T4iSB

DOCUMENTO | **ESPECIFICAÇÃO TÉCNICA**
PRODUTO | **SDK DE COLETA DE IMPRESSÕES DIGITAIS E FOTO DA FACE EM APLICATIVOS MOBILE**
FABRICANTE | **T4ISB TECNOLOGIA E PARTICIPAÇÕES LTDA.**
VERSÃO | **2022.1**

CONFIDENCIALIDADE

Este documento contém informações de propriedade da T4iSB BioLogica, sendo disponibilizadas de forma confidencial para uma finalidade específica. O destinatário garante a custódia e o controle, e concorda que este documento não será copiado ou reproduzido no todo ou em parte, nem seu conteúdo revelado de qualquer forma ou a qualquer pessoa, exceto no cumprimento da finalidade para a qual foi entregue.

Esta descrição se aplica a todas as páginas deste documento.

Este documento também poderá ser obtido a partir do endereço https://www.t4isb.com.

SUMÁRIO

1 INTRODUÇÃO

Este documento apresenta o SDK para coleta de impressões digitais e foto da face em aplicativos mobile, utilizado em processos de cadastramento, identificação e verificação de pessoas a partir de dispositivos móveis.

2 CARACTERÍSTICAS GERAIS

- SDK composto por biblioteca de funções, capaz de coletar no mínimo 8 impressões digitais e foto da face para uso em aplicativos mobile.
- Capaz de realizar a detecção de vivacidade de uma pessoa.
- Compatível com coleta das impressões digitais em imagem no formato WSQ.
- Compatível com coleta da face em imagem no formato JPEG.
- Capaz de coletar as impressões digitais e os registros da face a partir dos recursos da câmera convencional dos equipamentos mobile.
- Não necessidade de hardware adicional para leitura de impressões digitais e registro da face.
- Agnóstico aos sistemas operacionais, podendo operar nos principais sistemas operacionais utilizados nos dispositivos móveis existentes no mercado.
- Compatível com Android 5.1 ou superior.
- Compatível com iOS 11 ou superior.
- Disponibilizado de forma a suportar as linguagens *Java/Kotlin* para a plataforma Android e Swift para IOS, possibilitando desenvolvimento híbrido, com suporte aos seguintes frameworks:
    - *Cordova* CLI 6.x ou superior;
    - *Ionic* 3.18.X ou superior;
    - *Cordova* Android 6.3.x ou superior;
    - *Cordova* iOS 4.4.x ou superior.

3 VISÃO GERAL DO PRODUTO

O SDK de impressões digitais e face para utilização em dispositivos móveis da T4ISB permite o registro e a validação de identidade de uma pessoa, a partir do seu próprio dispositivo pessoal, possibilitando que sua identidade seja protegida e proporcionando controle total de seus dados biométricos e identidade soberana.

3.1 Registro de Face

O SDK fornece uma funcionalidade que possibilita registrar o rosto de uma pessoa. Para tal, o aplicativo cliente precisa implementar “EnrollFaceListener” para poder obter o “EnrollFaceResult”. O “EnrollFaceResult” possui dois atributos:

1. **livenessScore**: Verifica a vivacidade do rosto e retorna uma pontuação entre 0 e 100. A pontuação 100 é um rosto 100% vivo e 0 é falso.
2. **croppedFace**: Retorna um rosto recortado (formato de bitmap) para que o aplicativo cliente possa usá-lo para fins de verificação/identificação.

Logo abaixo, um exemplo de como acionar a atividade responsável pelo registro do rosto (versão Kotlin).

```
FaceActivity.setFaceListener(this)
val myIntent = Intent(this@MainActivity, FaceActivity::class.java)
myIntent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK)
myIntent.addFlags(Intent.FLAG_ACTIVITY_FORWARD_RESULT)
startActivity(myIntent)
```

Logo após o registro da face, o aplicativo cliente receberá o resultado substituindo a função “onEnrollFaceCompleted”, conforme apresentado no trecho de código a seguir.

```
override fun onEnrollFaceCompleted(enrollFaceResult: EnrollFaceResult?) {
    if (enrollFaceResult != null) {
        runOnUiThread {
            Toast.makeText(
                this@MainActivity,
                "Successful Face Enrollment. Liveness Score
${enrollFaceResult.livenessScore}",
                Toast.LENGTH_LONG
            ).show()
        }
    }
```

3.2 Registro de Impressões Digitais

O SDK fornece uma funcionalidade que permite o registro de impressões digitais da pessoa utilizando a câmera do dispositivo móvel. Para tal, o aplicativo cliente neccessita implementar "EnrollFingerListener" para poder obter o "EnrollFingersResult". O "EnrollFingersResult" possui três atributos:

1. **leftHandScore**: Um índice de qualidade de 0 a 100 da mão esquerda.
2. **rightHandScore**: Um índice de qualidade de 0 a 100 da mão direita.
3. **fingerMap**: Mapa contendo as imagens das impressões digitais cadastradas.

Logo abaixo, um exemplo de como acionar a atividade responsável pelo registro dos dedos (versão Kotlin).

```
FingerActivity.setEnrollFingerListener(this)
val myIntent = Intent(this@MainActivity, FingerActivity::class.java)
myIntent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK)
myIntent.addFlags(Intent.FLAG_ACTIVITY_FORWARD_RESULT)
startActivity(myIntent)
```

Após o registro dos dedos, o aplicativo cliente receberá o resultado substituindo a função "onEnrollFingerCompleted":

```
    override fun onEnrollFingerCompleted(enrollFingersResult:
EnrollFingersResult) {
      val keys: Set<*> = enrollFingersResult.fingersMap.keys
      println("keys $keys")

      val i = keys.iterator()
      while (i.hasNext()) {
        val key = i.next() as String
        println("key $key")
        val value = enrollFingersResult.fingersMap[key]
        println("value $value")

      }

      runOnUiThread {
        Toast.makeText(
          this@MainActivity,
          "Left Hand Score  ${enrollFingersResult.leftHandScore} \n Right Hand
Score ${enrollFingersResult.rightHandScore}",
          Toast.LENGTH_LONG
        ).show()
      }
    }
```

3.3 Outras Funções

- void **InitLiveness** (AssetManager assetManager)
- void **DetectFace** (long matAddrRgba, int[] box)
- float **TestLiveness** (long matAddrRgba, int[] box)
- boolean **CheckIllum** (long matAddrRgba)
- boolean **CheckBlur** (long matAddrRgba)
- boolean **CheckBackground** (long matAddrRgba, int[] box)
- native void **jniInitLiveness** (AssetManager assetManager)
- native void **jniDetectFace** (long matAddrRgba, int[] box)

- native float **jniTestLiveness** (long matAddrRgba, int[] box)

- native boolean **jniCheckIllum** (long matAddrRgba)
- native boolean **jniCheckBlur** (long matAddrRgba)
- native boolean j**niCheckBackground** (long matAddrRgba, int[] box)
- void **InitFinger** (String rootDir, String dicDir, int mode, String cfgPath, String weightPath)
- int **TestFinger** (long matAddrRgba, int[] capFlags)
- void **GetFingerMat** (long matA0, long matA1, long matA2, long matA3)
- void **GetQualityScores** (float[] scores)
- float **GetQScore** ()
- void **MyLog** (String info)
- void **GetFingerData** (int[] pPoints)
- native void j**niInitFinger** (String rootDir, String dicDir, int mode, String cfgPath, String weightPath)
- native int j**niTestFinger** (long matAddrRgba, int[] capFlags)
- native void j**niGetFingerMat** (long matA0, long matA1, long matA2, long matA3)
- native void **jniGetQualityScores** (float[] scores)
- native void j**niMyLog** (String info)
- native void **jniGetFingerData** (int[] pPoints)
- native float **jniGetQScore** ()