

Trabajo Final:

Introducción a la Programación de Dispositivos Móviles



Profesor:

Dr. Mauricio Arroqui

Integrantes:

Chiozza Logroño, Juan Ignacio. (email: chiozzajuani@gmail.com).

San Roman, Matias (email: matias.sanroman.32@gmail.com).

Introducción:

Se requiere implementar una aplicación que maneje los reciclados de un usuario.

Las peticiones al servidor varían en su URL respecto al enunciado provisto por la cátedra, ya que la parte de la API REST la desarrollamos en la optativa de COMPUTACION ORIENTADA A SERVICIOS.

El usuario cuando abra la aplicación podrá registrar un usuario. Lo cual podrá realizar enviando la petición al servidor provisto por la cátedra de esta forma:

Método POST

URL: `http://localhost:8080/api/usuario`

Body:

```
{  
  "firstname": "Matias",  
  "lastname": "San Roman",  
  "username": "Matt",  
  "address": "Pinto123",  
  "mail": "matt@gmail.com"  
}
```

Este usuario deberá ser recordado por la aplicación cada vez que se abra nuevamente, a no ser que otro usuario sea registrado y ahora será este el que aparezca por defecto cuando la aplicación es abierta. Por simplicidad de la solución no es necesario que se permita seleccionar con que usuario iniciar.

Luego se podrán registrar reciclados para el usuario actual. El usuario podrá ir almacenando localmente en la aplicación la cantidad reciclada de cada uno de los componentes (bottles, tetrabriks, glass, paperboard, cans) hasta que finalmente decida enviarlo a servidor (el usuario lleva lo reciclado a punto limpio Tandil o avisa al camión para que ya pueda pasar a buscarlo), lo cual será realizada enviando la petición al servidor de esta forma:

Método POST

URL: <http://localhost:8080/api/reciclado>

Body:

```
{  
  "bottles": 5,  
  "tetrabriks": 5,  
  "glass": 5,  
  "paperboard": 5,  
  "cans": 5,  
  "id_r": 1  
}
```

*Siendo el id_r, el id del usuario para identificar de quien es el reciclado.

La aplicación deberá permitir dos tipos de visualizaciones. La primera que muestre el listado de todos los reciclados, el cual se realizará enviando la petición al servidor:

Método GET

URL <http://localhost:8080/api/reciclados/id/>

*Siendo id, el id del usuario.

La segunda que muestre el total reciclado hasta la actualidad, para lo cual se realizará enviando la petición al servidor:

Método GET

<http://localhost:8080/api/reciclado/id/>

*Siendo id, el id del usuario.

Aclaraciones:

La app no permite Loguin, por lo tanto, cada vez que se cierre la sesión se deberá registrar un nuevo usuario.

Las peticiones se hicieron a través del ID y no del USERNAME, para controlar el tema de los repetidos. Pero a la hora de pedir el ID de cierto USERNAME el servidor no controla si ese USERNAME esta repetido, por lo tanto, será responsabilidad del usuario no registrar más de una vez el mismo nombre de usuario, ya que esto producirá un error en el servidor a la hora de listar los reciclados o realizar un envío de uno de ellos.

Resolución:

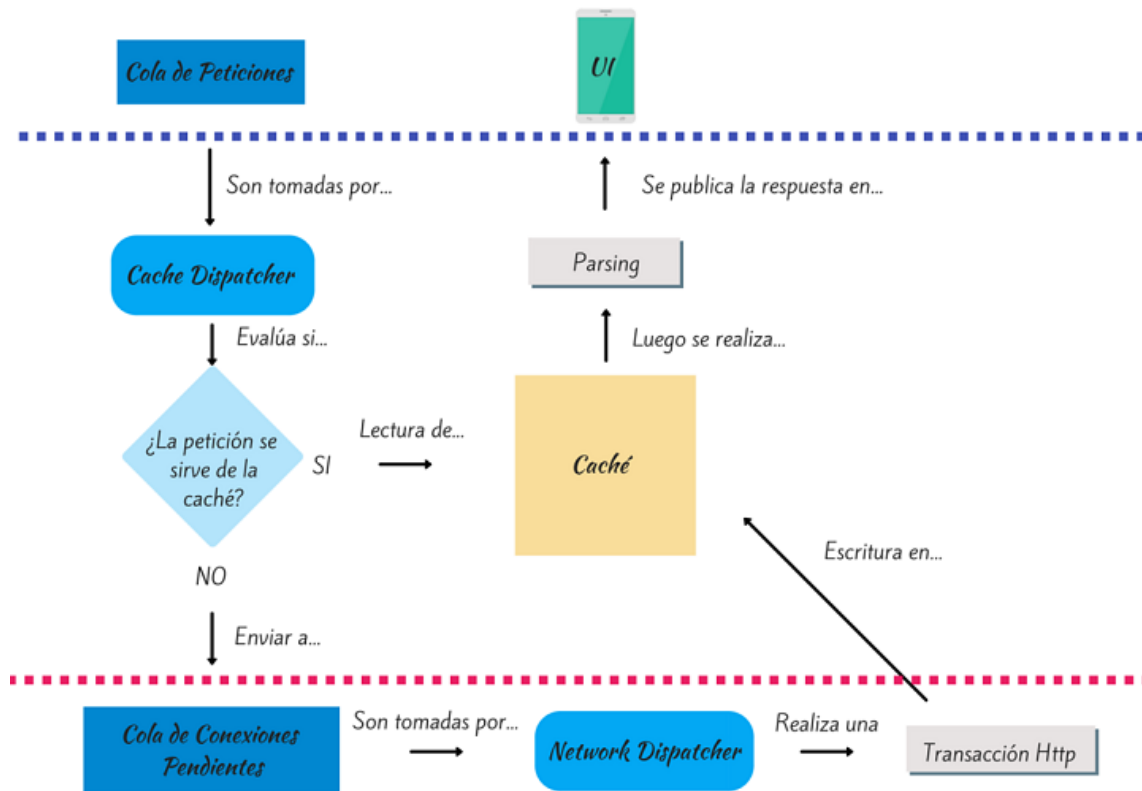
Para empezar, vamos a hablar como hicimos la conexión con nuestra API REST desde nuestra APP Android. Se utilizó la librería VOLLEY (desarrollada por GOOGLE) que sirve para hacer llamadas HTTP de forma sencilla sin tener que preocuparnos de la gestión de hilos.

Volley posee varios componentes que optimizan la administración de las peticiones generadas desde las aplicaciones Android. La gestión comienza en una Cola de Peticiones que recibe cada una de las peticiones generadas, donde son previamente priorizadas para su realización.

Luego son seleccionadas por un elemento llamado Cache Dispatcher, cuya función es comprobar si la respuesta de la petición actual puede ser obtenida de resultados previos guardados en caché. Si es así, entonces se pasa a parsear la respuesta almacenada y luego se presenta al hilo principal. En caso negativo, se envía la petición a la Cola de Conexiones Pendientes, donde reposan todas aquellas peticiones que están por ejecutarse.

Luego entra en juego un componente llamado Network Dispatcher, el cual se encarga de seleccionar las peticiones pendientes de la cola, para realizar las respectivas transacciones Http hacia el servidor. Si es necesario, las respuestas de estas peticiones se guardan en caché, luego se parsean y finalmente se publican en el hilo principal.

A continuación, se muestra una ilustración que resume el flujo:



En resumen, para hacer una llamada con Volley necesitaremos dos objetos:

Un objeto para declarar la petición (**Request**)

Un objeto que procese las peticiones (**RequestQueue**)

Tendremos diferentes tipos de objetos Request en función del tipo de respuesta que esperamos recibir en nuestra petición, por ejemplo:

StringRequest

JsonRequest

JSONArrayRequest

Cuando declaremos una petición usando un objeto Request, le indicaremos si la llamada es GET o POST, le diremos la URL a la que queremos hacer la llamada y le pasaremos los parámetros de llamada.

A continuación, se mostrará un ejemplo de como se hizo una llamada POST para el registro de usuario:

Primero, se crea la cola de peticiones.

```
cola = Volley.newRequestQueue(MainActivity.this);
```

Segundo, creo un objeto JSON.

```
//CREO OBJETO JSON
JSONObject j = new JSONObject();
try {
    j.put("firstname",firstname.getText().toString());
    j.put("lastname",lastname.getText().toString());
    j.put("username", username.getText().toString());
    j.put("address",address.getText().toString());
    j.put("mail", mail.getText().toString());
} catch (JSONException e) {
    e.printStackTrace();
}
```

Tercero, creamos la petición. Esta se ejecuta en segundo plano y de forma asíncrona. Con 2 callbacks, una para la respuesta afirmativa ONRESPONSE y otra por si falla ONERRORRESPONSE.

```
// CREO LA PETICION PARA ENVIAR AL SERVIDOR
JsonObjectRequest petition = new JsonObjectRequest(Request.Method.POST, url, j,
    new Response.Listener<JsonObject>() {

        //SI LA PETICION FUE EXITOSA, SE PASA A LA SIGUIENTE ACTIVIDAD.
        @Override
        public void onResponse(JsonObject response)
        {
            Intent i = new Intent(MainActivity.this, MenuPrincipal.class);
            startActivity(i);
            finish();
        }

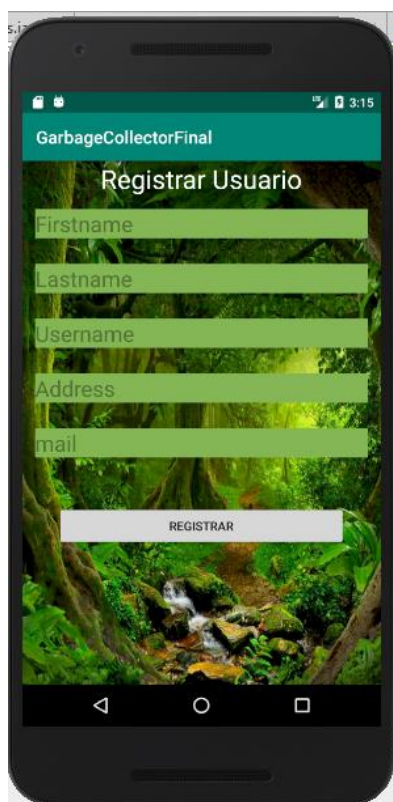
        //EN CASO DE ERROR, SE VUELVE A HABILITAR EL BOTON
    }, new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {
            Toast.makeText(MainActivity.this, error.toString(), Toast.LENGTH_SHORT).show();
            registrar.setEnabled(true);
            progressBar.setVisibility(View.INVISIBLE);
        }
    });
```

Y por último agregamos la petición a la cola para que se ejecute.

```
cola.add(peticion);
```

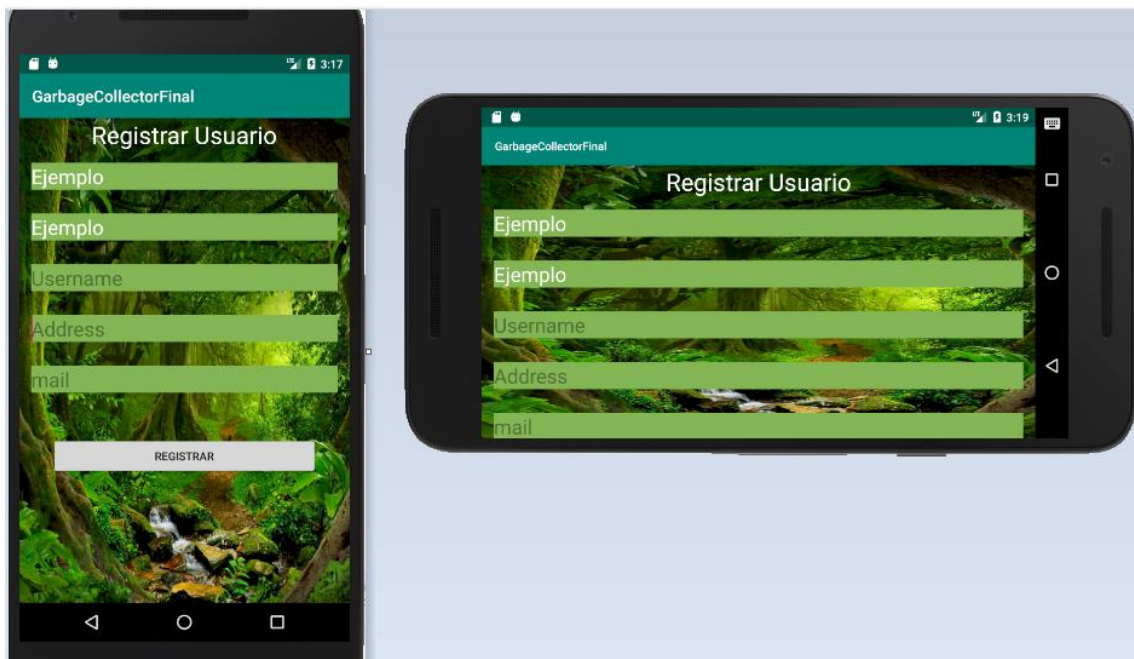
La App cuenta con 4 Activities: MainActivity, MenuPrincipal, Reciclar y ListasReciclados.

MainActivity: esta es la primera actividad que aparece cuando arranca la App o cuando el usuario cierre sección. Primero, va a comprobar si hay sección abierta, desde una variable guardada en un archivo usando Shared Preferences. En caso contrario se procede a realizar el registro de usuario. Para esto, como se explico mas arriba se crea un objeto JSON obteniendo los valores desde los EditText y se hace la petición a través de la librería VOLLEY con la llamada POST.



Se comprueba que el usuario haya completado todos los campos.

También cuenta con un Scroll por si no entra en la pantalla.

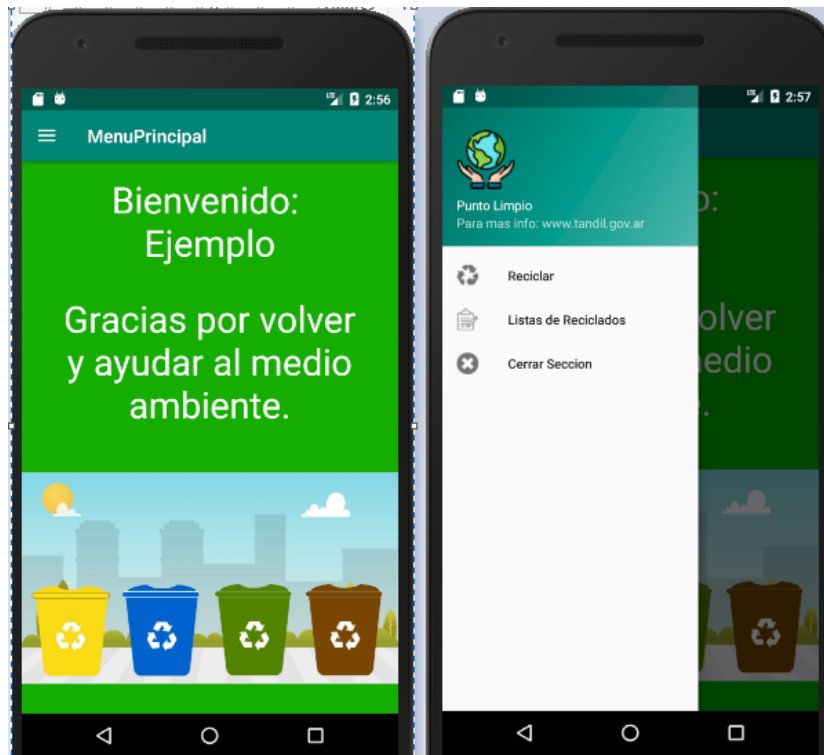


Esta actividad usa un método callback llamado `ONPAUSE()`.

El sistema llama a este método como el primer indicador de que el usuario está abandonando su actividad (aunque no siempre significa que la actividad se esté destruyendo). En este método se usa `Shared Preferences`, donde se guarda en un archivo el `Username` para su posterior uso en las peticiones `HTTP`.

En caso de que la sección este abierta, se finaliza esta actividad y se pasa a la nueva que es:

MenuPrincipal: esta actividad es llamada una vez que se registra el usuario o cuando el usuario ya se registro y no cerro sección. Se continua con el ejemplo anterior de crear un usuario llamado “Ejemplo”.

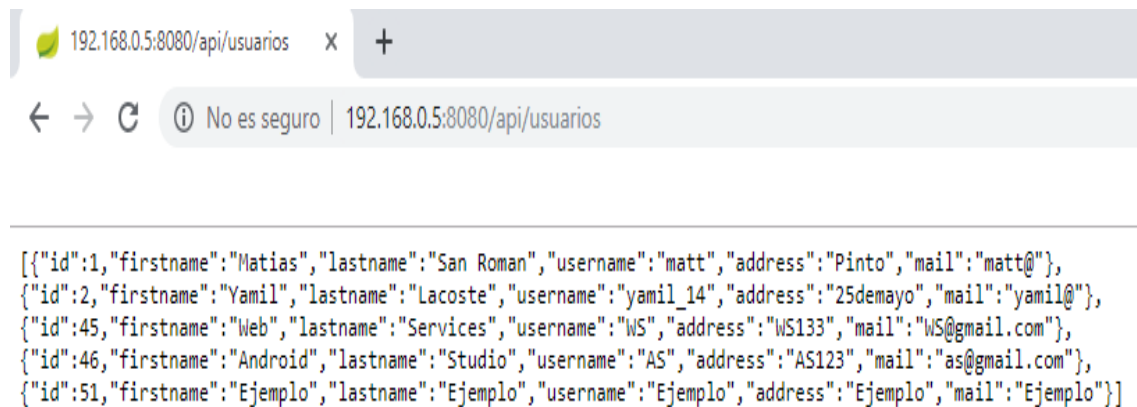


Como se puede observar, cuenta con un layout con un menú lateral (Navigation Drawer). Y con Shared Preferences primero se guarda una variable booleana en true de sección abierta, luego se captura el username y se da un mensaje de bienvenida.

A continuación, se muestra el estado de la base de datos antes y después de crear dicho usuario.

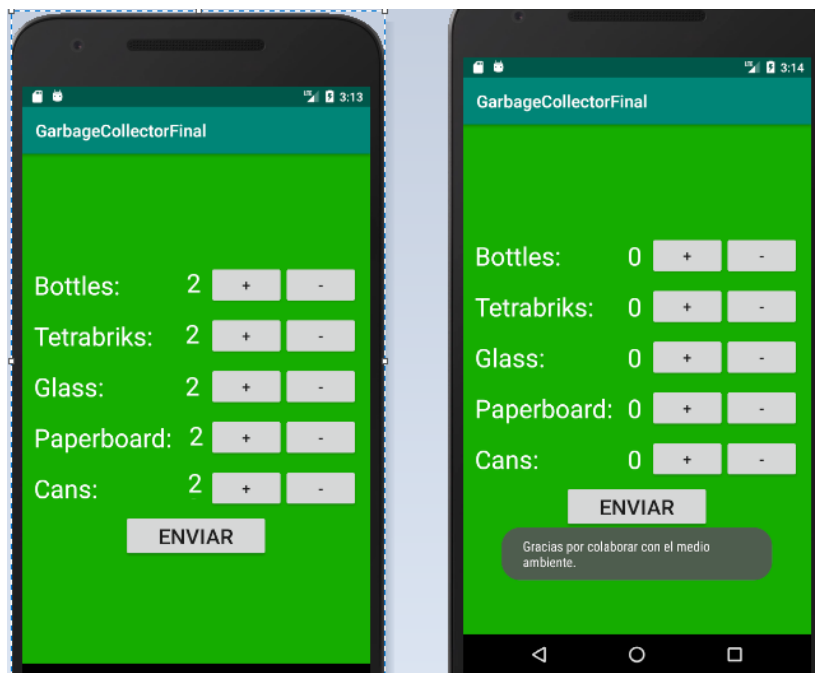


```
[{"id":1,"firstname":"Matias","lastname":"San Roman","username":"matt","address":"Pinto","mail":"matt@"}, {"id":2,"firstname":"Yamil","lastname":"Lacoste","username":"yamil_14","address":"25demayo","mail":"yamil@"}, {"id":45,"firstname":"Web","lastname":"Services","username":"WS","address":"WS133","mail":"WS@gmail.com"}, {"id":46,"firstname":"Android","lastname":"Studio","username":"AS","address":"AS123","mail":"as@gmail.com"}]
```



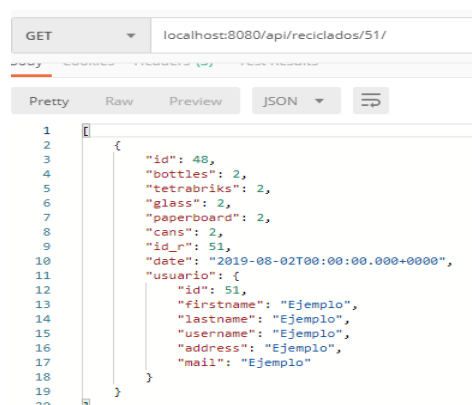
```
[{"id":1,"firstname":"Matias","lastname":"San Roman","username":"matt","address":"Pinto","mail":"matt@"}, {"id":2,"firstname":"Yamil","lastname":"Lacoste","username":"yamil_14","address":"25demayo","mail":"yamil@"}, {"id":45,"firstname":"Web","lastname":"Services","username":"WS","address":"WS133","mail":"WS@gmail.com"}, {"id":46,"firstname":"Android","lastname":"Studio","username":"AS","address":"AS123","mail":"as@gmail.com"}, {"id":51,"firstname":"Ejemplo","lastname":"Ejemplo","username":"Ejemplo","address":"Ejemplo","mail":"Ejemplo"}]
```

Reciclar: seguimos con la tercera actividad, cuenta con botones y textview simbolizando los elementos a reciclar. Se usa el Shared Preferences para guardar el estado de los botones en caso que el usuario todavía no quiera realizar la petición o si se cierra la app estos van a seguir cargados.



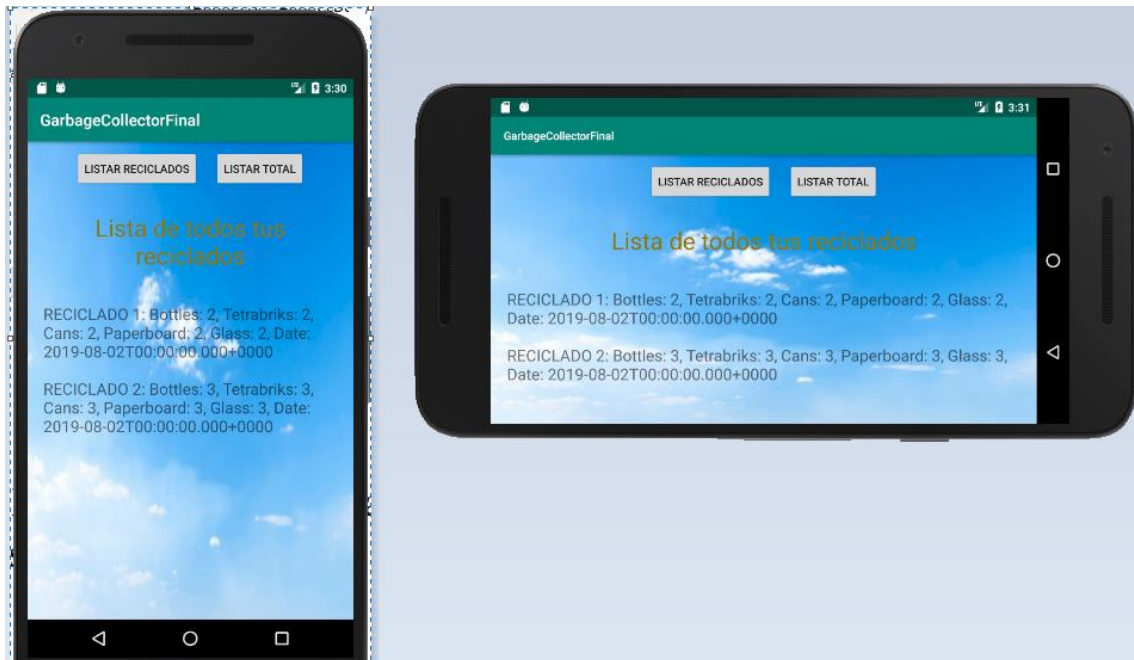
Como se puede apreciar, una vez mandada la petición, se registra el reciclado con el usuario asociado y, además, los botones se resetean a cero y se muestra un mensaje.

Desde Postman se puede ver el estado de la Base de Datos.

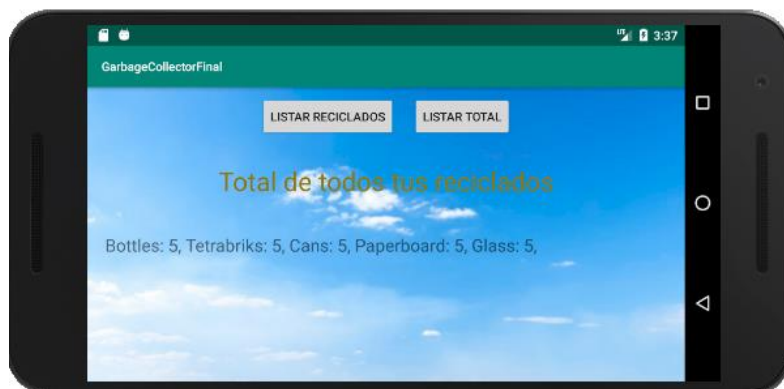


ListasReciclados: la última actividad, cuenta con dos botones, ambos hacen peticiones del tipo GET. Cuenta con un SCROLL por si la lista es grande y no entra en la pantalla.

Primer botón, lista los reciclados del usuario.

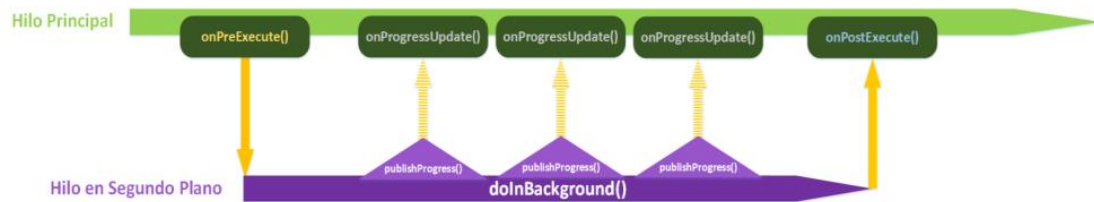


Se puede ver que por mas que se de vuelta la pantalla, no se pierden los valores. El siguiente botón es listar el total donde muestra la suma de todos los reciclados de ese usuario. La suma se hizo desde la API REST, aunque no es una operación compleja, la dejamos para que haga el trabajo al servidor.



Vamos a hablar sobre el uso de Thread para resolver las dos peticiones GET anteriores.

Se usó AsyncTask que sirve para realizar tareas en background.



Al crearse el objeto de la clase `AsyncTask` se llama, para empezar, su primer método `onPreExecute()` que se ejecuta sobre el hilo principal. Al terminar este `AsyncTask` crea un hilo secundario y ejecuta su trabajo dentro de `doInBackground()`.

Durante la ejecución en segundo plano podemos realizar llamadas al hilo principal desde `doInBackground()`, y con ayuda de `publishProgress()`, a un método sobrescrito llamado `onProgressUpdate()`.

No es obligatorio llamar a `publishProgress()`, y así tampoco llamaremos a `onProgressUpdate()`, o lo podemos llamar las veces que queramos.

Al terminar de ejecutarse `doInBackground()` se llama de inmediato a `onPostExecute()`, y se da por acabado a este hilo en segundo plano.

Para realizar las peticiones GET (listar reciclados) fue necesario crear un hilo en segundo plano para que trabaje con los datos recibidos, ya que si la lista es muy grande la aplicación no responderá y se cerrará.

Por eso se ejecutó el método ONRESPONSE en un hilo en segundo plano, mediante la clase AsyncTask para que la App responda en todo momento.

Por último, hablamos sobre el botón cerrar sección, se encarga de limpiar el archivo de Shared Preferences y finalizar la actividad.

```
} else if (id == R.id.cerrar_seccion) {  
    SharedPreferences preferencias = getSharedPreferences( name: "credenciales", Context.MODE_PRIVATE);  
    SharedPreferences.Editor editor = preferencias.edit();  
    editor.clear();  
    editor.commit();  
    finish();  
}
```

Conclusión:

Se resolvió el enunciado propuesto aplicando varios temas vistos en la cursada, como el uso de AsyncTask para tareas pesadas y poder ejecutarlas en segundo plano, también el uso de Shared Preferences para guardar estados. Utilizamos la librería Volley donde tuvimos que investigar como funciona y la facilidad de esta, para comunicar una Api Rest con una aplicación en este caso desarrollada en Android. También el uso de ScrollView para poder ver la información correctamente y el uso de un Navigation Drawer.

Se ha testeado la App tanto virtualmente desde el emulador de Android, como en un dispositivo físico y cumple con los requisitos pedidos de la catedra.