Manual

SDK Android

VALIDACIÓN QRs

Plataforma Android

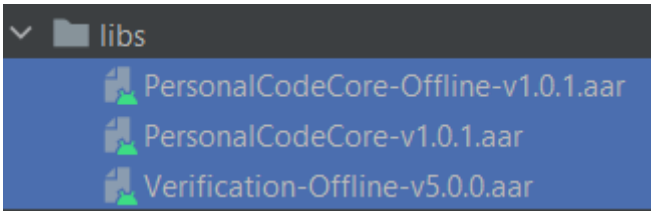
Versión 5.0.0

Manual de usuario para el programador Android de la librería Personal Code Offline

1 - Importación del SDK al proyecto

1.1 - Copiar los siguientes archivos al proyecto dentro de la carpeta libs:

- *PersonalCodeCore-v[VERSIÓN].aar*
- *PersonalCodeCore-Offline-v[VERSIÓN].aar*
- *Verification-Offline-v[VERSIÓN].aar*



1.2 - Agregar las líneas de implementación con sus respectivas versiones en el archivo build.gradle (Módulo app)

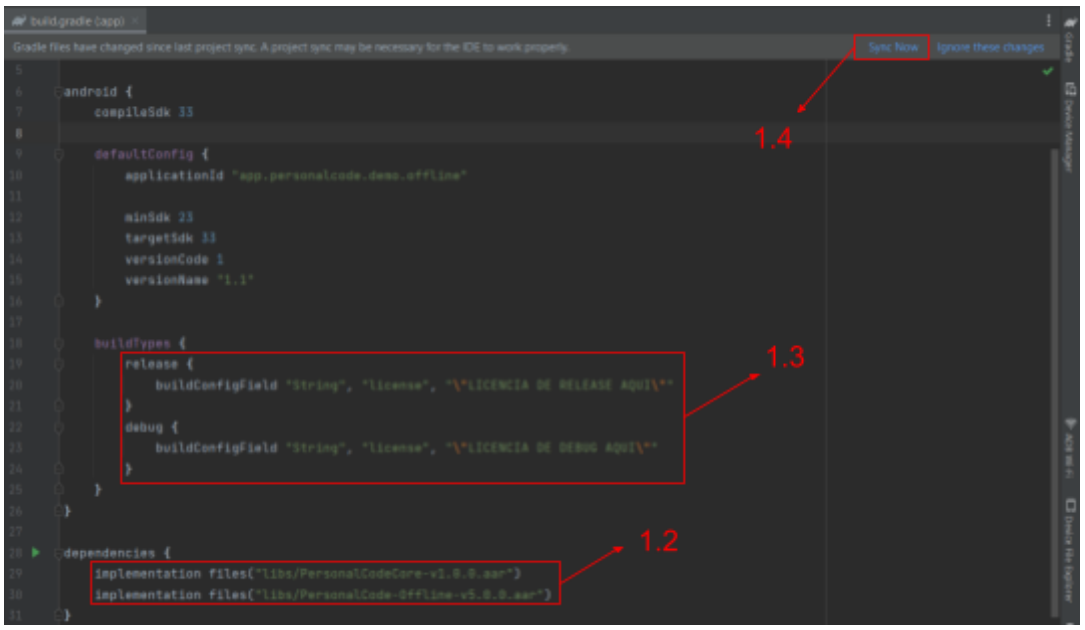
1.3 - Declarar las licencias referentes a Debug y Release en Gradle con las siguientes líneas de código:

```
buildTypes {
    release {
        buildConfigField "String", "license", "\"Android|RELEASE_LICENSE\""
    }
    debug {
        buildConfigField "String", "license", "\"Android|DEBUG_LICENSE\""
    }
}
```

Las licencias que proveemos son firmadas y se compilan junto con la aplicación en su respectivo entorno. Es decir que una vez que se compila en modo Release, la apk no llevará la licencia de Debug y vice-versa. Al ser firmadas las licencias digitalmente, no se podrá cambiar su contenido, por lo que al necesitar algún cambio en la licencia se deberá solicitar dichos cambios directamente al proveedor.

Dichas licencias van vinculadas al applicationId y al SHA1 de compilación de la apk, por lo que es de extrema importancia que se resguarden las llaves de compilación .jks, así como su keystore de Android Studio, de forma que solo los desarrolladores autorizados en las mismas puedan utilizar su licencia.

1.4 - Sincronizar el proyecto



2 - Implementación del SDK al proyecto

2.1 - Importar las siguientes clases a la clase que implementará el SDK:

```
import android.app.Activity;
import android.graphics.Bitmap;
import android.view.ViewGroup;
import com.personalcode.PersonalCodeInterface;
import com.personalcode.PersonalCodeOffline;
import org.json.JSONObject;
```

2.2 - Inicializar una instancia de *PersonalCodeInterface*, donde el SDK notificará sus eventos.

Los métodos *onCaptured* y *onClose* cuentan con el tipo de retorno *boolean*, donde se podrá solicitar el cierre de la actividad de Scanner mediante el valor true una vez obtenidos los datos, o simplemente conservarlo abierto con el valor false. En el caso de que se decida conservar abierto, se podrá realizar lo necesario posteriormente mediante el parámetro *activity* recibido en los callbacks.

```
public final PersonalCodeInterface personalCodeInterface = new PersonalCodeInterface() {
    @Override
    public boolean onCaptured(Activity activity, String txt, Bitmap img) {
        notificarValidos(activity, txt, img);
        return true;
    }

    @Override
    public void onOpened(Activity activity, ViewGroup rootView) {

    }

    @Override
    public boolean onClose(Activity activity, int error, String description) {
        if (error<100) {
            notificarInvalidos(activity, description);
            //context.startActivity(new Intent(context, Principal.class));
        }
        return true;
    }
};
```

2.2 - Inicializar el SDK mediante PersonalCodeOffline.getInstance(), donde enviamos la instancia de PersonalCodeInterface que se generó anteriormente.

```
public PersonalCodeOffline personalCodeOffline;
public Validacion(Activity actividad) {
    if (personalCodeOffline==null) {
        personalCodeOffline = PersonalCodeOffline.getInstance(actividad, personalCodeInterface);
    }
}
```

2.3 - Validar y solicitar permiso de cámara al usuario.
2.4 - Iniciar el scanner llamando el método *personalCodeOffline.capture(activity, false)*, donde:

- *Activity* es la instancia de la actividad actual de la aplicación
- Y el parámetro booleano solicita al SDK que cierre la actividad actual al abrir el scanner o no (true la cierra y false no hará nada)

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.layout_principal);

    Button btnScan = findViewById(R.id.btnScan);
    btnScan.setOnClickListener(v -> {
        if (tienePermisoDeCamara( actividad: this)) {
            validacion.personalCodeOffline.capture( activity: this,  b: false);
        } else {
            pedirPermisoDeCamara( actividad: this);
        }
    });
}
```

2.5 - Una vez hecho esto, basta con utilizar el SDK para escanear los códigos QR y finalmente recibir la información extraída en el método *onCaptured()* de la instancia del interfaz *PersonalCodeInterface* declarada en el paso 2.2 de éste documento para que se realice lo necesario con dicha información.

Los resultados de la extracción serán retornados en:
Imagen: una instancia de *Bitmap*.
JSON: una instancia de *JSONObject* conteniendo los datos biográficos extraídos de la credencial con las siguientes etiquetas:

- "tipo": "string",
- "cic": "string",
- “ocr": "string",
- “curp": "string",
- “nombre": "string",
- “apellido1": "string",
- “apellido2": "string",
- “entidad": "string",
- “municipio": "string",
- “seccion": "string",
- “vigencia": "string",
- “sexo": "string",
- “version": "string"

2.6 - *Descripción de los datos extraídos:*

NACIONAL

01.- tipo credencial N (Nacional)
02.- cic
03.- ocr
04.- curp
05.- nombre
06.- apellido paterno
07.- apellido materno
08.- entidad

EXTRANJERO

01.- tipo credencial E (Extranjero)
02.- cic
03.- ocr
04.- curp
05.- nombre
06.- apellido paterno
07.- apellido materno
08.- entidad

09.- municipio
10.- sección
11.- etnia
12.- vigencia (año de emisión - año de vigencia)
13.- sexo
14.- número de dedo huella 1
15.- número de dedo huella 2 (0)
16.- versión CPV (G)
17.- fecha generación lote
18.- firma verificadora

09.- vigencia (año de emisión - año de vigencia)
10.- sexo
11.- número de dedo huella 1
12.- número de dedo huella 2 (0)
13.- versión CPV(H)
14.- fecha generación lote
15.- firma verificadora

3 - Personalización de la vista del Scanner

Para la personalización del scanner contamos con los siguientes atributos públicos en la clase *PersonalCodeOffline* que pueden ser modificados según la necesidad al instanciar la clase (punto 2.2 del presente documento):

```
public String personalCodeLangQRs = "Enfoca los códigos QR";
public String personalCodeColorLine = "#FFFFFF";
public Typeface personalCodeFont = null;
public Bitmap imgFlashOn;
public Bitmap imgFlashOff;
```

3.1 - Botón de Flash

Caso se necesite modificar la ubicación o el aspecto del botón de flash, se podrá obtener la instancia de la actividad *ScannerActivity* mediante su método *ScannerActivity.getInstance()* dentro del callback *onOpened()* de la interfaz *PersonalCodeInterface* creada en el punto 2.2 de éste documento, donde tiene el atributo público *btnFlash* del tipo *Button*.

Mediante la instancia de éste atributo será posible cambiar aspecto, tamaño y ubicación del botón.

3.2 - Otras personalizaciones

Para agregar logo y personalizar más a fondo la vista de *ScannerActivity*, en el callback *onOpened()* de la interfaz *PersonalCodeInterface* se obtiene el parámetro *rootView* del tipo *ViewGroup*, donde se podrá implementar nuevas vistas con solo algunas pocas líneas de código.