IoT & SDKs - Workshop Mobile

Bienvenidos al workshop mobile de Internet de las cosas y SDKs.

En este documento voy a contarles brevemente cómo podemos lograr lo que realicé en la Full Stack. Un simple, pero divertido, cazador de ofertas al estilo Pokemon Go.

El objetivo del proyecto es mostrar cómo podemos utilizar la cámara para mejorar la experiencia del usuario, interactuar con beacons de forma escalable a otras tecnologías más económicas y también como monetizar utilizando otras SDKs de pagos, por ejemplo, la de MercadoPago.

En este link (https://github.com/matiasgualino/fullstacktech_iot_workshop) podrán encontrar el proyecto inicial (starter) y el proyecto finalizado (finished) que conseguirán siguiendo el paso a paso que les detallo a continuación:

1. Creamos un proyecto en Android, vacío. También podrás abrir el starter del link.
2. Creamos una Activity llamada `CamaraActivity`.
3. En su xml debemos agregar lo siguiente:

```
<SurfaceView
   android:id="@+id/surfaceView"
   android:layout_width="match_parent"
   android:layout_height="match_parent" />

<ImageView
   android:id="@+id/deal_icon"
   android:layout_width="128dp"
   android:layout_height="128dp"
   android:layout_centerInParent="true" />


<ImageView
   android:id="@+id/hand_icon"
   android:layout_width="128dp"
   android:layout_height="128dp"
   android:layout_centerInParent="true"
   android:layout_alignParentBottom="true"
   android:src="@mipmap/hand_icon"/>
```

   Surface indica que vamos a trabajar con la cámara. Las otras dos imágenes hacen referencia a la oferta que queremos cazar y la otra a la “mano” que va a cazarlas.
4. Vamos a utilizar la cámara y bluetooth, en las aplicaciones Android debemos pedirle permiso al sistema operativo para poder utilizarlos, los cuales son aceptados por el usuario al descargar la aplicación en versiones antiguas, pero desde Android 6 podemos pedir estos permisos a demanda para que no sean una traba a la hora de conseguir descargas.

```
<uses-permission android:name="android.permission.CAMERA"/>
<uses-feature android:name="android.hardware.camera" />

<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
```

```
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
```

5. En la clase de la activity tenemos que hacer referencia a los elementos agregados en el xml de la activity.
   a. Imports

```
import android.app.Activity;
import android.content.ClipData;
import android.content.Intent;
import android.content.res.Resources;
import android.os.Bundle;
import android.hardware.Camera;
import android.view.DragEvent;
import android.view.MotionEvent;
import android.view.SurfaceHolder;
import android.view.SurfaceView;
import android.view.View;
import android.widget.ImageView;
import android.widget.Toast;
```

   b. Agregar en declaración de clase

```
implements SurfaceHolder.Callback
```

   c. Variables de instancia

```
Camera camera;
SurfaceView surfaceView;
SurfaceHolder surfaceHolder;
ImageView dealIcon;
ImageView handIcon;

private Activity mActivity;
```

   d. OnCreate()

```
mActivity = this;

surfaceView = (SurfaceView) findViewById(R.id.surfaceView);
dealIcon = (ImageView) findViewById(R.id.deal_icon);
handIcon = (ImageView)findViewById(R.id.hand_icon);

surfaceHolder = surfaceView.getHolder();
surfaceHolder.addCallback(this);
surfaceHolder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
```

   e. Implementar SurfaceHolder.Callback

```
public void surfaceChanged(SurfaceHolder holder, int format, int w, int h) {
   if (surfaceHolder.getSurface() == null) {
       return;
   }

   try {
       camera.stopPreview();
   } catch (Exception e) {
   }

   try {
       camera.setPreviewDisplay(surfaceHolder);
       camera.startPreview();
   } catch (Exception e) {

   }
}
```

```
        public void surfaceCreated(SurfaceHolder holder) {
            try {
                camera = Camera.open();
            } catch (RuntimeException e) {
                System.err.println(e);
                return;
            }

            try {
                camera.setPreviewDisplay(surfaceHolder);
                camera.startPreview();
            } catch (Exception e) {
                System.err.println(e);
                return;
            }
        }

        public void surfaceDestroyed(SurfaceHolder holder) {
            if (camera != null) {
                camera.stopPreview();
                camera.release();
                camera = null;
            }

        }
```

6. Agregamos beacons al proyecto. En el build.gradle, añadimos un repositorio específico para buscar las dependencias y luego las mismas.

```
repositories {
    maven {
        url 'https://dl.bintray.com/mreverter/maven'
    }
}

dependencies {
    compile 'me.benear:core:0.0.1-rc2'
    compile 'me.benear:beacons:0.0.1-rc2'
}
```

7. Cuál es el modelo que querés taggear? En este caso vamos a taggear promociones.

```
import me.benear.model.BeRangedResource;

public class MyTaggedDeal implements BeRangedResource {

    // Atributos de la clase
    private String name;
    private String iconName;
    private Double price;

    // Identificador
    private String tagId;

    private Double distance;

    public String getTagId() {
        return tagId;
    }
```

```
    public void setTagId(String tagId) {
        this.tagId = tagId;
    }

    public Double getPrice() {
        return price;
    }

    public void setPrice(Double price) {
        this.price = price;
    }

    public String getIconName() {
        return iconName;
    }

    public void setIconName(String iconName) {
        this.iconName = iconName;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    @Override
    public void setDistance(Double distance) {
        this.distance = distance;
    }

    @Override
    public Double getDistance() {
        return distance;
    }
}
```

8. Pero los datos de dónde los saco? Puede ser de un servicio, de intancias de objetos o mismo de un archivo .json como en este caso. Vamos a guardarlo en la carpeta assets con el nombre deals_database.json y el siguiente contenido:

```
[
  {
     "tag_id": "2f234454-cf6d-4a0f-adf2-f4911ba9ffa6_1_1",
     "name": "Hamburguesa 20% OFF",
     "icon_name": "hamburger_icon",
     "price": 50.0
  },
  {
     "tag_id": "0x2f234454f4911ba9ffa6_0x000000000001_",
     "name": "Cafe 50% OFF",
     "icon_name": "coffee_icon",
     "price": 20.0
  }
]
```

9. Necesitamos agregar variables de instancia propias para esta integración. Un cliente de beacons (el lector) y la oferta que vamos a estar leyendo mediante un beacon.

```
private BenearBeaconsClient mBeaconsClient;
private MyTaggedDeal myTaggedDeal;
```

10. Ahora tenemos que convertir el .json en instancia de MyTaggedDeal

```
List<MyTaggedDeal> taggedDeals =
JsonUtil.getInstance().listFromJson(FileUtils.getStringFromAsset(this, "deals_database.json"),
MyTaggedDeal.class);
```

11. Creá una base de datos offline y relacioná a cada objeto con su respectivo tagId. Si! TagId porque aún no hablamos de tecnologías y esta puede ser beacons, qr, nfc, geolocation, realidad aumentada, zigbee, e incluso conectarse con Arduino!

```
BenearOffline.DatabaseBuilder databaseBuilder = new BenearOffline.DatabaseBuilder();
for(MyTaggedDeal currentTaggedDeals : taggedDeals) {
   databaseBuilder.addTaggedResource(currentTaggedDeals.getTagId(), currentTaggedDeals);
}

BenearOffline.Database myLocalDatabase = databaseBuilder.build();
```

12. Iniciamos un provider al cual le indicamos cuál es la base de datos que vamos a usar y quiénes somos (public_key).

```
BenearOffline offlineProvider = new BenearOffline.Builder()
       .setContext(this)
       .setPublicKey("PK-12331-321321-231")
       .setDatabase(myLocalDatabase)
       .build();
```

13. Ahora iniciamos el cliente encargado de leer beacons:

```
mBeaconsClient = new BenearBeaconsClient.Builder()
       .setContext(this)
       .addOnDetectionCallbackForType(MyTaggedDeal.class, new
OnDetectionCallback<MyTaggedDeal>() {
           @Override
           public void onDetected(List<MyTaggedDeal> deals) {
               resolveDealsFound(deals);
           }
       })
       .setForegroundScanPeriod(3000L)
       .setBenearProvider(offlineProvider)
       .build();
mBeaconsClient.startScanning();
```

14. Como verás resolveDealsFound es la función que recibe los tags leídos convertidos en objetos propios de tu modelo. Permitimos múltiples addOnDetectionCallbackForType de modo tal que puedas reconocer diferentes tipos para los mismos tags! Entonces que vamos a hacer cuando leamos las ofertas? Vamos a representarla con un ícono en el centro de la pantalla y activar un ícono para atrapar la oferta.

```
public void resolveDealsFound(List<MyTaggedDeal> deals) {
   MyTaggedDeal newDeal = deals.get(0);
   Log.d("DALE CHE", "Algo encontré = " + newDeal.getTagId());
   if (myTaggedDeal == null || !newDeal.getTagId().equals(myTaggedDeal.getTagId())) {
       myTaggedDeal = newDeal;
   }
   Resources r = getResources();
       final int resourceId = r.getIdentifier(myTaggedDeal.getIconName(), "mipmap",
getPackageName());

   runOnUiThread(new Runnable() {
       @Override
       public void run() {
           dealIcon.setImageResource(resourceId);
```

```
        }
    });

    mBeaconsClient.startScanning();

}
```

15. Ahora tenemos que hacer el efecto de "atrapar". Para eso necesitamos poner al ícono de la oferta como objetivo y el ícono del cazador móvil para validar la superposición. Agregar lo siguiente en onCreate():

```
dealIcon.setOnDragListener(new ContainerDragListener());
handIcon.setOnTouchListener(new ImgTouchListener());
```

16. Agregamos los listener:
    a. ImgTouch es para mover la mano

```
private class ImgTouchListener implements View.OnTouchListener {
    public boolean onTouch(View view, MotionEvent motionEvent) {
        if (motionEvent.getAction() == MotionEvent.ACTION_DOWN) {
            ClipData data = ClipData.newPlainText("", "");
            View.DragShadowBuilder shadowBuilder = new
View.DragShadowBuilder(view);
            view.startDrag(data, shadowBuilder, view, 0);
            view.setVisibility(View.INVISIBLE);
            return true;
        } else {
            return true;
        }
    }
}
```

    b. ContainerDrag es para setear a la imagen de la oferta como objetivo.

```
private class ContainerDragListener implements View.OnDragListener {
    @Override
    public boolean onDrag(View v, DragEvent event) {
        int action = event.getAction();
        switch (event.getAction()) {
            case DragEvent.ACTION_DROP:
                if (myTaggedDeal != null) {
                    // Y ahora? Que hacemos?
                }
            default:
                break;
        }
        return true;
    }
}
```

17. Y ahora? Que hacemos? Vamos a monetizar el negocio! Mirá qué rápido:
    a. Agregamos la dependencia de MercadoPago

```
compile ('com.mercadopago:sdk:2.1.0-rc5@aar') { transitive = true }
```

    b. Llamamos a MercadoPago con nuestros colores, indicando quiénes somos, cuánto queremos cobrar el producto (el de la promoción taggeada!) e iniciamos el flujo:

```
DecorationPreference decorationPreference = new DecorationPreference();
decorationPreference.setBaseColor("#FFCC3E");
decorationPreference.enableDarkFont();

new MercadoPago.StartActivityBuilder()
        .setActivity(mActivity)
```

```
        .setPublicKey("TEST-d941de2c-22bd-4e9e-a06c-48a404b66080")
        .setAmount(new BigDecimal(myTaggedDeal.getPrice()))
        .setSite(Sites.ARGENTINA)
        .setInstallmentsEnabled(true)
        .setDecorationPreference(decorationPreference)
        .startCardVaultActivity();
```

c. Esperamos los resultados en onActivityResult

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {

   if (requestCode == MercadoPago.CARD_VAULT_REQUEST_CODE) {
       if (resultCode == RESULT_OK && data != null) {

           // Obtener os dados inseridos pelo usuario

           PaymentMethod paymentMethod =
JsonUtil.getInstance().fromJson(data.getStringExtra("paymentMethod"),
PaymentMethod.class);
           Issuer issuer =
JsonUtil.getInstance().fromJson(data.getStringExtra("issuer"), Issuer.class);
           Token token =
JsonUtil.getInstance().fromJson(data.getStringExtra("token"), Token.class);
           PayerCost payerCost =
JsonUtil.getInstance().fromJson(data.getStringExtra("payerCost"),
PayerCost.class);

           String message = "PaymentMethod: " + paymentMethod.getId();

           if (issuer != null) {
               message += " - Issuer: " + issuer.getName();
           }

           if (payerCost != null) {
               message += " - PayerCost: " + payerCost.getInstallments();
           }

           if (token != null) {
               message += " - Token: " + token.getId();
           }

           Toast.makeText(getApplicationContext(), message,
                   Toast.LENGTH_LONG).show();
       } else {

           if ((data != null) &&
                   (data.getSerializableExtra("apiException") != null)) {
               ApiException apiException =
                       (ApiException) data.getSerializableExtra("apiException");

               Toast.makeText(getApplicationContext(), apiException.getMessage(),
                       Toast.LENGTH_LONG).show();
           }
       }
   }
}
```

d. Y listo! Ahora deberíamos enviar los datos a nuestros servidores para finalizar el pago, pero ahora es tu turno! Seguí estos pasos acá:
https://www.mercadopago.com.ar/developers/en/tools/sdk/server/node-js/