Manual de Integración de Canales BIOmetric ID

Mayo 2024

Somos especial COSTA RICA

Hagamos el futuro juntos

racsa

Table of Contents

1) Biometric ID - API

En este documento encontrará una guía de pasos para realizar la integración con el API Biometric ID.

Prerrequisitos

Para hacer uso del API debe contar con un API Key válido.

Toda petición que se realice al api debe contener el siguiente header

```
ApiKey: {{ apiKey }}
```

> **Nota**
>
> Según el endpoint que se use se solicitarán headers adicionales.

Flujo de uso

El api está diseñado para utilizarse de manera secuencial, el flujo es el siguiente:

Endpoints expuestos

DocumentQuery

Este endpoint permite consultar si un documento está disponible para su verificación biométrica.

Headers requeridos:

```
ApiKey: {{ apiKey }}
Authorization: ApiKey {{ apiKey }}
Content-Type: application/json
```

Ejemplo de request:

```
curl --location --request POST 'https://apibiometricid.racsa.go.cr/bioapi/api/DocumentQuery' \
--header 'Content-Type: application/json' \
--header 'Accept: application/json' \
--header 'ApiKey: {{ APIKEYVALID }}' \
--header 'Authorization: ApiKey {{ APIKEYVALID }}' \
--data-raw '{
  "documentType": "Foreigner",
  "documentNumber": "1234566"
}'
```

Ejemplo de respuesta para un documento disponible:

```
{
    "biometricAvailable": true,
    "token": "{{ JWTVALIDTOKEN }}",
    "availableFingers": [
        "RightThumb",
        "RightIndex"
    ],
    "facialRecognitionAvailable": false,
    "name": "Pepito",
    "surname": "Peréz",
    "secondSurname": "HIT",
    "transactionId": "c3717a3c-32d5-b62b-b5d4-3a092a3dcb84"
}
```

Nota

En este caso la respuesta nos entrega un token que será requerido para hacer la verificación biométrica

Ejemplo de respuesta para un documento no disponible:

```
{
    "transactionId": "ab57808b-b6cd-c8db-5485-3a092a1ffc54",
    "biometricAvailable": false
}
```

BiometricVerification

Este endpoint permite realizar la verificación biométrica de una persona dada la imagen de su rostro o la imagen de una huella en formato wsq.

Headers requeridos:

```
ApiKey: {{ apiKey }}
Authorization: Bearer {{ JWTVALIDTOKEN }}
Content-Type: application/json
```

JWTVALIDTOKEN

Este token es entregado por la respuesta de DocumentQuery

Ejemplo de request verificación por huella:

```
curl --location --request POST
'https://apibiometricid.racsa.go.cr/bioapi/api/BiometricVerification' \
--header 'Content-Type: application/json' \
--header 'Accept: application/json' \
--header 'Authorization: Bearer {{ JWTVALIDTOKEN }}' \
--header 'ApiKey: {{ APIKEYVALID }}' \
--data-raw '{
  "finger": "RightThumb",
  "fingerPrint": "BASE64ENCODEDWSQFINGER"
}'
```

Ejemplo de respuesta para una verificación exitosa:

```
{
    "hit": true,
    "documentImage": "BASE64ENCODEDDOCUMENTIMAGE",
    "documentExpirationDate": "2024-10-25T00:00:00",
    "token": "JWTVALIDTOKEN",
    "externalDataSources": [],
    "transactionId": "c3717a3c-32d5-b62b-b5d4-3a092a3dcb84",
    "remainingTries": 1,
    "scoreResult": 0,
    "scoreLimit": 0
}
```

Nota

En este caso la respuesta nos entrega un token que será requerido para hacer la verificación biométrica

Ejemplo de respuesta para una verificación fallida:

```
{
    "transactionId": "c3717a3c-32d5-b62b-b5d4-3a092a3dcb84",
    "remainingTries": 1,
    "hit": false,
    "scoreResult": 0,
    "scoreLimit": 0
}
```

Más información:

Puede ver la documentación actualizada del enpoint en el swagger

2) Biometric ID - SDK Android

En este documento encontrará una explicación de cómo se realiza la integración con el SDK Android BiometricID

Pre requisitos

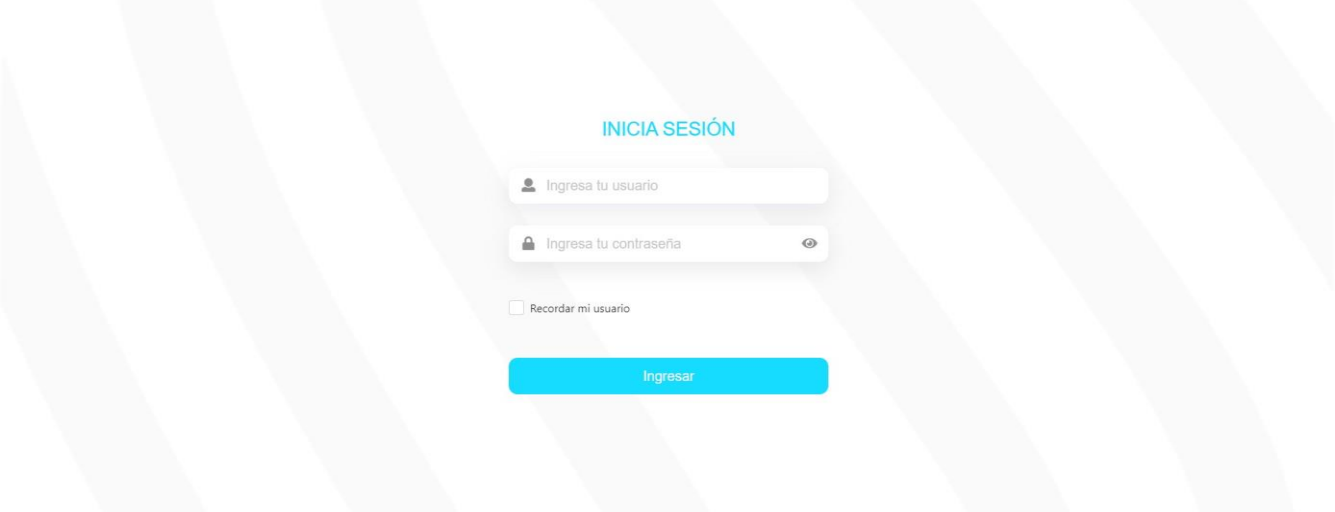
Para hacer uso del SDK se debe contar con

```
ApiKey válido: {{ API_KEY }}
Usuario Maven: {{ USUARIO_MAVEN }}
Password Maven: {{ PASSWORD_MAVEN }}
```

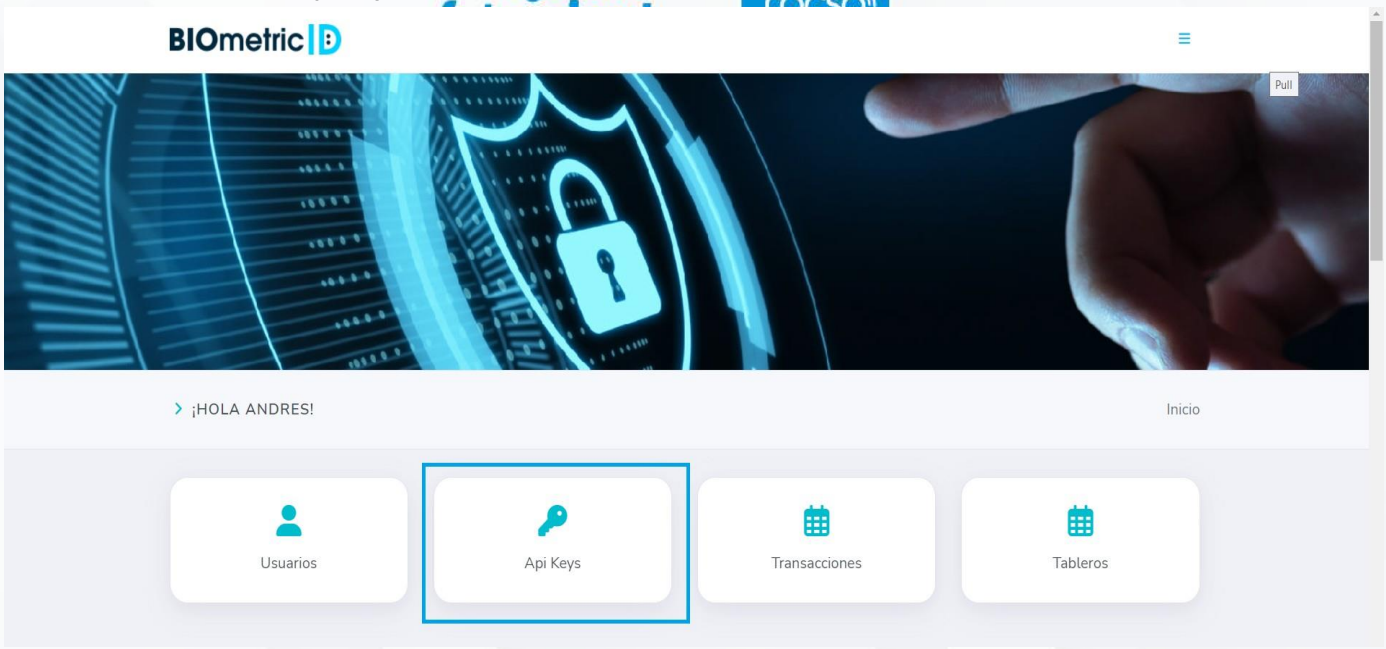
Crear un Api Key

Para usar el sdk se necesita tener un api key válido, en caso de que tengas uno omite este paso

1. Inicia sesión en el selfservice con tus credenciales

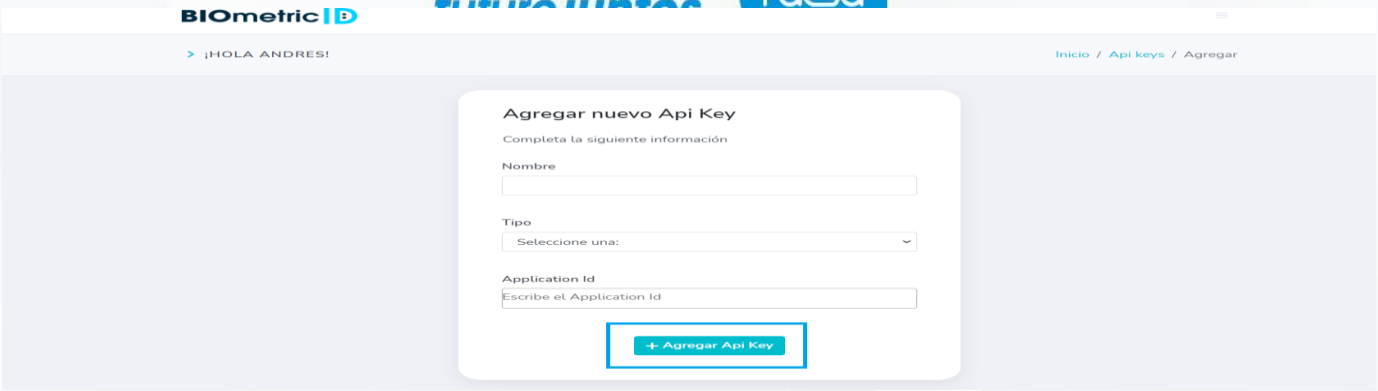


2. Da click en la sección de Api Keys



3. Da click agregar apikey

4. Ingresa un nombre y selecciona el tipo sdk android por último escribe el application Id de tu proyecto



Instalación

Instalacion del SDK

1 . Vamos a settings.gradle y en la sección de repositories añadimos la sección de maven:

```
dependencyResolutionManagement {

repositories {
    maven {
        url 'https://w2dit.pkgs.visualstudio.com/4eb99ce3-afda-4374-8793-
7a1ff2f6a414/_packaging/IntegraIDArtifacts/maven/v1'
        name 'com.getgroup.integraidsdk'
        credentials {
            username "{{ USUARIO_MAVEN }}"
            password "{{ PASSWORD_MAVEN }}"
        }
        authentication {
            basic(BasicAuthentication)
        }
    }
}
```

2 . En el build.gradle de la app en la sección de android colocamos el minSdk con un valor minimo de 22 y en compileSdk y targetSdk 33

```
android {
    compileSdk 33
        defaultConfig {
        minSdk 22
        targetSdk 33
    }
}
```

3 . En el build.gradle de la app en la sección de android añadimos la sección packagingOptions con lo siguiente

```
android {
    packagingOptions {
        pickFirst 'lib/x86/libc++_shared.so'
        pickFirst 'lib/x86_64/libc++_shared.so'
        pickFirst 'lib/armeabi-v7a/libc++_shared.so'
        pickFirst 'lib/arm64-v8a/libc++_shared.so'
        pickFirst 'lib/armeabi-v7a/libopencv_java4.so'
        pickFirst 'lib/arm64-v8a/libopencv_java4.so'
    }
}
```

4 . En el build.gradle de la app en la sección de dependencies añadimos la dependencia del sdk

```
dependencies {
    //sdk
    implementation(group: 'com.getgroup.integraidsdk', name: 'integraid-sdk', version: '1.0.89')
}
```

Instalación dagger

El sdk necesita instalar dagger para trabajar con inyección de dependencias

1 . En el build.gradle de la app en la sección dependencies añadimos

```
dependencies {
    //inyección de dependencias
    implementation "com.google.dagger:hilt-android:2.42"
    kapt "com.google.dagger:hilt-compiler:2.42"
    kapt "androidx.hilt:hilt-compiler:1.0.0"
}
```

2 . En el build.gradle de la app en la sección plugins añadimos

```
plugins {
    //inyeccion de dependencias
    id 'kotlin-kapt'
    id 'dagger.hilt.android.plugin'
}
```

3 . En el build.gradle de la app en la sección defaultConfig añadimos

```
android {
    defaultConfig {
        //inyeccion de dependencias

javaCompileOptions.annotationProcessorOptions.arguments['dagger.hilt.disableCrossCompilationRootValid
= 'true'
    }
}
```

4 . En el settings.gradle en la sección pluginManagement

```
pluginManagement{
    //inyeccion de dependencias
    resolutionStrategy {
        eachPlugin {
            if( requested.id.id == 'dagger.hilt.android.plugin') {
                useModule("com.google.dagger:hilt-android-gradle-plugin:2.39.1")
            }
        }
    }
}
```

Inyectar la configuración al sdk

1 . Creamos una clase llamada App que hereda de ApplicationBase y vamos a sobreescribir el metodo getSdkConfigModule

```
import com.getgroup.integraidsdk.ApplicationBase
import com.getgroup.integraidsdk.Domain.SdkConfig.SdkConfigModule
import dagger.hilt.android.HiltAndroidApp

@HiltAndroidApp
class App : ApplicationBase() {

   //para sobreescribir la configuración
   override fun getSdkConfigModule(): SdkConfigModule {
       return AppConfigModule(this)
   }
}
```

2 . Creamos la clase AppConfigModule

```
import android.app.Application
import com.getgroup.integraidsdk.Domain.SdkConfig.SdkConfig
import com.getgroup.integraidsdk.Domain.SdkConfig.SdkConfigModule

class AppConfigModule(private val context: Application) : SdkConfigModule() {
   override fun provideSdkConfig(): SdkConfig {
       return SdkConfig(
           UrlBaseApi = "{URL_API}",
           ShowScore = false,
           IsDebug = false,
           TimeOutApi = 20 * 1000,
           TimeOutMostrarDocumento = 5 * 60 * 1000,
           ApiKey = "{API_KEY}",
           WriteLogsConsola = false,
           WriteLogsDispositivo = false,
           UrlTerminosCondiciones = "",
           LivenessEscaneoFacial = 0.8,
           ScoreMinimoEscaneoFacial = 40f,
           ScoreMinimoEscaneoHuella = 40,
       )
   }
}
```

3 . Vamos al manifest y asociamos la clase de la app como el contexto en la sección de application

```
<application
    android:name=".App"
    tools:replace="android:name,android:theme"
...
```

Definiciones

Variables de configuración (SdkConfig)

Nombre	Tipo	Descripción
UrlBaseApi	String	Url del api https://apibiometricid.racsa.go.cr/bioapi
ShowScore	bool	Si esta en true cuando falle la verificación biometrica mostrará el score o puntaje
IsDebug	bool	Para definir si la app esta en debug
TimeOutApi	int	Asigna el tiempo máximo de espera en la conexión con el api, el valor recomendado es de 20000 milisegundos oséa 20 segundos
TimeOutMostrarDocumento	long	Asigna el tiempo máximo que permitirá mostrar la información del documento consultado en InformacionPersonalView

Nombre	Tipo	Descripción
ApiKey	string	El token generado por el self service que tiene asociado el applicationId del proyecto, el apikey permite realizar peticiones al api
WriteLogsConsola	bool	Si está en true escribirá logs en la consola
WriteLogsDispositivo	bool	si está en true escribirá logs en el dispositivo, sin embargo se deberán declarar permisos de lectura y escritura en el manifest
UrlTerminosCondiciones	string	Asigna la url de terminos y condiciones que se mostrará al usuario cuando realice una consulta de documento
LivenessEscaneoFacial	double	Valor entre 0 y 1 entre más alto el número verificará que el escaneo se le tome a una persona para evitar fraudes, el valor recomendado es 0.8
ScoreMinimoEscaneoFacial	float	Valor entre 0f y 100f, es el score o puntuación mínima local de rostro que realiza el sdk antes de enviarlo al api
ScoreMinimoEscaneoHuella	int	Valor entre 0 y 100, es el score o puntuación mínima local de huellas que realiza el sdk antes de enviarlo al api

Clases

Nombre	Descripción	Propiedades
Documento	Clase que muestra la información básica de un documento consultado	`Nombre: String, Apellido: String, SegundoApellido: String, Huellas: List<HuellaEscanearEnum?>, IsVerificacionFacial: Boolean, TipoDocumentoEnum:TipoDocumentoEnum, TransactionId:String`
TipoDocumentoEnum	Enumerador con el tipo de documento	`NACIONAL, EXTRANJERO`
ResponseException	clase que trae las excepciones internas y del api	`code: Int, error: String, response: Any?,, responseBody: String?, transactionId: String, extras: Any?`
HuellaEscanearEnum	Enumerador con los tipos de huella para escanear	`PULGAR_DERECHO = 1, PULGAR_IZQUIERDO = 6, INDICE_DERECHO= 2, INDICE_IZQUIERDO = 7,`

Nombre	Descripción	Propiedades
		`MEDIO_DERECHO = 3, MEDIO_IZQUIERDO = 8, ANULAR_DERECHO = 4, ANULAR_IZQUIERDO= 9, MENIQUE_DERECHO = 5, MENIQUE_IZQUIERDO = 10`
VerificarFacialHuella	clase para realizar la verificación biometrica (facial o huella)	`Token: String, Documento: String, NombreCompleto: String, Huella: HuellaEscanearEnum, PathHuellaWsq: String, PathFoto: String`
InformacionPersonal	clase de información personal de un documento consultado y que haya pasado la verificación biometrica	`TransactionId: String, ImagenDocumento: String, FechaExpiracion: String, NombreCompleto: String, Documento: String, ListFuentesExternas:List<FuenteExterna>`
FuenteExterna	clase de fuente externa disponible para consultar	`Id: Int, Nombre: String, Seleccionada: Boolean`

Códigos de error

Cada variable se encuentra en la clase ApiConstants

Nombre	Código	Explicación
RESPONSE_OK	200	Todo fue correcto
RESPONSE_NOT_FOUND	404	No encontró el documento consultado
RESPONSE_ERROR_VALIDATIONS	400	Errores de validación, por ejemplo no se están enviando datos con el formato correcto
RESPONSE_INTERNAL_ERROR	500	Error interno del servidor
RESPONSE_UNAUTHORIZARED	401	Petición sin autorización, el token de consulta de documento expiro ó se esta enviando un Api Key sin autorización
RESPONSE_UNAUTHORIZARED_FORBIDDEN	403	Se esta enviando un Api Key inválido
RESPONSE_IMAGES_NOT_FOUND	512	Error local de la aplicación que ocurre cuando se perdió el cache de la verificación facial o huella, se debe realizar una nueva verificación

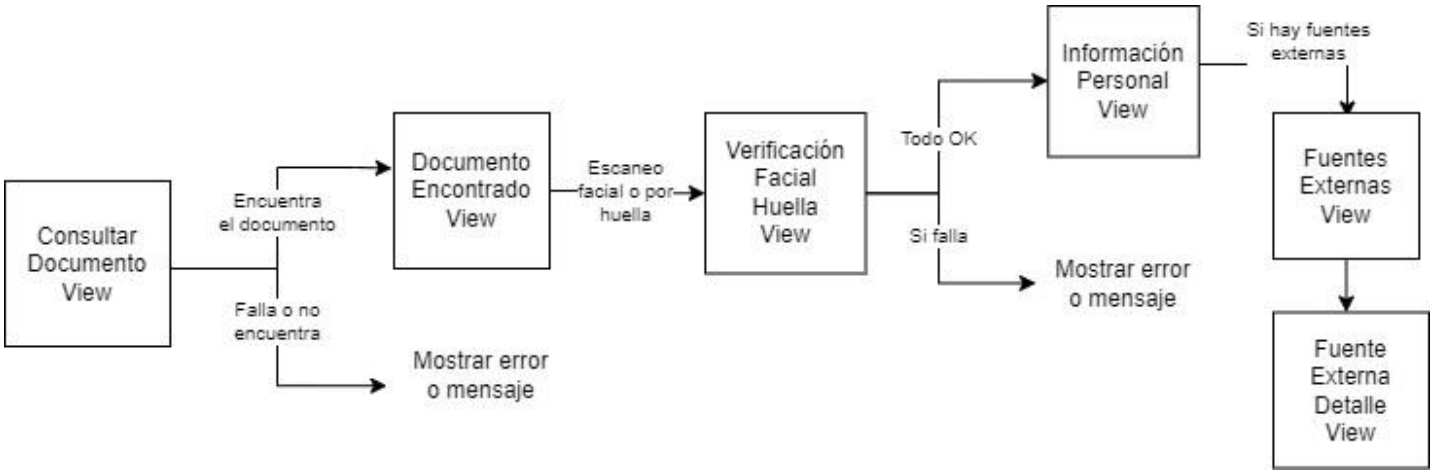
Permisos

El sdk ya tiene definidos los permisos y no es necesario declararlos en el manifest de tu proyecto sin embargo a continuación se muestran

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

Diagrama de navegación

Cada interfaz (activity) debe usar un view que se encarga de la logica de negocio, a continuación el flujo de los activities por defecto con el nombre del view que utliza



Activities por defecto

Para invocar el flujo de navegación por defecto debes hacer un intent a ConsultarDocumentoActivity

```
val intent = Intent(this, ConsultarDocumentoActivity::class.java)
startActivity(intent)
```

Crea tus activities

Activity

Cada activity debe heredar de BaseActivity y arriba de la clase colocar @AndroidEntryPoint para que recibir las dependencias del sdk por ejemplo:

```
@AndroidEntryPoint
class CustomActivity : BaseActivity() {
}
```

xml

En el xml del activity se debe de usar el view deseado

```
<com.getgroup.integraidsdk.Ui.ViewDeseado
   android:id="@+id/identificador"
   android:layout_width="match_parent"
   android:layout_height="wrap_content" />
```

Evento

Cada view tiene una función init donde se le pasa un evento que te notificará para que puedas tomar acción, por ejemplo:

```
private val event = object : Evento{
   override fun onEvento(): Boolean {
       return super.onEvento()
   }
}
```

Consultar documento

Este view permite realizar la consulta de un documento especificando el tipo de documento y aceptando los términos y condiciones

paquete: `com.getgroup.integraidsdk.Ui.ConsultarDocumento.ConsultarDocumentoView`

Método	Descripción
`init(event: ConsultarDocumentoViewEvent?)`	Se llama para inicializar el view, se le pasa el evento
`OnActivityResult(requestCode: Int, resultCode: Int, data: Intent?)`	Se debe llamar en el onActivityResult del activity donde se usa el view, pasandole los mismos argumentos
`mostrarError(ex: ResponseException?)`	Permite mostrar un error enviando una instancia de tipo `ResponseException`
`mostrarError(text: String)`	Permite mostrar un error enviando una cadena de texto

ConsultarDocumentoViewEvent

Método	Descripción
`onLoader(isVisible: Boolean)`	Método que notifica que si se debe o no mostrar un loader
`onResult(result: Result<Documento?`	Método que traerá un resultado de Documento o un error. Para saber si todo fue correcto preguntar `result.isSuccess`, para obtener el documento `result.getOrNull()`, para

Método	Descripción
`>)`	obtener el error `val error = result.exceptionOrNull() as ResponseException`
`onEndShowError(code: Int)`	Método que notifica una vez se haya terminado de mostrar un error desde el view `mConsultarDocumentoView.mostrarError(error)`

Documento encontrado

Este view muestra el documento encontrado y las opciones para la verificación facial o huella

paquete: `com.getgroup.integraidsdk.Ui.DocumentoEncontrado.DocumentoEncontradoView`

Método	Descripción
`init(context: Activity, model: Documento, event: DocumentoEncontradoEvent?)`	inicializa el view, se le debe enviar el activity, un objeto de tipo Documento y el evento de tipo `DocumentoEncontradoEvent`
OnActivityResult(requestCode: Int, resultCode: Int, data: Intent?)	Se debe llamar en el onActivityResult del activity enviando los mismos argumentos

DocumentoEncontradoEvent

Método	Descripción
onEscaneoFacial(): Boolean	Cuando se le da click a la verificación facial llega aquí, si se retorna el metodo `super().onEscaneoFacial()` entonces continua, en caso de retornar false no permite continuar con el escaneo facial
onEscaneoHuella(huella: HuellaEscanearEnum?): Boolean	Al dar click para escanear una huella llega aquí y entrega la huella que va a escanear, si `return super().onEscaneoHuella()` continua si return false entonces no realiza el escaneo
onVerificarEscaneoFacial(model: VerificarFacialHuella): Boolean	una vez realiza el escaneo facial o por huella si pasa el score mínimo local que se le ha configurado al sdk llega aquí, entrega un objeto para realizar la verificación biométrica, para realizar la verificación con una activity propia se retorna false
onShowMensajes(mensaje: String)	cada vez que el view necesite mostrar un mensaje llega aquí una cadena que se puede mostrar con un Toast por ejemplo

Verificación biométrica (facial o huella)

Es un view que se comunica con el api enviando la foto o huella y retorna la información personal con las fuentes externas disponibles

paquete: `com.getgroup.integraidsdk.Ui.VerificacionFacialHuella.VerificacionFacialHuellaView`

Método	Descripción
init(event: VerificacionFacialHuellaEvent?)	para inicializar el view, se le debe enviar un evento de tipo VerificacionFacialHuellaEvent
verificar(model: VerificarFacialHuella)	para realizar la verificación biometrica (por huella o facial), se le debe enviar el objeto que se extrae de `DocumentoEncontradoView`

VerificacionFacialHuellaEvent

Método	Descripción
onLoader(isVisible: Boolean)	cada vez que el view necesite mostrar o no un loader
onError(mensaje: String, error: ResponseException?)	aquí llegarán los errores como texto o como un objeto `ResponseException`
onResult(result: Result)	metodo que traera un objeto de tipo InformacionPersonal, para obtener el resultado `val informacionPersonal = result.getOrNull() as InformacionPersonal`

Información personal

Muestra la información personal de un documento consultado (Nombres, Apellidos, Foto del documento, fecha de expiración y la opción de zoom en el documento)

paquete: `com.getgroup.integraidsdk.Ui.InformacionPersonal.InformacionPersonalView`

Método	Descripción
init(event: InformacionPersonalEvent?, model: InformacionPersonal)	inicializa el view, se debe pasar un evento de tipo `InformacionPersonalEvent` y el objeto `InformacionPersonal` extraído de `VerificacionFacialHuellaView`.
Iniciara un temporizador con el tiempo que se haya configurado el sdk	

Método	Descripción
destroy()	detiene el temporizador y destruye el view

InformacionPersonalEvent

Método	Descripción
onMostrarFuentesExternas(model: InformacionPersonal): Boolean	cuando hay fuentes externas disponibles, el usuario puede dar click para verlas y primero pasará por aquí, para mostrarlas en un activity propio retornar false

Fuentes externas

Muestra la lista de fuentes externas disponibles de un usuario

paquete: `com.getgroup.integraidsdk.Ui.ListaFuentesExternas.FuentesExternasView`

Método	Descripción
init(event: FuentesExternasEvent?, list:List)	inicializa el view, se le pasa un evento de tipo `FuentesExternasEvent` y la lista de fuentes externas que se toma a partir del objeto `InformacionPersonal` que se extrae de `InformacionPersonalView`

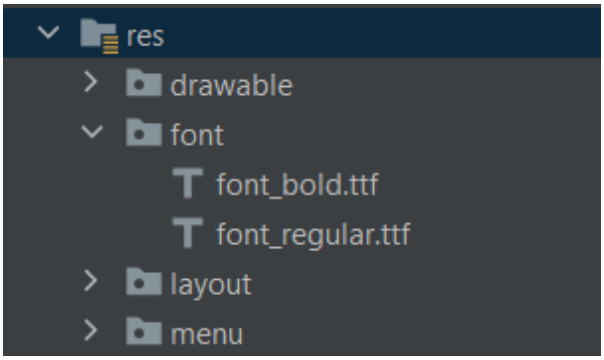
FuentesExternasEvent

Método	Descripción
onFuenteClick(model:FuenteExterna)	Evento que notifica cada vez que se hace click a una fuente externa

Otras funciones

Cambiar tipografías

Existen 2 tipografía en los views bold y regular, para reemplazarlas solo se debe crear una carpeta font dentro de la carpeta res y pegarlas con los nombres font_bold.ttf y font_regular.ttf



Cambiar textos

En el archivo de strings del proyecto se pueden cambiar los valores de estos strings

```
<string name="doc_nacional">Nacional</string>
<string name="doc_extranjero">Extranjero Residente</string>
<string name="numero_identificacion">Número de identificación</string>
<string name="consultar">Consultar</string>
<string name="regresar">Regresar</string>
<string name="wait">Espera un momento por favor...</string>
<string name="verificacion_facial">Reconocimiento Facial</string>
<string name="verificacion_huella">Verificación por huella</string>
<string name="seleccione_tipo_verificacion">Selecciona el tipo de verificación para</string>
<string name="intentar_de_nuevo">Intentar de nuevo</string>
<string name="verificar">Verificar</string>
<string name="necesitas_conceder_permisos">Necesitas conceder permisos</string>
<string name="consultar_mas_informacion">Consultar más información</string>
<string name="aceptar">Aceptar</string>
<string name="informacion_eliminada">La información ha sido eliminada del dispositivo</string>
<string name="antes_continuar">Antes de continuar:</string>
<string name="antes_continuar_lea">Antes de continuar:\n\nPor favor, lea atentamente lo
siguiente.\nAl continuar, usted confirma que está de acuerdo con las siguientes declaraciones:
</string>
<string name="terminos_condiciones">Ver términos y condiciones</string>
<string name="acepto_terminos">He leído y estoy de acuerdo con estos términos y
condiciones</string>
<string name="confirmar_ciudadano">Por favor confirmar que el ciudadano:</string>
<string name="no_documento">No. Documento</string>
<string name="esta_confirmando">Esta confirmando que esta de acuerdo con las siguientes
declaraciones</string>
<string name="declaraciones">Acepta dar su consentimiento para usar su fotografía (Selfie) y
Huellas dactilares para verificar su identidad.\n\nHa leido y aceptado la politica de
privacidad\n\nHa leido y acepta las condiciones de uso</string>
<string name="autorizo">Autorizo</string>
<string name="ok_">Ok</string>
<string name="exp">EXP:</string>
<string name="No">No:</string>
<string name="limpiar">Limpiar</string>
<string name="camara_frontal">Cámara frontal</string>
<string name="camara_trasera">Cámara trasera</string>
<string name="no_hay_fuentes">No hay fuentes para consultar</string>
<string name="transaccion_copiada">¡Transacción copiada!</string>
<string name="transaccion">Transacción</string>
<string name="identificacion">Identificación</string>
<string name="confirmar">Confirmar</string>
<string name="iniciar_nueva_consulta">Iniciar nueva consulta</string>
<string name="doble_click_zoom">Doble click para zoom</string>
<string name="cuatro_dedos">4 dedos de la mano</string>
<string name="los_pulgares">Los Pulgares</string>
<string name="el_pulgar_de_mano">El Pulgar de la mano</string>
<string name="derecha">Derecha</string>
<string name="izquierda">Izquierda</string>
<string name="ubique">Ubique</string>
<string name="acerque_su_mano">Acerque su mano</string>
<string name="aleje_su_mano">Aleje su mano</string>
<string name="enfoque_mano">Mueva su mano para enfocar</string>
<string name="mantenga_mano_quita">Mantenga su mano quieta</string>
<string name="camera_close_message">Acercate a la cámara</string>
<string name="camera_far_message">Alejate un poco de la cámara</string>
<string name="camera_up_message">Sube la cámara hacia ti</string>
<string name="camera_down_message">Baja la cámara hacia ti</string>
<string name="open_eyes_message">Abre los ojos y mira a la cámara</string>
<string name="multiple_faces_detected">Multiples rostros detectados</string>
<string name="detection_failed_message">Detección de rostro fallida</string>
```

```
<string name="no_faces_detected_message">No se reconoce un rostro</string>
<string name="look_straight_message">Mira recto</string>
<string name="mask_detected_message">Máscara detectada</string>
<string name="sunglasses_detected_message">Gafas detectadas</string>
<string name="eyes_closed_message">Abre los ojos</string>
<string name="hold_still_message">Quedate quieto</string>
<string name="spoof_detected_message">Falsificación detetada</string>
<string name="ok_str">Capturando...</string>
<string name="please_wait">Por favor espera...</string>
<string name="network_activity_error">Algo va mal, por favor intenta de nuevo</string>
<string name="label_occulusion">Oclusión</string>
<string name="label_eye_close">Ojo cerrado</string>
<string name="label_livehness">Liveness</string>
<string name="hold_still_and_center">Centra tu rostro y quédate quieto</string>
<string name="te_queda">"Te queda: "</string>
<string name="error_documento_no_encontrado">El documento no esta disponible para
consulta</string>
<string name="error_generico">Ocurrio un error, comuniquese con el administrador</string>
<string name="error_no_autorizado">Su sesión ha expirado</string>
<string name="error_no_autorizado_forbidden">No estas autorizado para esta consulta [Apikey
invalido]</string>
<string name="error_req_identificacion">Ingrese el número de identificación</string>
<string name="error_req_tipo_identificacion">Seleccione el tipo de documento</string>
<string name="error_timeout">La solicitud excedió el tiempo máximo de espera</string>
<string name="error_unauthorizes_verificacion">El token expiro por favor realice la consulta del
documento nuevamente</string>
<string name="error_intentos_maximos">Se alcanzaron la cantidad de intentos máximos</string>
<string name="error_no_coincidencias">No se encontraron coincidencias ó no alcanzó el rango de
verificación mínimo</string>
<string name="error_imagenes_cache">Ocurrio un error con la memoria del disposito, se necesita
realizar la verificación facial o de huellas nuevamente</string>
<string name="error_req_terminos">Debes aceptar los terminos y condiciones</string>
<string name="error_score">No se alcanzo el score mínimo, intentalo de nuevo</string>
<string name="error_req_firma">Necesitas firmar para autorizar</string>
<string name="error_seguridad">Error de seguridad</string>
<string name="error_desconocido">Error desconocido</string>
<string name="error_copiar_transaccion">Ocurrio un error al copiar la transacción</string>
<string name="error_cargar_pagina">Ocurrio un error al cargar la pagina</string>
<string name="error_tomar_captura">Ocurrio un error al tomar la captura</string>
<string name="error_liveness">No se identificó una persona</string>
<string name="error_response">Error en el servidor, por favor intenta de nuevo</string>
```

3) Biometric ID - SDK - IOS

En este documento encontrará una explicación de cómo se realiza la integración con el SDK BiometricID en iOS

Pre requisitos

- Para hacer uso del SDK debe contar con un API Key válido.
- El Bundle Identifier debe ser previamente autorizado para activar el uso del SDK

Flujo de uso

El SDK está diseñado para utilizarse de manera secuencial, el flujo es el siguiente:

Modo de uso

Existen 2 formas de usar el SDK

- Haciendo uso de los View predefinidos
- Haciendo uso de los ViewModel predefindos

> **Nota**
>
> El SDK está diseñado usando el patron MVVM

Instalación del SDK

Parametros de confguración

Views

En caso de hacer uso de las vistas, estas están diseñadas con la siguiente secuencia, sin embargo cada una funciona de manera independiente, permitiendo al desarrollador control en el paso a paso de las vistas.

DocumentQueryView

Esta vista permite ingresar los datos de consulta, en Tipo de documento y Número de documento

El constructor de la vista contiene 3 parametros:

- model

Debe ser de tipo `DocumentQueryViewModel`

- onNext

Acción a ser llamada al oprimirse el botón de Continuar

- content

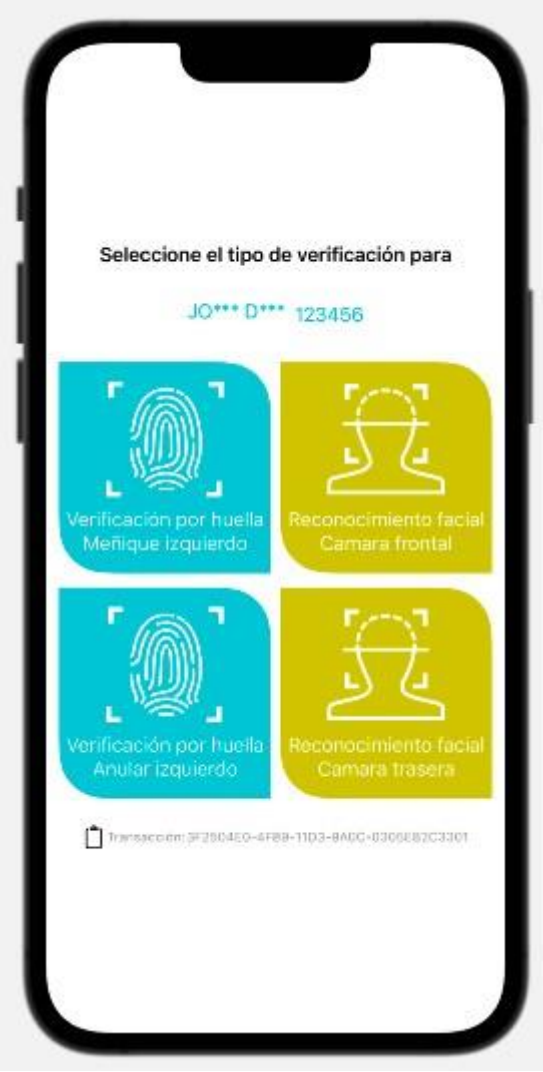
El contenido entre el TextField de número de documento y el Button de Continuar es un ViewContent que permite ingresar lo que se necesite.

```
@ObservedObject var documentQueryViewModel = DocumentQueryViewModel()
```

```
DocumentQueryView(
    model: documentQueryViewModel,
    onNext:{
        print("onNext")
        //Paso sugerido:
        //Usar documentQueryViewModel.Query para consultar el documento
    })
    {
        VStack{
            Text("Antes de continuar").padding(.bottom, 20)
            Text("Por favor, lea atentamente lo siguiente.\nAl continuar, usted confirma que está
de acuerdo con las siguientes declaraciones:").padding(.bottom, 20)
            Link("Ver términos y condiciones", destination: URL(string:
"https://www.w2dit.com")!).padding(.bottom, 20)
            Text("He leído y estoy de acuerdo con estos términos y condiciones").padding(.bottom,
20)
        }
    }
```

BiometricQueryView

Esta vista permite iniciar la verificación biometrica por reconocimiento facial o captura de huella según estén disponibles los metodos de validación para el documento consultado.



Info

A partir de esta vista se puede identificar el ID asociado a esta transacción.

```
var available = DocumentAvailableResponse()
```

Nota

DocumentAvailableResponse es la respuesta obtenida al consultar si un documento está disponible para su verificación biometrica usando DocumentQueryViewModel.Query

```
var biometricQueryViewModel = BiometricQueryViewModel(model: available, documentNumber:
documentQueryViewModel.documentNumber, documentType: documentQueryViewModel.documentType))

BiometricQueryView(model: biometricQueryViewModel, onFaceCaptured: { image in
                            //la imagen del rostro fue capturada

                            biometricQueryViewModel.QueryByPhoto(fullFacePhoto: image) { response
in
                                //Resultado de verificación biometrica

                                onBiometricQueryResponse(response: response, viewModel: viewModel)
                            }

                        }, onFingerCaptured: { finger , fingerPrint in
                            //la imagen de la huella fue capturada

                            biometricQueryViewModel.QueryByFinger(finger: finger, fingerPrint:
fingerPrint) { response in
                                //Resultado de verificación biometrica

                                onBiometricQueryResponse(response: response, viewModel: viewModel)
                            }

                        }, onFaceFailed: { error in
                           //La captura del rostro falló
                        }, onFingerFailed: { error
                            in
                           //La captura de la huella falló
                        })
```

```
func onBiometricQueryResponse(response:Result<BiometricVerificationResponseBase?, Error>,
viewModel:BiometricQueryViewModel){
    switch response {
    case .success(let data):
        if let match = data as? BiometricVerifiedResponse, match.hit == true
        {
            //La verificación biometrica fue exitosa
            //Paso sugerido, Mostrar resultado en BiometricMatchView
        }else{
            //La verificación biometrica no exitosa
            if(data?.remainingTries ?? 0 <= 0){
                //Se alcanzó el limite de intentos
            }else{
                //Aún cuenta con intentos disponibles
            }
        }
        break;
    case .failure(let error):
        if let validationProblem = error as? ValidationProblemDetails{
            //Ocurrió un error de validación
            if(validationProblem.errors != nil && !validationProblem.errors!.isEmpty){
                //Fallas de validación de entrada
                var message = "";
                validationProblem.errors?.forEach({ key, errors in
                    message += errors.joined(separator: "\n")
                })
            }else {
                //Ocurrió otro error de validación
            }
        } lse if let problem = error as? ProblemDetails {
           //Ocurrió un error en el servidor
        }else{
           //Ocurrió un error inesperado o de red
        }
    }
}
```

BiometricMatchView

Esta vista permite mostrar el resultado de la verificación biometrica exitosa en un formato estandarizado.


```
var biometricMatchViewModel = BiometricMatchViewModel(
        currentBiometricResult: match,
        currentQuery: biometricQueryViewModel.currentQuery,
        documentNumber: biometricQueryViewModel.documentNumber)
```

```
BiometricMatchView(model: biometricMatchViewModel,
onFinish: {
    //Acción a ser ejecutada al finalizar el contador de tiempo
},
onGetMoreInformation: {
   //Acción a ser ejecutada al oprimir el botón de más información
   //Paso suguerido:
   //Mostrar fuentes adicionales disponibles
})
```

ViewModels:

DocumentQueryViewModel

- Query

Metodo que permite ejecutar la consulta de documento.

```
documentQueryViewModel.Query(completionHandler: { response in
    switch response {
    case .success(let data):
        if let available = data as? DocumentAvailableResponse, available.biometricAvailable ==
true
        {
            //El número de documento está disponible para su consulta
            //Paso sugerido:
            //Mostrar view para capturar huella o foto del rostro según corresponda.
        }else{
           //El número de documento no está disponible para su verificación
        }

    case .failure(let error):
        if let validationProblem = error as? ValidationProblemDetails{
          //Ocurrio un error de validación de la entrada
        }else if let problem = error as? ProblemDetails {
          //"Ocurrió un error en el servidor")
        }else{
          //Ocurrió un error inesperado
        }
    }
})
```

BiometricQueryViewModel

- Constructor

Recibe como modelo la respuesta a la llamada de documentQueryViewModel.Query cuando el documento esta disponible para su consulta.

```
init(model:DocumentAvailableResponse,
    documentNumber:String,
    documentType:DocumentTypeEnumeration)
```

- StartFaceCapture

Inicia la camara del dispositivo para el proceso de captura facial.

> **Nota**
>
> Solo funciona en dispositivos fisicos. En caso de usar el SDK para emulador, esta función no ejecuta código

```
func StartFaceCapture(
        isUseBackCamera:Bool,
        isAutoCapture:Bool,
        isFastCapture:Bool,
        isoEnabled:Bool,
        isGetFullFrontalCrop:Bool,
        useCompresion:Bool,
        onFaceCaptured: @escaping (NSData?,[String],[String]) -> Void,
        onCancelled: @escaping() -> Void,
        onFaceCaptureFailed: @escaping (String) -> Void,
        onTimedout: @escaping (NSData?) -> Void) -> Void
```

- QueryByPhoto

Ejecuta la consulta de verificación biometrica por rostro usando una imagen en base64

```
func QueryByPhoto(
    fullFacePhoto:String,
    completionHandler:
        @escaping (Result< BiometricVerificationResponseBase?, Error>) -> Void
```

- StartFingerCapture

Inicia la camara del dispositivo para el proceso de captura de huella.

> **Nota**
>
> Solo funciona en dispositivos fisicos. En caso de usar el SDK para emulador, esta función no ejecuta código

```
func StartFingerCapture(
    finger:Finger,
    onFingersCaptured: @escaping ([CapturedFinger]) -> Void,
    onFingerCaptureFailed: @escaping (String)->Void) -> Void
```

- QueryByFinger

Ejecuta la consulta de verificación biometrica por huella usando una huella WSQ en base64

```
func QueryByFinger(
    finger:Finger,
    fingerPrint:String,
    completionHandler:
        @escaping (Result< BiometricVerificationResponseBase?, Error>) -> Void
```

BiometricMatchViewModel

- Constructor

Recibe el resultado exitoso de una verificación biometrica y el resultado exitoso de la consulta de documento

```
init(currentBiometricResult: BiometricVerifiedResponse,
    currentQuery:DocumentAvailableResponse,
    documentNumber:String)
```

4) Biometric ID - SDK Web

En este documento encontrará una explicación de cómo se realiza la integración con el SDK Web BiometricID

Pre requisitos

Para hacer uso del SDK se debe contar con un Api Key válido: {{ API_KEY }}

Definiciones

Nombre	Descripción
{{ URL_SDK }}	https://websdkbiometricid.racsa.go.cr

Paso 1

En el html definimos un iframe y le asignamos la {{ URL }} del sdk

```
<iframe id="sdkFrame" class="sdk" src="{{ URL_SDK }}" frameborder="0" allow="camera 'src'">
</iframe>
```

Paso 2

Creamos un script que iniciará el iframe `index.js`

```
let frameId = 'sdkFrame';
document.getElementById(frameId).onload = function () {
    var msg = {
        apiKey:"{{ API_KEY }}",
    };
    this.contentWindow.postMessage(msg,this.src);
};
```

Paso 3

Cargar el script al html al final del body

```
<script src="index.js"></script>
```