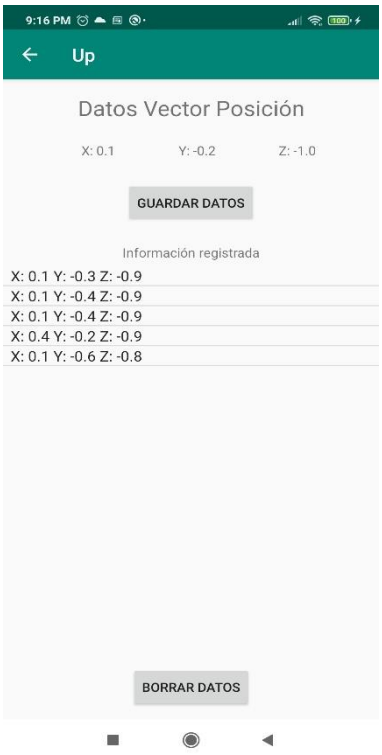
- Nuestra vista principal es bastante sencilla, se podrá ingresar a la aplicación o registrarse.
- El ingreso a la aplicación estará validado a través de una librería externa incluida en el gradle con el siguiente código: **implementation 'com.mobsandgeeks:android-saripaar:2.0.3'**
- Al intentar ingresar a la aplicación, la misma validará la existencia del usuario con el servicio provisto por la cátedra.
- Se guardará dicho evento utilizando el servicio proporcionado por la cátedra.



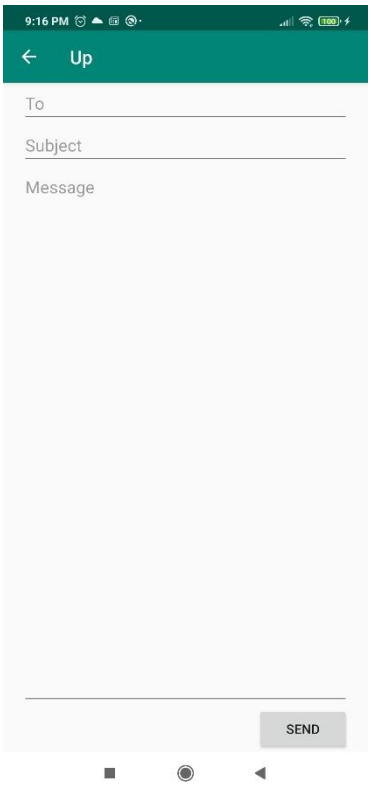
- Nuestra vista de registro será igual de sencilla que la de ingreso, se podrá registrar un usuario ingresando los datos que figuran en la misma.
- Este formulario que se envía al registrarse también se encuentra validado por lo que no permitirá ingresos erróneos de datos al intentar registrarse informando al usuario el problema en el campo puntual.
- Al registrarse, se utilizará el servicio de registro proporcionado por la cátedra.
- Se guardará dicho evento utilizando el servicio proporcionado por la cátedra.

- Nuestra vista de Home dará la posibilidad de acceder a las vistas donde se verán los datos del acelerómetro o del sensor vectorial.
- Se mostrará el porcentaje de batería que posee el teléfono al momento del ingreso en la parte inferior de la pantalla.
- Se podrá ejecutar una funcionalidad muy sencilla que envía un email al presionar el botón “Send Email” la cual llamará a otra aplicación para realizar dicha acción.

- Al ingresar desde el Home al acelerómetro, se verá una pantalla que mostrará simplemente los datos obtenidos por este sensor los cuales se actualizarán cuando los mismos cambien.



- Al ingresar desde el Home a ver los datos del sensor vectorial, se verán los datos obtenidos por este sensor los cuales irán variando según cambie dicha lectura del sensor.
- Adicionalmente, se podrán guardar estos mismos datos de manera persistente los cuales se irán mostrando en una tabla en la misma vista al presionar el botón “Guardar datos”.
- Se podrán borrar los datos almacenados con el botón inferior “Borrar datos”.
- Se ha utilizado SharedPreferences para poder realizar el guardado de datos persistente.

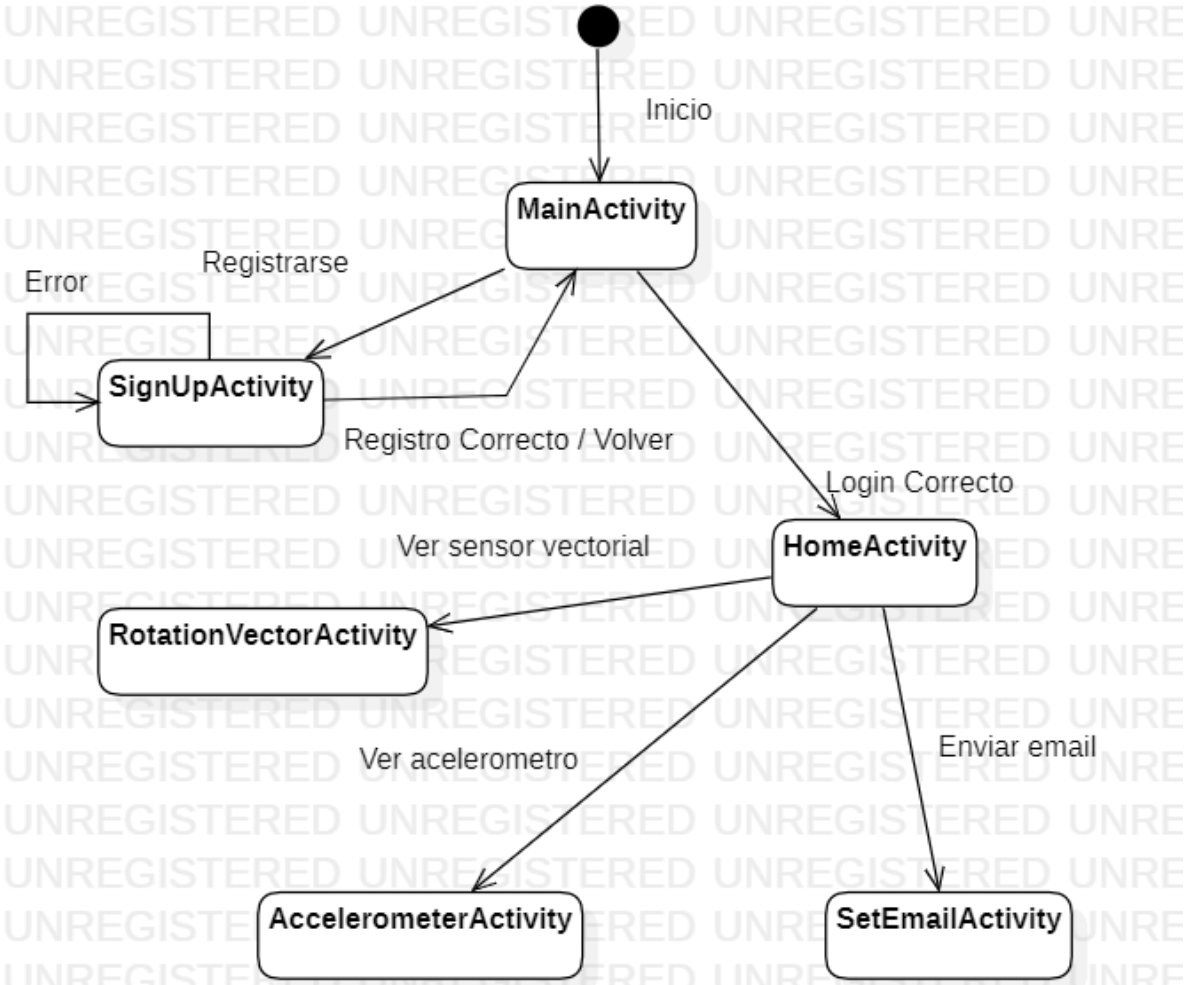


- Desde el Home, será posible ingresar a una pequeña funcionalidad para enviar un email, la misma utilizará otra aplicación para realizar dicha tarea.

Repositorio GitHub

https://github.com/pablobermudez/Up

Realice un diagrama funcional/navegación de las Activities.



Describa cómo realizó la sincronización de la ejecución concurrente del programa

En la aplicación, la sincronización de procesos o tareas concurrente realizó implementando distintos servicios que serán llamados desde las Activities para ejecutarse en segundo plano mientras que la ejecución del programa principal continua. El resultado de estos servicios ejecutados en segundo plano es recibido por clases que extienden de BroadcastReceiver las cuales se instanciaran justo antes de iniciar el servicio en la misma Activity y realizarán distintas acciones según la respuesta que les provea dicho servicio.

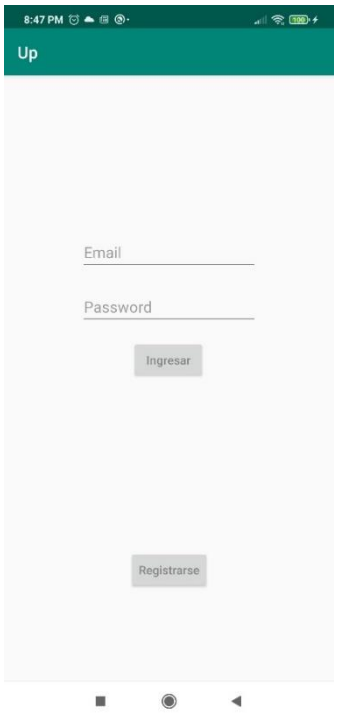
Según la distribución del código, en el paquete Model se verán dos sub-paquetes:

- El paquete Requests que será el que agrupe los servicios a excepción del servicio declarado para manejar FireBase.
- El paquete Receivers que será el que agrupe las clases que extienden de BroadcastReceiver y actuarán según la devolución de los servicios instanciados.

Esta implementación me permitió la ejecución de servicios de manera asincrónica, y delegó la responsabilidad de realizar acciones según la devolución de los propios servicios a los Receivers brindándome de esa manera la posibilidad de ejecutar una o más tareas de manera concurrente.

Para mejor comprensión de este hecho, vamos a dar un ejemplo concreto de la aplicación:

Para iniciar, nos ubicamos en la vista inicial de la aplicación:



En esta vista, veremos la existencia de un formulario y un botón con la leyenda “Ingresar”. El cual ejecutará el método denominado “SignIn” correspondiente al MainActivity.

En dicho método, de ingresar los datos correctamente y tener conexión a internet, se ejecutará el siguiente bloque de código:

```
JSONObject jsonObject = new JSONObject();
jsonObject.put( name: "email", txtEmail.getText().toString());
jsonObject.put( name: "password", txtPassword.getText().toString());

Intent intentLogin = new Intent( packageContext: MainActivity.this, RegisterRequest.class);
intentLogin.putExtra( name: "uri", StaticServiceData.LOGIN_URI);
intentLogin.putExtra( name: "jsonData",jsonObject.toString());
intentLogin.putExtra( name: "action", StaticServiceData.LOGIN_ACTION);

configureBroadCastReceiver();
startService(intentLogin);
```

Notamos entonces una serie de pasos en este bloque de código:

El primer paso relevante será instanciar un objeto de la clase Intent denominado “intentLogin” el cual además de indicar que servicio será llamado, también contendrá toda la información relevante que será transmitido desde la propia Activity al servicio en el momento de iniciarlo.

Luego de terminar de construir el objeto y guardar en él toda la información necesaria para la ejecución del servicio, se invocará a un método denominado “configureBroadCastReceiver” el cual se encargará de iniciar un BroadcastReceiver, denominado “ReceiverLogin”, que escuchará la devolución de dicho servicio y actuará según dicha devolución.

Finalmente, una vez instanciado el Receiver, se invocará la ejecución del servicio, el cual correrá en segundo plano mientras que el sistema sigue su ejecución en el hilo principal.

Si dicho servicio se ejecuta y da una devolución que es escuchada por el Receiver, dicha respuesta en caso de ser incorrecta mostrará un mensaje de error y en caso de ser correcta mostrará un mensaje de ingreso exitoso e iniciará la activity “HomeActivity”.

En este caso, además, el Receiver anteriormente mencionado llamará a la ejecución de otro servicio que también será ejecutado en segundo plano de manera asincrónica el cual se comunicará con la API brindada por la cátedra para que se registre el evento de ingreso exitoso por parte del usuario.

Mostramos entonces el siguiente código que será ejecutado por el “ReceiverLogin” en caso de que la respuesta del servicio “RegisterRequest” haya sido exitosa:

```
if (responseLogin.getSuccess() == true ) {
    Toast.makeText(context.getApplicationContext(), text: "Login exitoso", Toast.LENGTH_LONG).show();
    StaticServiceData.sessionToken = responseLogin.getToken();
    StaticServiceData.sessionTokenRefresh = responseLogin.getToken_refresh();
    EventRequestHandler.saveEvent(context, StaticEventTypes.EV_LOGIN, StaticEventTypes.EV_LOGIN_DESC_SUCCESS);
    context.startActivity(new Intent(context, HomeActivity.class));
}
```

Notamos en el código que además de enviar un mensaje informando del éxito del ingreso e indicar el cambio de Activity, este bloque de código en su ante última línea llamará a otra clase denominada “EventRequestHandler”, el cual finalmente terminará por llamar al servicio “EventRequest” que se comunicará con la API brindada por la cátedra para guardar el evento de “Login Exitoso” del usuario.

El servicio “EventRequest” no tendrá ningún Receiver que escuche su devolución ya que el mismo no afectará a la ejecución del hilo principal en ningún momento.

En resumen, vemos en este ejemplo que se ejecutarán dos servicios asincrónicos:

- Uno será invocado desde el MainActivity y validará los datos ingresados por el usuario usando la API de la cátedra. En la misma Activity se instanciará el Receiver que escuchará la devolución de este servicio y actuará en consecuencia.
- El otro servicio que se ejecutará de manera asincrónico inicia su ejecución en el mismo Receiver, el cual lo llamará de haber un ingreso exitoso y utilizará la API de la cátedra para almacenar dicho evento.

Describa cómo realizó la comunicación entre los componentes (Activites, Servicios, etc.).

La comunicación entre Activities y Servicios fue realizada a través de la clase IntentService, se instancia esta clase en cada Activity que llame a algún servicio como por ejemplo los servicios relacionados con la API brindada por la cátedra. Se transfiere información dentro de los objetos instanciados en cada Activity los cuales se utilizan en la clase del servicio.

Luego de que el servicio haya podido realizar sus tareas, la devolución será escuchada por un BroadcastReceiver, el cual también se instancia en la Activity al mismo momento que el Intent vinculado a dicho servicio. Finalmente, el BroadcastReceiver realiza acciones determinadas dado la devolución brindada por el servicio.

¿Qué técnica utilizó para la comunicación con el servidor (HttpConnection, Retrofit, etc)?. Detalle como lo implementó desde la generación de la solicitud al servidor hasta la recepción de la respuesta.

Para la comunicación con el servidor se utilizó HttpConnection, para esto se implementaron dos servicios distintos, uno que se comunicaba con el servidor para realizar el Login y el Registro, y otro algo diferente que se utilizó para registrar eventos.

En las activities que realizaban Login o Registro, se generó un objeto Json dentro de dicha activity, el cual era transferido al servicio mediante un Intent además de la URI y el ACTION relacionados con el BroadcastReceiver al cual estos servicios debían responder.

El registro de eventos de Login y Registro se realiza dentro de los mismos BroadcastReceiver que reciben la respuesta de los servicios, al ser el Login o el Registro exitosos entonces dentro de dicho Broadcast se llama a otro servicio que realiza el POST para guardar este evento.

El servicio que envía el evento al servidor no responde a ningún BroadcastReceiver, ya que se consideró que no afecta el funcionamiento de la aplicación si este servicio se ejecutó o no correctamente más que informarlo por consola.

Durante el desarrollo ¿Surgieron problemas? ¿Cómo fueron resueltos?

Durante el desarrollo surgieron variedad de inconvenientes en relación al diseño que debe llevar el sistema entre otros de implementación.

Se optó por un diseño en donde las activities estén agrupadas en la carpeta raíz, mientras que se generó un paquete denominado “Model” para agrupar los servicios, los receivers, los modelos y las clases con atributos estáticos útiles.

Se encontraron variedad de problemas en el uso de sensores debido a que el dispositivo personal no posee varios de los sensores más útiles para poder utilizar en una aplicación, debido a este motivo se optó por usar dos de los pocos que poseía que fueron el acelerómetro y el sensor vectorial.