Certiface SDK

Esta documentação contempla as instruções, conceito operacional como também funcionalidades do serviço em nuvem de certificação facial.

Resumo geral:

Este documento visa mostrar de maneira clara e objetiva o conceito operacional e exemplos de integração com o produto Certiface SDK. Exemplos serão disponibilizados neste documento assim facilitando o tempo de absorção de conhecimento e integração da tecnologia junto ao legado do cliente. Dúvidas, críticas e sugestões enviar para o e-mail contato@oititec.com.br

Sumário

Introdução:

A fraude de documentos, clonagens e falsificação de cartões e outros tipos, estão presentes constantemente nos veículos de comunicação. Para combater este cenário, somente um produto/serviço com recursos inovadores com tecnologia de ponta tem a capacidade de proteger a identidade das pessoas, este denomina-se Certiface.

O Certiface é um serviço de certificação de pessoas utilizando a tecnologia de biometria facial em nuvem que proporciona alta disponibilidade e resiliência exigido por toda operação em nuvem. Este serviço utiliza diversos algoritmos de reconhecimento facial e um módulo de (AI) inteligência artificial com constante aprendizado que define o peso de cada algoritmo (por consequência definindo a confiabilidade de similaridade).

O módulo de inteligencia artificial foi construído devido à experiência desde 1997 com visão computacional e biometria do seu inventor. Portanto o módulo neural absorve toda perícia do seu criador junto ao aprendizado e/ou percepção matemática e evolutivas dos algoritmos de biometria facial.

Apesar do seu complexo funcionamento, o Certiface identifica milhões de usuários por segundo. A conversão da face em código biométrico binário, gravada na base de dados centralizada permite comparar faces de fraudadores em tempo real, assim verificando a duplicidade e inconsistências das faces cadastradas. Com tais atributos a solução é adequada para o mercado de varejo, bancário, serviços públicos e qualquer mercado que corre o risco de fraude de identidade.

Conceito operacional

O sistema consiste no envio da foto do usuário presente diante a câmera para um posterior cadastramento e/ou autenticação. Como o sistema concentra-se nos servidores do produto, a implantação em larga escala acontece de maneira simples e transparente para o usuário. Pois o Certiface trabalha com cluster de reconhecimento e biometria facial, assim proporcionando escalabilidade e alta disponibilidade na autenticação e/ou identificação do usuário.

O serviço carrega a principal função de vincular a face do cliente ao seu respectivo documento (CPF). Além de centralizado, proporciona um conjunto de ferramentas denominado Certiface SDK, ou seja um kit de desenvolvimento que permite uma fácil e rápido integração com todo código legado do cliente

e/ou instituição.

Certiface é um serviço intuitivo, não invasivo (não existe contato fisíco) e possibilita identificar inconsistência durante a autenticação de clientes. No ato da concessão de crédito, abertura de conta-corrente ou financiamento. Vale a pena ressaltar, que a validação convencional de CPF do mercado atualmente, não consegue alertar uma provável ocorrência de fraude com documentos.

A Face é capturada durante a consulta do cliente, a foto é oriunda de qualquer dispositivo que proporcione a captura de imagem (webcam, scanner, camera digital e outros). Com isso, a face obtida é enviada através da rede para pesquisa na base de dados centralizada, onde o principal objetivo é a validação no cluster biométrico. Clientes cadastrados anteriormente, serão validados com o template biométrico posteriormente inseridos na base de dados. Novos clientes tem sua amostragem biométrica pesquisada na base centralizada, ou, verificada contra duplicidade (Faces similares com CPFs distintos).

Funcionamento na interface Web

O serviço Certiface como também os métodos de integração do SDK (ferramenta de desenvolvimento) é disponibilizado através de URL (endereço de internet). Este endereço pode variar dependendo da magnitude referente ao volume de acesso do cliente e/ou complexidade de integração a rede privada do cliente. Ao acessar o endereço, um login será solicitado baseado na ilustração abaixo (figura 1):

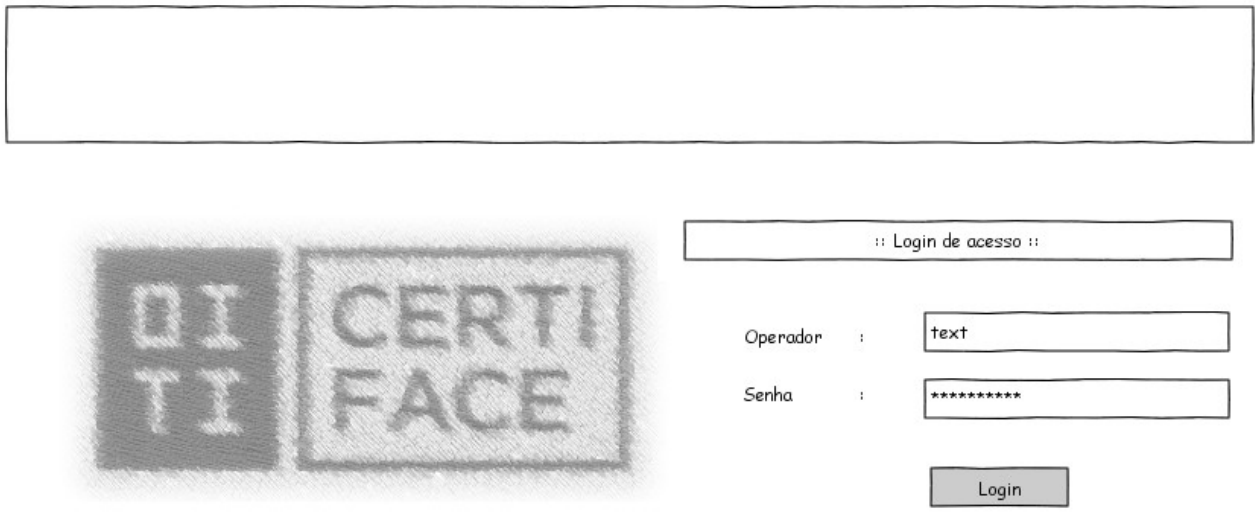


figura 1: login de acesso

A autenticação é efetuada baseada no preenchimento dos campos operador e senha. Com os dados devidamente informados, o sistema obtêm informações suficientes para liberação do serviço contratado. Se os dados informados forem digitados incorretamente, o sistema mostrará uma mensagem de dados incorretos.

A próxima tela (figura 2) apresenta ao operador/usuário a imagem ao vivo do dispositivo de captura onde o operador do sistema deve centralizar a face do cliente para posteriormente recortá-la ao clicar no botão “Capturar Imagem”. Vale a pena evidenciar, que qualquer anomalia na captura da imagem, o operador deve capturar novamente clicando no botão “Capturar imagem”. Ao concluir a captura certificando que a face esta visível e com aparência visual suficiente para processamento, o operador deve pressionar o botão “Enviar”, assim iniciando o processo de upload da imagem e consequentemente submetendo a foto para processamento de qualidade baseado no ISO 19794-5. Caso a imagem submetida seja insuficiente em termos computacional, o sistema informará o operador, assim sendo necessário repetir o procedimento de captura.

figura 2: Captura da imagem

A etapa seguinte é informar o CPF do cliente (figura 3), se o mesmo não estiver presente na base de dados, outros campos na tela posterior, serão disponibilizados para o preenchimento da ficha cadastral, os campos presentes no formulário além do CPF são: nome e data de nascimento conforme figura a seguir (figura 4). Nesta fase será apresentada a face do usuário, seguido dos campos referente aos dados cadastrais.

Após o preenchimento de todos os dados, o operador deve pressionar o botão verificar onde será analisado a existência do usuário na base centralizada, caso o CPF não esteja cadastrado, o processo de inclusão será efetuado e a face associada ao CPF será submetida ao processo de identificação para checar a existência de outros cadastros da face atual com CPF distinto. O resultado será apresentado em forma de lista de similares identificados (figura 6).

Se o CPF já estiver presente na base de dados, a face recém-capturada será comparada matematicamente com imagem armazenada anteriormente no banco de dados. O resultado do processamento será exibição da similaridade da foto cadastral comparada a foto capturada (figura 5).

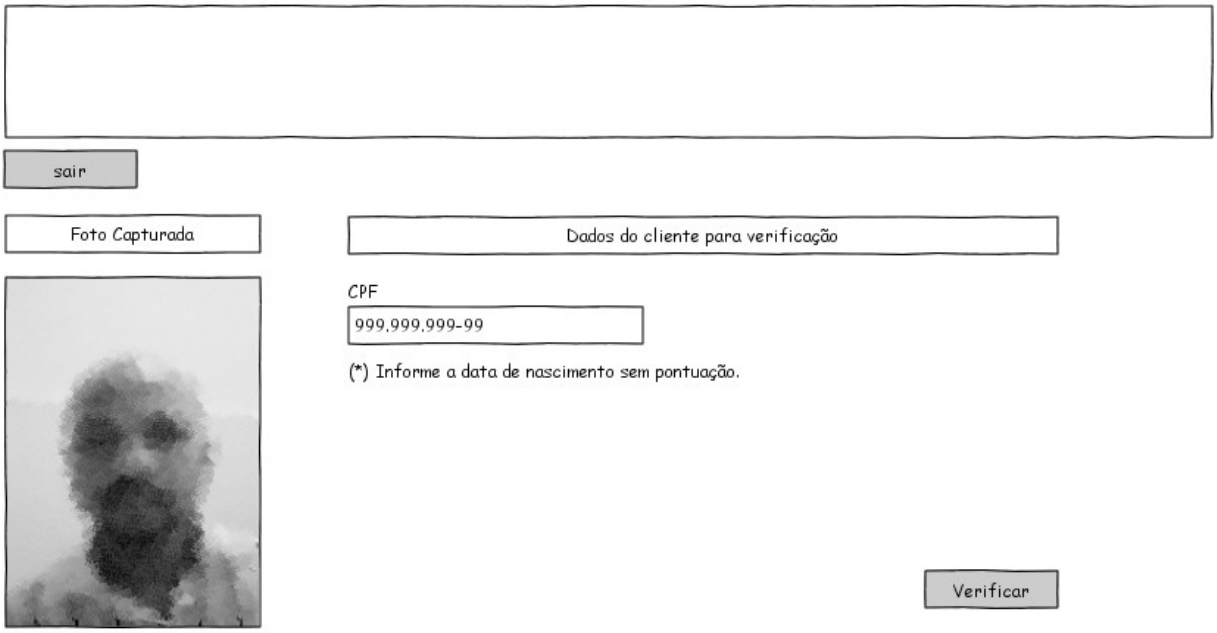


figura 3: digitação do CPF

Nesta etapa a imagem foi submetida e aprovada no processo bioCubeAF102 que verifica as propriedades da face, ou seja, olhos abertos, boca fechada, face frontal, iluminação uniforma e outros necessários para uma boa amostra para calcular o template biométrico.

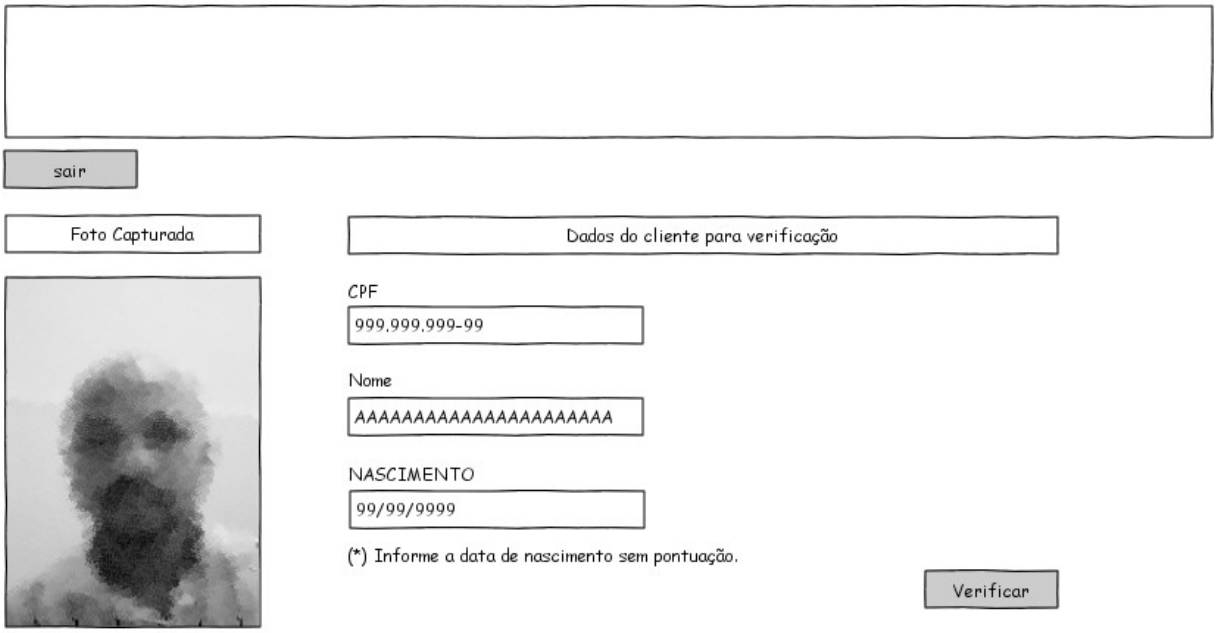


figura 4: dados cadastrais

figura 5: certificação biométrica

Após o processamento biométrico, é disponibilizado a interface de upload de documentos onde o balconista/atendente digitaliza os documentos como comprovantes de renda, de endereço, CPF, RG e outros necessários para aprovação do processo em questão. A digitalização de documentos é um processo/recurso opcional e importante para comprovar a fraude. Pois além de servir como prova da fraude, o documento oferece uma comprovação visual da

inconsistência. Em casos como este, o cliente recebe uma notificação de alta similaridade através de e-mail, SMS e outros meios de comunicação.

Existem diversas parametrizações que permite habilitar e desabilitar a visualização no balcão de determinadas informações. Por exemplo, a operação na loja pode ser parametrizada para não exibir nenhum resultado biométrico do Certiface, e sim apresentar apenas a conclusão de todo o processo, e consequentemente se ocorreu algum erro ou foi devidamente finalizado.

Todo digitalizado e armazenado no serviço Certiface, está disponível para consulta a qualquer momento independente da operação atual. Então sempre que necessário o departamento de credito ou outro departamento (com direitos designados) poderão obter acesso aos arquivos de imagens digitalizadas sem a necessidade da operação biométrica.

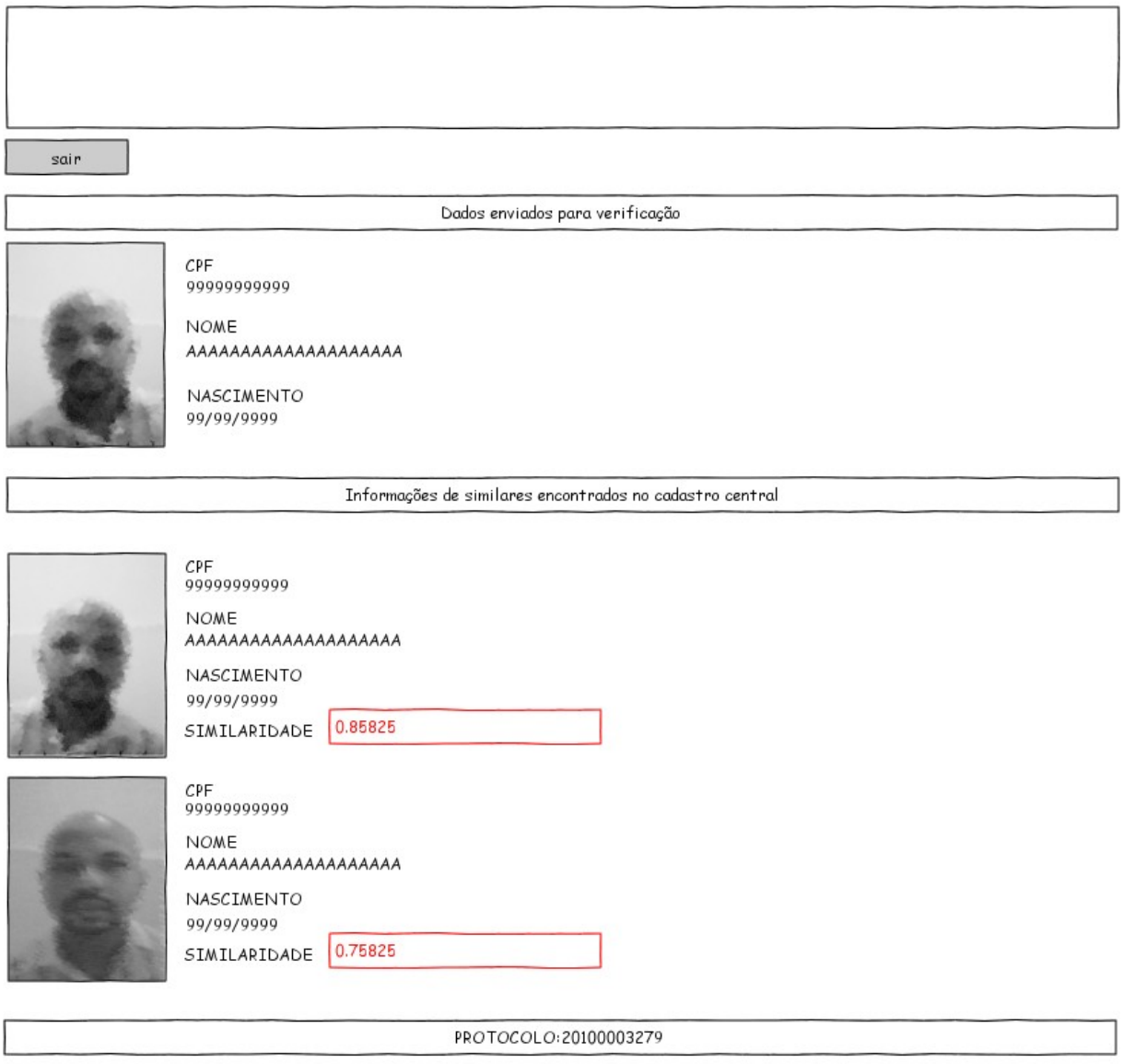


figura 6: lista de similares

Conceito da qualidade da imagem

Para obter o melhor resultado do serviço Certiface, devemos atender dentro da possibilidade operacionais da loja a ISO 19794-5. A seguir veremos uma imagem com o resultado do processamento de verificação da qualidade da imagem (figura 7).

O processamento de análise verifica em diversos aspectos as propriedades da face, mas os principais atributos são a posição frontal da face, olhos abertos, boca fechada (teste de expressão), óculos escuros ausente e distância mínima entre os olhos. Valer apena ressaltar que o formato deve se preferencialmente JPG/JPEG, e sua compressão **não deve inferior a 85%**. Pois imagens com alta compressão elimina informações relevantes para o processamento da biometria facial.

Não obedecer aos principais itens da ISO 19794-5 pode trazer menos precisão na identificação/verificação facial. Então como resultado deste cenário podemos aumentar a taxa de Falso positivo ou Falso negativo.

figura 7: atributos da imagem

Existem diversos atributos que classificam uma imagem como imprópria para o processamento do template matemático. A seguir veremos alguns poucos exemplos de não conformidade da imagem durante um cadastramento ou certificação.

figura 8: face não frontal

figura 9 : fotos inconsistentes.

Requisitos de integração com a serviço Certiface

Independente da tecnologia REST ou SOAP, o Certiface apresenta poucos requisitos mínimos. Este requisito estão relacionados ao conceito operacional como também a qualidade da imagen. A seguir os requisitos obrigatórios:

- Resolução mínima de 640x480;
- Recorte na proporção 3:4.5 (320x480);
- Compressão superior ou igual a de 85%;
- Em caso de integração em computadores instalado no estabelecimento, sempre utilizar um login por loja.

Integração com o serviço Certiface SOAP

A integração com o serviço Certiface utiliza a tecnologia SOAP (originado do acrônimo inglês Simple Object Access Protocol) é um protocolo para troca de informações estruturadas em uma plataforma descentralizada e distribuída, utilizando tecnologias baseadas em XML. Sua especificação define um framework que provê maneiras para se construir mensagens que podem trafegar através de diversos protocolos e que foi especificado de forma a ser independente de qualquer modelo de programação ou outra implementação específica. Por não se tratar de um protocolo de acesso a objetos, o acrônimo não é mais utilizado.

Geralmente servidores SOAP são implementados utilizando-se servidores HTTP, embora isto não seja uma restrição para funcionamento do protocolo. As mensagens SOAP são documentos XML que aderem a uma especificação fornecida pelo órgão W3C. O primeiro esforço do desenvolvimento do SOAP foi implementar RPCs sobre XML.

Consumo do WebService com WSDL

O consumo do WebService Certiface, acontece com a tecnologia WSDL. O Web Services Definition Language (WSDL) é uma linguagem baseada em XML utilizada para descrever Web Services. Trata-se de um documento escrito em XML que além de descrever o serviço, especifica como acessá-lo e quais as

operações ou métodos disponíveis.

Foi submetido ao W3C por Ariba, IBM e Microsoft em março de 2001 sendo que seu primeiro rascunho foi disponibilizado em julho de 2002. A versão atual é 2.0; a versão 1.1 não foi endossada pelo W3C. O WSDL 1.2 foi renomeado para 2.0 e aceita todos os métodos de requisição HTTP (não apenas GET e POST).

Cliente Java para serviço Web

O cliente Java pode consumir o serviço Certiface das mais diversas maneiras. Os desenvolvedores java geralmente utilizam o Netbeans (no qual foi elaborado o exemplo deste documento) ou o Apache Axis. O Apache Axis é um framework de código aberto, baseado na linguagem Java e no padrão XML, utilizado para construção de web services no padrão SOAP. Através do Axis os desenvolvedores podem criar aplicações distribuídas. Além da versão para Java, existe uma implementação baseada na linguagem C++. O projeto Apache Axis é suportado pela Apache Software Foundation.

Requisitos de desenvolvimento Java:

Este capítulo do documento parte do princípio que o leitor ***possui profundos conhecimentos na linguagem java*** como também ***sólidos conhecimentos para desenvolvimento web.*** Sendo assim, a seguir os requisitos básico do desenvolvimento na linguagem Java:

- Windows XP ou superior;
- GNU Linux 32 ou 64 bits;
- MacOS X;
- Java 1.6.0.24 ou superior;
- Apache AXIS 1.4 ou superior;
- Eclipse / Netbeans / Modo console;
- Bibliotecas;
    - Metro 2.0/JAX-WS;

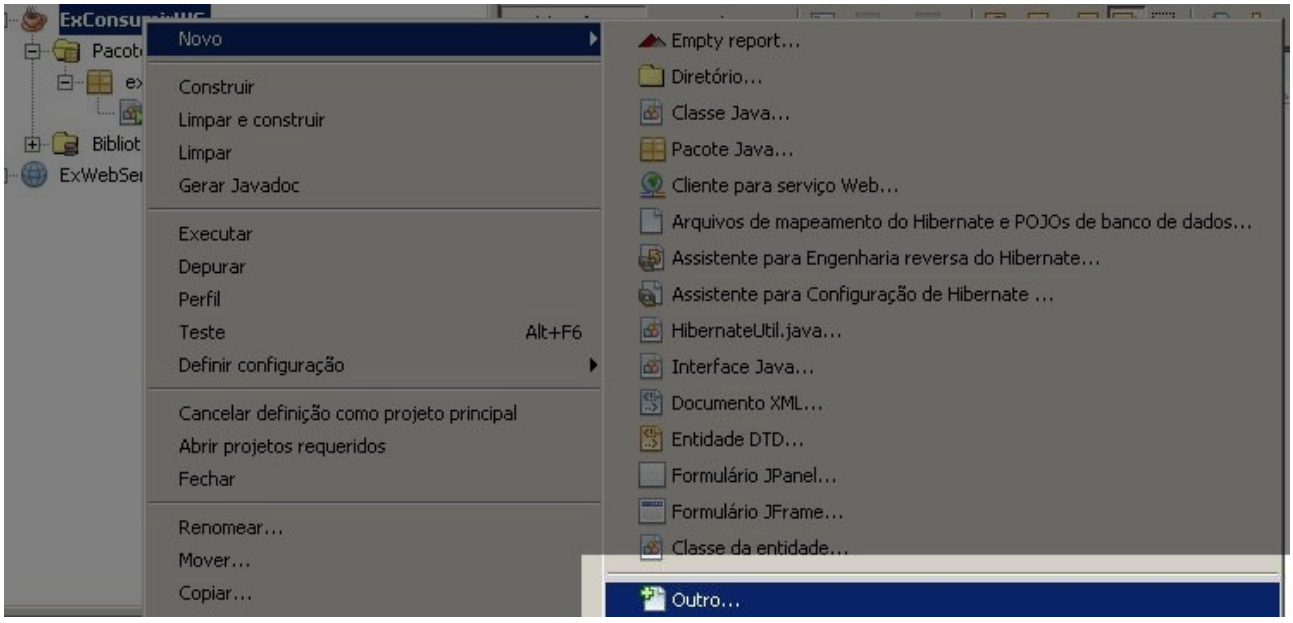
Criação das classes com o Apache AXIS:

Para criar as classes java do WebService Certiface, com o Apache Axis devidamente instalado e configurado, execute:

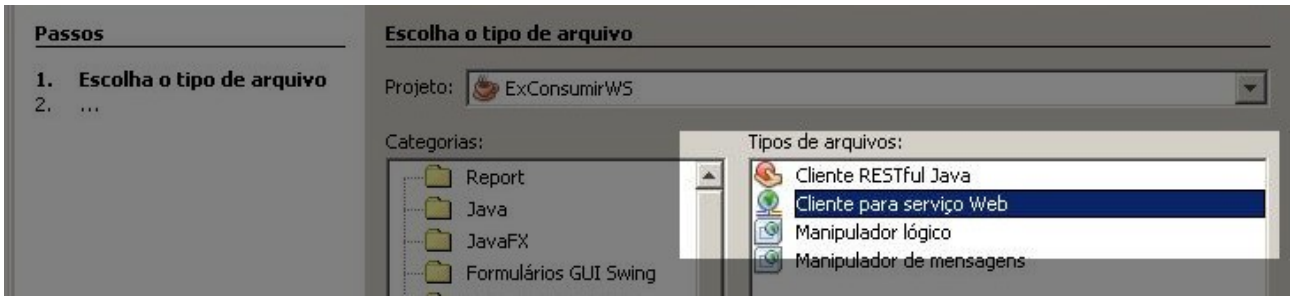
```
$ java org.apache.axis.wsdl.WSDL2Java www.certiface.com.br/CertiFaceWS/CertiFaceWS?wsdl
```

Criação das classes com o NetBeans:

A seguir instruções básicas e superficiais de como criar as classes a partir da URL/Arquivo WSDL. Com o java e o Netbeans devidamente instalado e configurado, inicie a interface Netbeans, crie um novo projeto, clique com o botão direito do mouse e selecione a opção “NOVO” e logo em seguida a opção “OUTRO” conforme a imagem abaixo (fig.10):



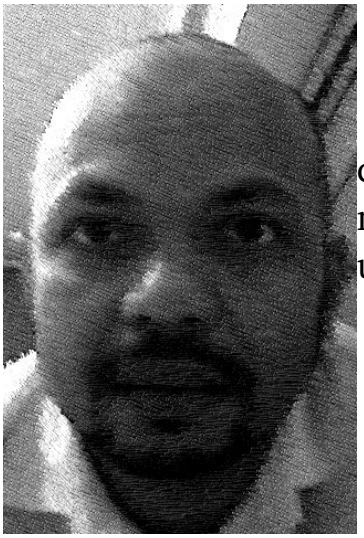
Na caixa de diálogo, selecione o tipo de arquivo como “Cliente para serviço Web”:



Agora digite a URL WSDL do serviço Certiface:
http://www.certiface.com.br/CertiFaceWS/CertiFaceWS?wsdl

Material utilizado

O material/imagens utilizados neste documento estão classificados logo a seguir:

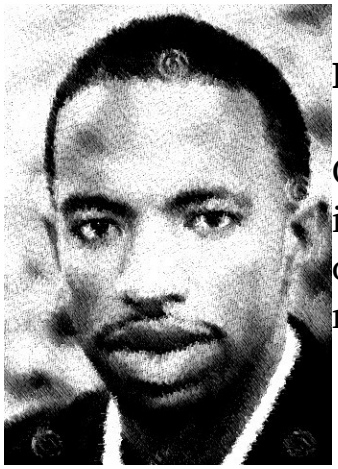


Usuário 1:

O arquivo ***usuario1.jpg*** será utilizado para simular um cadastramento com fraude, ou seja uma face é inserida no certiface com mais de um único CPF. O resultado será uma advertência via SMS e e-mail.

Alguém:

O arquivo ***alguem.jpg*** será utilizado para simular um cadastramento sem fraude. Nesta simulação o cadastro é realizado com sucesso sem notificação alguma.



Perfil:

O arquivo ***perfil.jpg*** será utilizado para simular uma imagem que não obedece a ISO 19794-5. O retorno é uma operação cadastral sem sucesso, com uma mensagem de notificação, mas sem alerta via e-mail e SMS.

Asiático 1:

O arquivo ***asiatico1.jpg*** será utilizado para simular uma operação com certificação negativa, ou seja uma face é submetida para certificação no certiface é o resultado obtido é uma não consistência biométrica. O resultado será uma advertência via SMS e e-mail.

Compilação no terminal e preparação do ambiente

O código fonte exemplo baseia-se no ambiente modo console, assim não favorecendo nenhum sistema e IDE. Antes de compilar e detalharmos o funcionamento do código fonte, devemos preparar o ambiente para tal execução, sendo assim a primeira tarefa é definir as variáveis ambientais JAVA_HOME e ANT_HOME.

```
export JAVA_HOME="/usr/java/latest/"
export ANT_HOME="/usr/share/ant/"
```

O próximo passo é copiar as bibliotecas do pacote Metro (.jar) para a pasta lib do ANT e as biliotecas JAXB para a pasta endorsed:

```
# cd [localização dos jar do pacote Metro]
# cp  webservices-api.jar /usr/share/ant/lib/
# cp webservices-extra-api.jar /usr/share/ant/lib/
# cp webservices-extra.jar /usr/share/ant/lib/
# cp webservices-rt.jar /usr/share/ant/lib/
# cp webservices-tools.jar /usr/share/ant/lib/
```

```
# cd  [localização dos jar do pacote JABX]
# cp jaxb-api.jar /usr/java/latest/jre/lib/endorsed
# cp jaxws-api.jar /usr/java/latest/jre/lib/endorsed
```

Se preferir, em sistema UNIX e derivados, podemos criar links simbólicos (para evitar duplicidades dos arquivos):

```
# cd /usr/share/ant/lib/
# ln -s /home/user/libs/metro/webservices-api.jar
# ln -s /home/user/libs/metro/webservices-extra-api.jar
# ln -s /home/user/libs/metro/webservices-extra.jar
# ln -s /home/user/libs/metro/webservices-rt.jar
# ln -s /home/user/libs/metro/webservices-tools.jar
```

```
# cd /usr/java/latest/jre/lib/endorsed
# ln -s /home/user/libs/jaxb/jaxb-impl.jar
# ln -s /home/user/libs/jaxb/jaxb-xjc.jar
```

Agora se tudo estiver funcionando corretamente, entre na pasta no qual foi descompactado o código exemplo e efetue o comando ANT:

```
$ ant
Buildfile: /home/cabelo/pipoca/src.ok/JCertifaceConsole/build.xml
-pre-init:
-init-private:
-init-user:
…    ...     …
…    ...     …
…    ...     …
javadoc:
default:


BUILD SUCCESSFUL
Total time: 3 seconds
```

Código fonte exemplo

A seguir o código exemplo utilizado para os testes e também exemplificar o uso da ferramenta Certiface.

```
 1 package jcertifaceconsole;
 2
 3 import java.io.File;
 4 import java.io.FileInputStream;
 5 import java.net.MalformedURLException;
 6 import java.net.URL;
 7 import java.util.List;
 8 import net.java.dev.jaxb.array.AnyTypeArray;
 9
10 /**
11  *
12  * @author Leandro
13  */
14 public class JCertifaceConsole {
15
16     private static final String SERVICE_URL =
17         "http://www.certiface.com.br/CertiFaceWS/CertiFaceWS?wsdl";
18
19     private static final Long EMP_COD = 1L;
20     private static final Long FIL_COD = 1L;
21     private static final String USER_ID = "2";
22     private static final String USER_PASS =
"c4ca4238a0b923820dcc509a6f75849b";
```

```
 23
 24     public static class Service{
 25         public static final String VALIDATION = "VAL";
 26         public static final String INDENTIFICATION = "IDE";
 27         public static final String ERROR = "ERR";
 28     }
 29
 30     public static class Result{
 31         public static final String POSITIVE = "POS";
 32         public static final String NEGATIVE = "NEG";
 33         public static final String SUCCESS = "SUC";
 34         public static final String PENDENT = "PEN";
 35          public static final String FACE_QUALITY = "FAS";
 36     }
 37
 38     /**
 39      * @param args the command line arguments
 40      */
 41     public static void main(String[] args) {
 42
 43         try{
 44
 45             if( (args.length == 3) && (String.valueOf(args[2]).split("
").length >= 2) ){
 46
 47                 FileInputStream fis = new FileInputStream(new File(args[0]));
 48                 byte[] image = new byte[fis.available()];
 49                 fis.read(image);
 50                 fis.close();
 51
 52                 List<Object> result = certiface(EMP_COD, FIL_COD, USER_ID,
 53                     USER_PASS, Long.parseLong(String.valueOf(args[1])),
 54                         String.valueOf(args[2]), "01/01/1960", null, null,
null,
 55                             null, null, null, null, null, null, image);
 56
 57                 System.out.println("\nPROTOCOLO:
".concat(result.get(0).toString()));
 58
 59                 if(result.get(1).toString().equals(Service.VALIDATION)){
 60                     if(result.get(2).toString().equals(Result.POSITIVE)){
 61                         System.out.println("CERTIFICACAO POSITIVA !");
 62                     }else
if(result.get(2).toString().equals(Result.NEGATIVE)){
 63                         System.out.println("CERTIFICACAO NEGATIVA !");
 64                     }
 65                 }else
if(result.get(1).toString().equals(Service.INDENTIFICATION)){
 66                     if(result.get(2).toString().equals(Result.SUCCESS)){
 67                         System.out.println("CADASTRO POSITIVO !");
 68                     }else
if(result.get(2).toString().equals(Result.PENDENT)){
 69                         System.out.println("CADASTRO PENDENTE !");
 70                         System.out.println("SIMILARES: ");
 71                         AnyTypeArray similars = (AnyTypeArray)
result.get(3);
 72                         for(Object similar : similars.getItem()){
```

```
 73                             AnyTypeArray sim = (AnyTypeArray) similar;
 74                             String name = sim.getItem().get(2).toString();
 75                             String similarity =
sim.getItem().get(5).toString();
 76                             System.out.println("SIMILAR: ".
 77                                 concat(name).concat(" ").
 78                                     concat(String.valueOf(Float.parseFloat(
 79                                         similarity)*100)).
 80                                             concat("%"));
 81                         }
 82                     }
 83                 }else if(result.get(1).toString().equals(Service.ERROR)){
 84                     System.out.println("ERRO:
".concat(result.get(2).toString()));
 85
 86                     if(result.get(2).toString().equals(Result.FACE_QUALITY))
{
 87                         AnyTypeArray fQuality = (AnyTypeArray)
result.get(3);
 88                         if(fQuality != null){
 89                             System.out.println("RESPOSTA QUALIDADE DA
IMAGEM: ".
 90
concat(fQuality.getItem().get(0).toString()));
 91                         }
 92                     }
 93
 94                 }
 95
 96             }else{
 97
 98                 System.out.println("\nUSE: java -jar JCertifaceConsole.jar
<image full path> <cpf> <fist and last name>");
 99                 System.out.println("EX: java -jar JCertifaceConsole.jar
/dados/example/image.jpg 00000014141 \"jon jones\"");
100
101             }
102
103         }catch(Exception e){
104             e.printStackTrace();
105         }
106
107     }
108
109     private static java.util.List<java.lang.Object> certiface(
110         java.lang.Long empCodAuth, java.lang.Long filCodAuth,
111         java.lang.String loginAuth, java.lang.String passwordAuth,
112         java.lang.Long cpf, java.lang.String name,
113         java.lang.String birth, java.lang.String rg,
114         java.lang.String dtExpedition, java.lang.String motherName,
115         java.lang.String phone, java.lang.String cellPhone,
116         java.lang.String address, java.lang.String city,
117         java.lang.String state, java.lang.String mail,
118                     byte[] image) throws MalformedURLException {
119         br.com.certifacews.service.CertiFaceWSService service =
120                 new br.com.certifacews.service.CertiFaceWSService(
121                     new URL(SERVICE_URL));
```

```
122         br.com.certifacews.service.CertiFaceWS port =
service.getCertiFaceWSPort();
123         return port.certiface(empCodAuth, filCodAuth, loginAuth,
passwordAuth, cpf, name, birth, rg, dtExpedition, motherName, phone,
124                     cellPhone, address, city, state, mail, image);
125     }
126
127 }
```

Na linha 1 até a 8 importamos as bibliotecas/classes necessárias para o funcionamento do exemplo. Já da linha 16 até 35 são declaradas as constantes.

Na linha 41 a classe principal (main) inicia; na linha 47 a imagem é carregada em memória para posteriormente ser enviada ao Certiface junto aos parâmetros informados na linha de comando.

O retorno é tratado, e na linha 57 é exibido o numero do protocolo e, logo em seguida na linha 59 se o processamento executado no Certiface foi uma certificação, então é apresentado a mensagem "CERTIFICAÇÃO POSITIVA OU NEGATIVA".

Caso contrário na linha 64 é exibido o tipo de cadastro efetuado, ou seja, um cadastro positivo ou pendente. Caso o cadastro estiver com o status pendente, os similares são exibidos com as respectivas taxas de similaridade biométrica.

Para finalizar na linha 83, se houver alguma inconsistência, o mesmo é informado com o seu correspondente erro.

O campo senha é um hash MD5 da senha, ou seja, um algoritmo de hash de 128 bits unidirecional desenvolvido pela RSA Data Security, Inc., descrito na RFC 1321, e muito utilizado por softwares com protocolo ponto-a-ponto (P2P, ou Peer-to-Peer, em inglês) na verificação de integridade de arquivos e logins.

A seguir, um exemplo em java para criar string MD5:

```
import java.security.*;
import java.math.*;

public class MD5 {
    public static void main(String args[]) throws Exception{
```

```
        String s="Texto de Exemplo";
        MessageDigest m=MessageDigest.getInstance("MD5");
        m.update(s.getBytes(),0,s.length());
        System.out.println("MD5: "+new BigInteger(1,m.digest()).toString(16));
    }
}
```

Execução do programa exemplo

A seguir executaremos o programa exemplo para simular alguns dos principais cenários, ou seja, a certificação negativa, cadastro positivo e cadastro pendente.

Inicialmente efetuaremos o cadastro com um novo CPF e uma face já presente no Certiface, com isto estaremos simulando um cadastro com fraude.

```
$ java -jar JCertifaceConsole.jar usuario1.jpg  30124664601 "ALESSANDRO FARIA"

PROTOCOLO: 20140016768
CADASTRO PENDENTE !
SIMILARES:
SIMILAR: LEANDRO TESTE 99.9715%
SIMILAR: LEANDRO TESTE 99.970695%
```

Após o processamento, automaticamente o gestor da empresa receberá um e-mail e um SMS contendo o seguinte conteúdo:

Agora utilizaremos a imagem asiatico1.jpg para simular uma certificação negativo onde simularemos uma fraude com um documento já cadastrado:

```
$ java -jar JCertifaceConsole.jar asiatico1.jpg  30124664601 "ALESSANDRO FARIA"

PROTOCOLO: 20140016769
CERTIFICACAO NEGATIVA !
```

No exemplo abaixo, utilizaremos a imagem perfil.jpg para simular uma certificação ou cadastro com imagem inconsistente:

```
$ java -jar JCertifaceConsole.jar perfil.jpg  30124664601 "ALESSANDRO FARIA"

PROTOCOLO: 20140016770
ERRO: FAS
```

Para finalizar, efetuaremos um cadastro convencional sem inconsistência alguma:

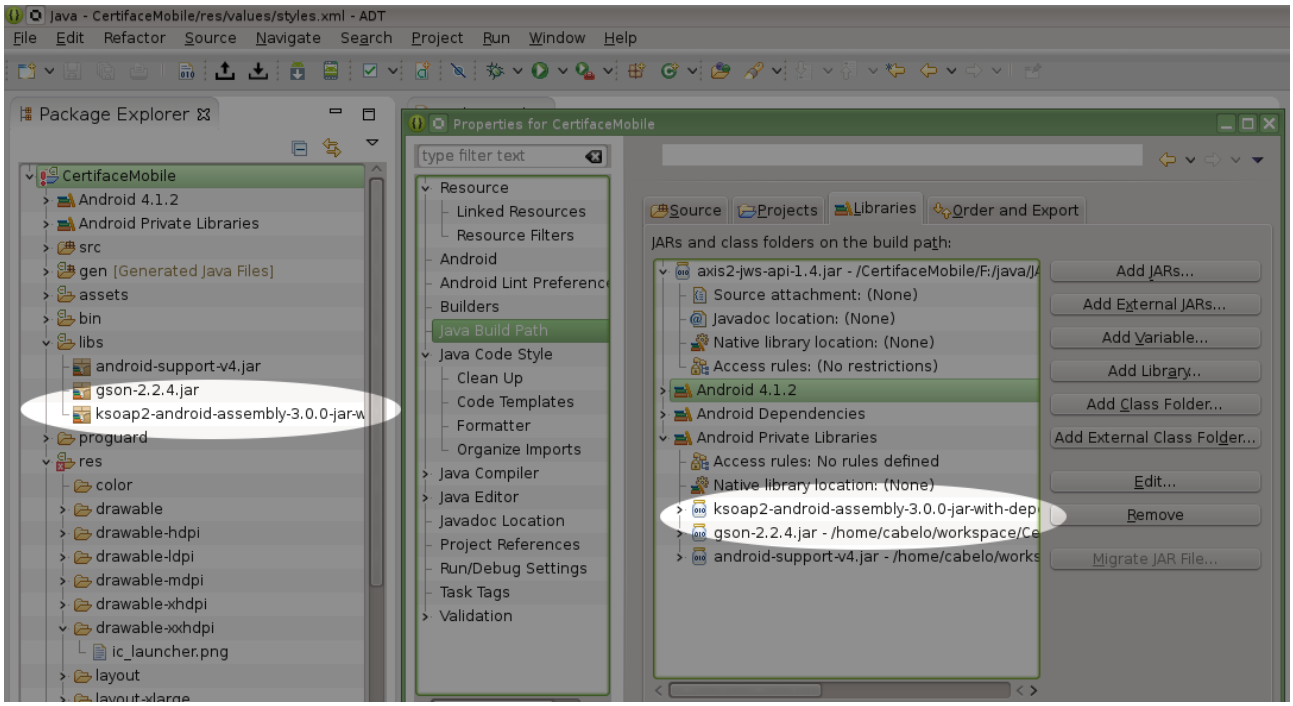
```
$ java -jar JCertifaceConsole.jar alguem.jpg  97458097566 "ANTONIO DA SILVA"

PROTOCOLO: 20140016771
CADASTRO POSITIVO !
```

Outros exemplos

Nos próximos capítulos veremos alguns exemplos de integração em outras linguagens de programação. Maiores detalhes entrar em contato com o suporte de desenvolvimento/integração do Certiface.

Integração na plataforma Android

Para os programadores java, a plataforma Android é bem similar o tratamento de sintaxe, entretanto a tecnologia sugerida é a biblioteca ksoap2. A biblioteca Ksoap2-android é uma biblioteca de SOAP leve e eficiente para a plataforma Android. Faça o download em http://code.google.com/p/ksoap2-android/ e inclua o JAR no seu projeto como na figura a seguir:



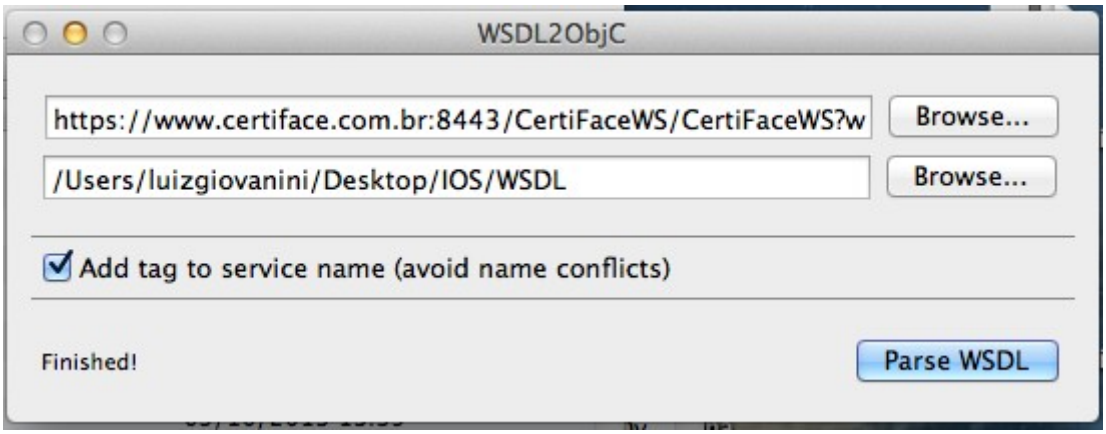
O código fonte abaixo traz o exemplo de utilização e /ou integração com o serviço Certiface, utilizando a biblioteca KSOAP2.

```
     public ResponseBean certiface(String cpf, String name, String birth,
     byte[] image) throws IOException, XmlPullParserException, SoapFault {

           SoapObject request = null;
           SoapSerializationEnvelope envelope = null;
           HttpTransportSE androidHttpTransport = null;
           SoapObject response = null;
           request = new SoapObject(namespace, CERTIFACE_METHOD_NAME);
           request.addProperty("empCodAuth", String.valueOf(empresa));
           request.addProperty("filCodAuth", String.valueOf(filial));
           request.addProperty("loginAuth", cript(Cipher.ENCRYPT_MODE,
usuario.getBytes(),
                 String.valueOf(empresa), String.valueOf(filial)));
           request.addProperty("passwordAuth", cript(Cipher.ENCRYPT_MODE,
senha.getBytes(),
                 String.valueOf(empresa), String.valueOf(filial), usuario));
           request.addProperty("cpf", cript(Cipher.ENCRYPT_MODE,
cpf.getBytes(),
                 String.valueOf(empresa), String.valueOf(filial), usuario,
senha));
           request.addProperty("name", cript(Cipher.ENCRYPT_MODE,
name.getBytes(),
                 String.valueOf(empresa), String.valueOf(filial), usuario,
senha, cpf));

           request.addProperty("birth", cript(Cipher.ENCRYPT_MODE,
birth.getBytes(),
                 String.valueOf(empresa), String.valueOf(filial), usuario,
senha, cpf, name));
```

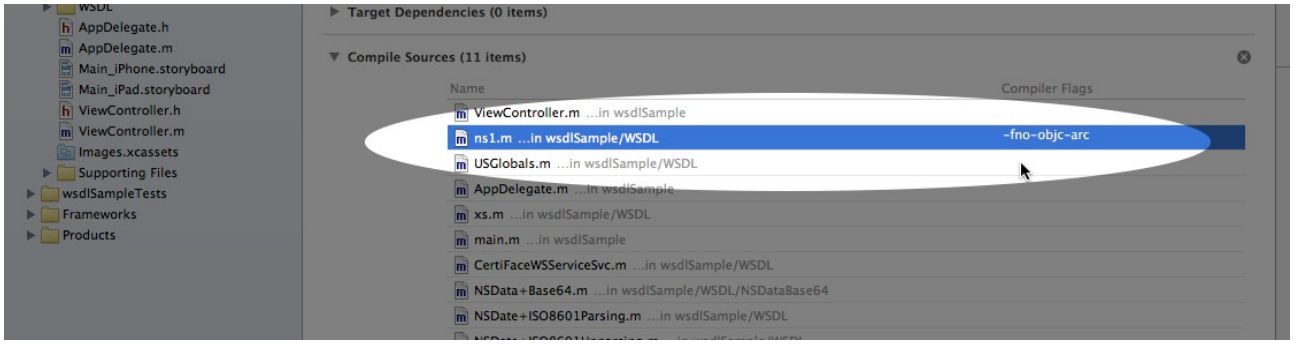
```
            request.addProperty("image",  image, String.valueOf(empresa),
String.valueOf(filial), usuario, senha, cpf, birth, name);
            envelope = new SoapSerializationEnvelope(SoapEnvelope.VER11);
            envelope.dotNet = false;
            new MarshalBase64().register(envelope);
            envelope.setOutputSoapObject(request);
            androidHttpTransport = new HttpTransportSE(URL,
            Integer.parseInt(ConfigMobileBean.getInstance().getTimeOut()));
            androidHttpTransport
                  .call(("\"".concat(soapAction.concat(CERTIFACE_ACTION)))
                        .concat("\""), envelope);
            if(envelope.bodyIn instanceof SoapFault){
                  throw new SoapFault();
            }
            response = (SoapObject) envelope.bodyIn;
            return new ResponseBean(response);
      }
```

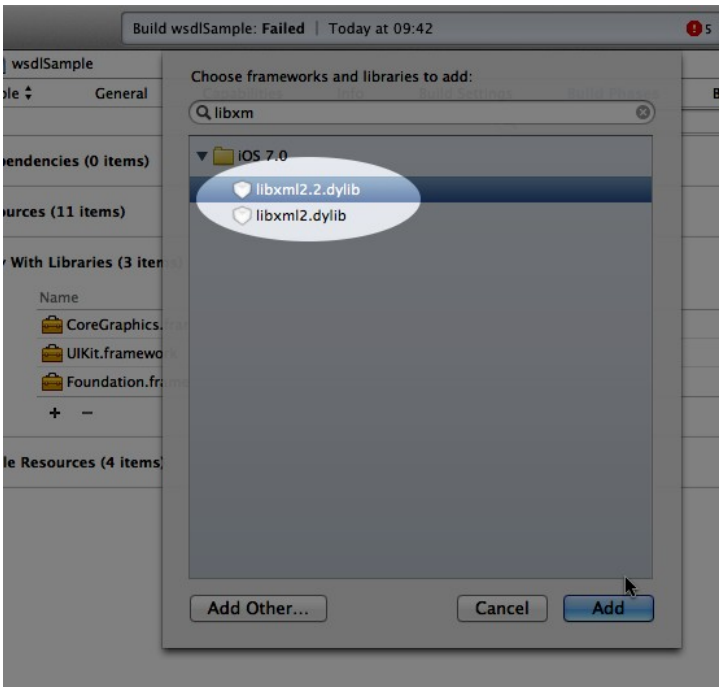
Integração na plataforma iOS

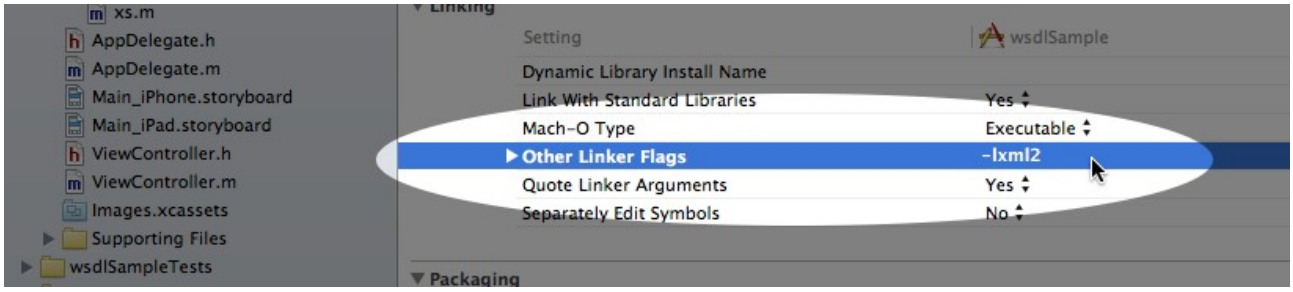
Para efetuar a integração na plataforma iOS, utilizamos o utilitário WSDL2Objc. Para criar as classes Objc a partir do WSDL, devemos apenas informar o caminho do arquivo ou URL WSDL e logo em seguida o local de criação dos arquivos fontes em Objc com as Classes para acesso ao serviço Certiface.

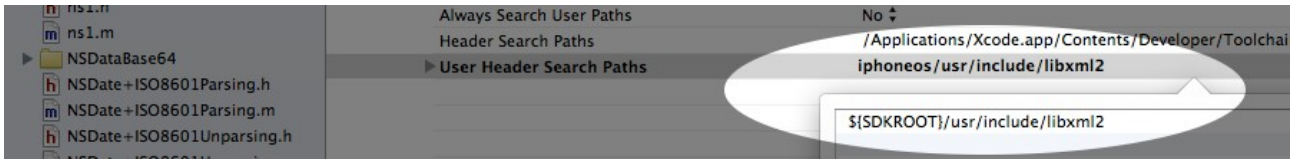


Após a criação dos arquivos, devemos incluir os arquivos em Objc criado pelos WDSL2Objc no projeto. Não devemos esquecer de configurar as diretivas de compilação e/ou dependência da libxml2. A seguir algumas imagens para referencia operacional como também um código fonte exemplo de chamada do serviço.


AppDelegate.h
AppDelegate.m
Main_iPhone.storyboard
Main_iPad.storyboard
ViewController.h
ViewController.m
Images.xcassets
Supporting Files
wsdlSampleTests
Frameworks
Products
Target Dependencies (0 items)
Compile Sources (11 items)
Name
Compiler Flags
ViewController.m ...in wsdlSample
ns1.m ...in wsdlSample/WSDL
-fno-objc-arc
USGlobals.m ...in wsdlSample/WSDL
AppDelegate.m ...in wsdlSample
xs.m ...in wsdlSample/WSDL
main.m ...in wsdlSample
CertiFaceWSServiceSvc.m ...in wsdlSample/WSDL
NSData+Base64.m ...in wsdlSample/WSDL/NSDataBase64
NSDate+ISO8601Parsing.m ...in wsdlSample/WSDL


Build wsdlSample: Failed | Today at 09:42
wsdlSample
General
Choose frameworks and libraries to add:
libxm
iOS 7.0
libxml2.2.dylib
libxml2.dylib
Add Other...
Cancel
Add


xs.m
AppDelegate.h
AppDelegate.m
Main_iPhone.storyboard
Main_iPad.storyboard
ViewController.h
ViewController.m
Images.xcassets
Supporting Files
wsdlSampleTests
Setting
wsdlSample
Dynamic Library Install Name
Link With Standard Libraries
Yes
Mach-O Type
Executable
Other Linker Flags
-lxml2
Quote Linker Arguments
Yes
Separately Edit Symbols
No
Packaging


ns1.m
NSDataBase64
NSDate+ISO8601Parsing.h
NSDate+ISO8601Parsing.m
NSDate+ISO8601Unparsing.h
Always Search User Paths
No
Header Search Paths
/Applications/Xcode.app/Contents/Developer/Toolchai
User Header Search Paths
iphoneos/usr/include/libxml2
${SDKROOT}/usr/include/libxml2

Partindo do principio que o leitor tem experiência no desenvolvimento na plataforma iOS, abaixo veremos um exemplo em Obj-c de como integrar o serviço Certiface com a chamada WSDL utilizando as ferramentas mencionada anteriormente.

```
//  ViewController.m
//  wsdlSample
//  Created by luiz giovanini on 13/06/14.
//  Copyright (c) 2014 luiz giovanini. All rights reserved.

#import "ViewController.h"
#import "WSDL/CertiFaceWSServiceSvc.h"
static NSString* wsdlURL =
@"https://www.certiface.com.br:8443/CertiFaceWS/CertiFaceWS?wsdl";
@interface ViewController ()
@end
@implementation ViewController
@synthesize lblAbout;
- (void)viewDidLoad
{
    [super viewDidLoad];
    CertiFaceWSPortBinding *binding = [[CertiFaceWSServiceSvc
CertiFaceWSPortBinding] initWithAddress:wsdlURL];
    CertiFaceWSServiceSvc_about *request = [[CertiFaceWSServiceSvc_about alloc]
init];
    @try {
        CertiFaceWSPortBindingResponse *resp = [binding
aboutUsingParameters:request];

        NSError *responseError = resp.error;
        if (responseError!=nil){
            NSLog(@"%@",@"FALHA AO ACESSAR WEBSERVICE");
        }
        for (id mine in resp.bodyParts)
        {
            if ([mine isKindOfClass:[CertiFaceWSServiceSvc_aboutResponse
class]])
            {
                lblAbout.text = (NSString *) [mine return_];
            }
        }
    }
    @catch (NSException *exception) {
    }
      // Do any additional setup after loading the view, typically from a nib.
}
- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}
@end
```

Integração na liguagem C/C++

Para efetuar a integração na linguagem C, embora estão disponíveis diversas alternativas, utilizaremos a biblioteca GSOAP como exemplo. Primeiramente obtenha a biblioteca em: http://sf.net/projects/gsoap2/. Após o download descompacte a biblioteca e execute o comando ./configure, make e make install. Se tudo estiver funcionando corretamente teremos a saída a seguir:

```
# ./configure
# make
make  all-recursive
make[1]: Entrando no diretório `/dados/programas/gsoap-2.8'
Making all in gsoap
make[2]: Entrando no diretório `/dados/programas/gsoap-2.8/gsoap'
make  all-recursive
make[3]: Entrando no diretório `/dados/programas/gsoap-2.8/gsoap'
Making all in .
make[4]: Entrando no diretório `/dados/programas/gsoap-2.8/gsoap'
make[4]: Nada a ser feito para `all-am'.
make[4]: Saindo do diretório `/dados/programas/gsoap-2.8/gsoap'
Making all in src
make[4]: Entrando no diretório `/dados/programas/gsoap-2.8/gsoap/src'
gcc -DHAVE_CONFIG_H -I. -I../..    -DWITH_BISON -DWITH_FLEX  -
DSOAPCPP_IMPORT_PATH="\"/usr/local/share/gsoap/import\"" -DLINUX -g -O2 -MT
soapcpp2-symbol2.o -MD -MP -MF .deps/soapcpp2-symbol2.Tpo -c -o soapcpp2-
symbol2.o `test -f 'symbol2.c' || echo './'`symbol2.c
  …   …   …   …   …   …   …   …   …   …   …   …   …   …
  …   …   …   …   …   …   …   …   …   …   …   …   …   …
  …   …   …   …   …   …   …   …   …   …   …   …   …   …
wsdlC.cpp:27132:21: note: in expansion of macro ‘SOAP_NEW_ARRAY’
  { cp->ptr = (void*)SOAP_NEW_ARRAY(std::vector<xs__element * >, n);
                     ^
mv -f .deps/wsdl2h-wsdl.Tpo .deps/wsdl2h-wsdl.Po
mv -f .deps/wsdl2h-schema.Tpo .deps/wsdl2h-schema.Po
mv -f .deps/wsdl2h-types.Tpo .deps/wsdl2h-types.Po
mv -f .deps/wsdl2h-service.Tpo .deps/wsdl2h-service.Po
mv -f .deps/wsdl2h-wsdlC.Tpo .deps/wsdl2h-wsdlC.Po
g++  -DWITH_OPENSSL -DWITH_GZIP -g -O2 -L.  -o wsdl2h wsdl2h-wsdl2h.o wsdl2h-
wsdl.o wsdl2h-schema.o wsdl2h-types.o wsdl2h-service.o wsdl2h-soap.o wsdl2h-
mime.o wsdl2h-wsp.o wsdl2h-wsdlC.o ../../gsoap/libgsoapssl++.a -lssl -lcrypto -
lz
make[5]: Saindo do diretório `/dados/programas/gsoap-2.8/gsoap/wsdl'
make[4]: Saindo do diretório `/dados/programas/gsoap-2.8/gsoap/wsdl'
make[3]: Saindo do diretório `/dados/programas/gsoap-2.8/gsoap'
make[2]: Saindo do diretório `/dados/programas/gsoap-2.8/gsoap'
make[2]: Entrando no diretório `/dados/programas/gsoap-2.8'
make[2]: Saindo do diretório `/dados/programas/gsoap-2.8'
make[1]: Saindo do diretório `/dados/programas/gsoap-2.8'
```

```
# make install
```

Para criar a aplicação cliente, os comando wsdl2h e soapcpp2 conforme o exemplo a seguir:

```
$ wsdl2h -c -o rcx.h  "http://www.certiface.com.br/CertiFaceWS/CertiFaceWS?wsdl"

**  The gSOAP WSDL/Schema processor for C and C++, wsdl2h release 2.8.15
**  Copyright (C) 2000-2013 Robert van Engelen, Genivia Inc.
**  All Rights Reserved. This product is provided "as is", without any warranty.
**  The wsdl2h tool is released under one of the following two licenses:
**  GPL or the commercial license by Genivia Inc. Use option -l for details.

Saving rcx.h

Cannot open file 'typemap.dat'
Problem reading type map file 'typemap.dat'.
Using internal type definitions for C instead.


Connecting to 'http://www.certiface.com.br/CertiFaceWS/CertiFaceWS?wsdl' to
retrieve WSDL/XSD...
Connected, receiving...

Connecting to 'http://10.0.1.103:8080/CertiFaceWS/CertiFaceWS?xsd=1' to retrieve
schema...
Connected, receiving...
Done reading 'http://10.0.1.103:8080/CertiFaceWS/CertiFaceWS?xsd=1'

Connecting to 'http://10.0.1.103:8080/CertiFaceWS/CertiFaceWS?xsd=2' to retrieve
schema...
Connected, receiving...
Done reading 'http://10.0.1.103:8080/CertiFaceWS/CertiFaceWS?xsd=2'
Done reading 'http://www.certiface.com.br/CertiFaceWS/CertiFaceWS?wsdl'

To complete the process, compile with:
> soapcpp2 rcx.h
```

Para completar o processo execute o comando soapcpp2 -c.

```
$ soapcpp2 -c rcx.h

**  The gSOAP code generator for C and C++, soapcpp2 release 2.8.15
**  Copyright (C) 2000-2013, Robert van Engelen, Genivia Inc.
**  All Rights Reserved. This product is provided "as is", without any warranty.
**  The soapcpp2 tool is released under one of the following two licenses:
**  GPL or the commercial license by Genivia Inc.

Saving soapStub.h annotated copy of the source input
Saving soapH.h interface declarations
Using ns1 service name: CertiFaceWSPortBinding
Using ns1 service style: document
```

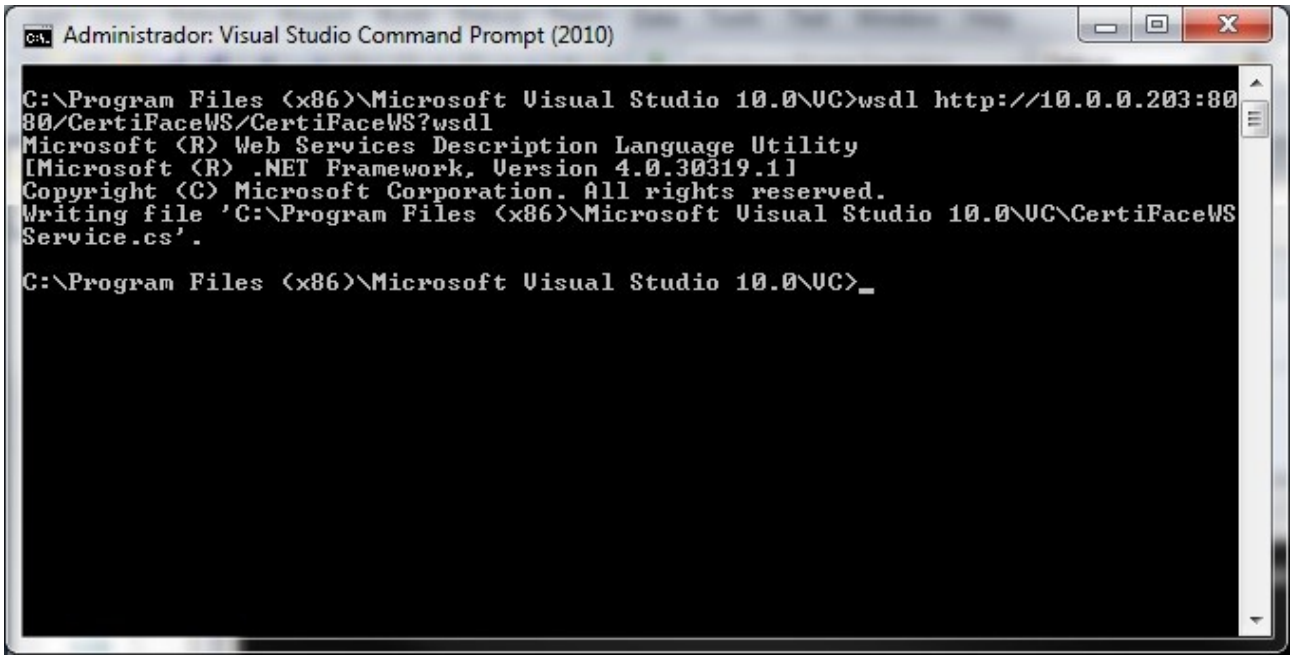
```
Using ns1 service encoding: literal
Using ns1 service location: http://10.0.1.103:8080/CertiFaceWS/CertiFaceWS
Using ns1 schema namespace: http://service.certifacews.com.br/
Saving CertiFaceWSPortBinding.login.req.xml sample SOAP/XML request
Saving CertiFaceWSPortBinding.login.res.xml sample SOAP/XML response
Saving CertiFaceWSPortBinding.listDocuments.req.xml sample SOAP/XML request
Saving CertiFaceWSPortBinding.listDocuments.res.xml sample SOAP/XML response
  …   …   …   …   …   …   …   …   …   …   …   …   …   …
  …   …   …   …   …   …   …   …   …   …   …   …   …   …
  …   …   …   …   …   …   …   …   …   …   …   …   …   …

Saving CertiFaceWSPortBinding.DocumentsExist.req.xml sample SOAP/XML request
Saving CertiFaceWSPortBinding.DocumentsExist.res.xml sample SOAP/XML response
Saving CertiFaceWSPortBinding.nsmap namespace mapping table
Saving soapClient.c client calling stubs
Saving soapClientLib.c client stubs with serializers (use only for libs)
Saving soapServer.c server request dispatcher
Saving soapServerLib.c server request dispatcher with serializers (use only for
libs)
Saving soapC.c serializers
Compilation successful
$
```

Agora com todos arquivos criados com sucesso, criaremos um programa exemplo em C acessando o método About:

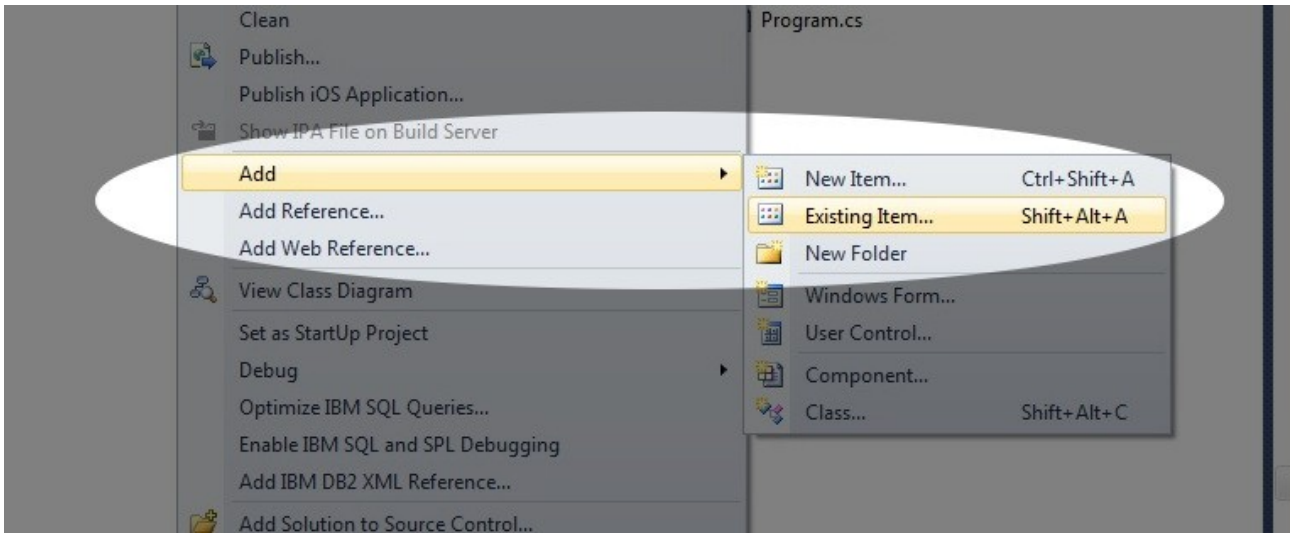
```
#include "soapH.h"
#include "CertiFaceWSPortBinding.nsmap"

int main(int argc, char **argv)
{

   struct soap soap;

   struct ns1__about *_req;
   struct ns1__aboutResponse *_res;

   soap_init1(&soap, SOAP_XML_INDENT);
   soap_call___ns1__about(&soap,argv[1], "", _req, _res);
   if (soap.error)
   {
      printf("Erro:\n");
      soap_print_fault(&soap, stderr);
      exit(1);
   }
   else
   printf("result = %s\n", _res->return_);

   soap_destroy(&soap);
   soap_end(&soap);
   soap_done(&soap);
   return 0;
}
```

Integração na linguagem C#

Para efetuar a integração na linguagem C#, utilizaremos o utilitário wsdl precedido da localização da URL ou arquivo wsdl para criarmos as classes e respectivos métodos no arquivo fonte CertiFaceWSService.cs. Veja o exemplo a seguir:



Agora abra o seu projeto em C# no Visual Studio, clique o botão direito do mouse para obter o menu de opções. Primeiramente selecione o item “ADD” e após o sub-item “Existing Item...” para adicionar o arquivo CertiFaceWSService.cs.



Para exemplificar a integração e/ou funcionalidade logo abaixo, um código fonte que acessa o serviço Certiface consumindo o WebService.

```
using System;
using System.Collections.Generic;
using System.Text;
using System.IO;
namespace TesteWSDL
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Bem  vindo...");
            Console.WriteLine();
            Console.WriteLine("  <<  Funções de teste  >>");
            Console.WriteLine("about:               Metodo About CertiFace.");
            Console.WriteLine("assessment:          Metodo assesment.");
            Console.WriteLine("cls:                 Limpa a tela do Console.");
            Console.WriteLine("sair:                Para sair do programa.");
            Console.WriteLine();

            // Instancia Objeto CertiFaceWSService de coonexão com wsdl
            CertiFaceWSService services = new CertiFaceWSService();
            while (true)
            {
                string funcao = Console.ReadLine();
                switch (funcao.ToLower())
                {
                    case "assessment":
                        try
                        {
                            MemoryStream stream = new MemoryStream();
                            Properties.Resources.image.Save(stream,
System.Drawing.Imaging.ImageFormat.Jpeg);
                            byte[] img = stream.ToArray();
                            anyTypeArray[] resultArray = services.assessment( 1,
true, 1, true, "rica", Crypto.CreateMD5Hash("rica"), img, 14141, true);
                            foreach (anyTypeArray anyType in resultArray)
```

```
                            {
                                Console.WriteLine(String.Format("{0}:{1}",
anyType.item[0], anyType.item[1]));
                            }
                            Console.WriteLine();
                        }
                        catch (Exception ex)
                        {
                            Console.WriteLine(ex.Message);
                            Console.WriteLine();
                        }

                        break;
                    case "about":
                        string result = string.Empty;

                        try
                        {
                            result = services.about();
                        }
                        catch (Exception ex)
                        {
                            Console.WriteLine(ex.Message);
                            Console.WriteLine();
                        }

                        Console.WriteLine(result);
                        Console.WriteLine();

                        break;
                    case "cls":
                        Console.Clear();
                        Console.WriteLine("Bem  vindo...");
                        Console.WriteLine();
                        break;
                    case "sair": return;
                    default:
                        Console.WriteLine("");
                        Console.WriteLine("  <<  Funções de teste  >>");
                        Console.WriteLine("about:               Metodo About
CertiFace.");
                        Console.WriteLine("assessment:          Metodo
assesment.");
                        Console.WriteLine("cls:                 Limpa a tela do
Console.");
                        Console.WriteLine("sair:                Para sair do
programa.");
                        Console.WriteLine();
                        break;
                }
            }
        }
    }
}
```

Integração na linguagem PHP

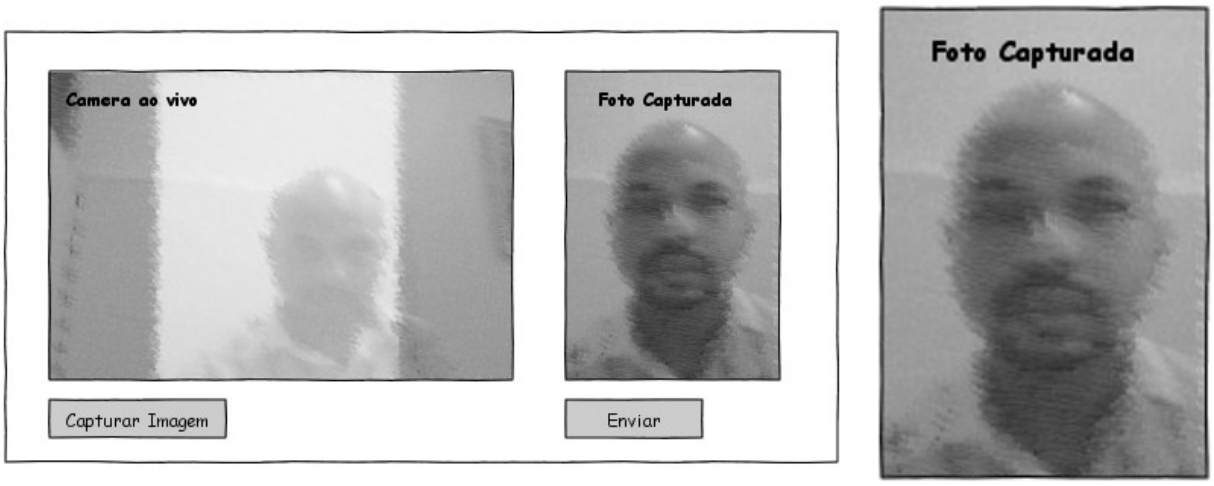
Para efetuar a integração na linguagem PHP, devemos instalar as dependências php5-xmlreader, php5-xmlwriter, php5-xmlrpc e php5-curl. A seguir um exemplo de acesso ao serviço WSDL:

```
<?php
if ($argc != 2) {
    die("Usage: ident.php <path-imagem> \n");
}
$encryptedDate = file_get_contents( $argv[1]);
try {
   $client = new
SoapClient('http://www.certiface.com.br/CertiFaceWS/CertiFaceWS?wsdl',
array('trace' => true, 'exceptions' => true));
   $client->__setLocation("http://www.certiface.com.br/CertiFaceWS/CertiFaceWS?
wsdl");

   $result = $client->about("");
   var_dump($result);
}
catch (SoapFault $e) {
 print $e->faultcode.' - '.$e->faultstring;
}

?>
```

Vídeo Captura em Paginas WEB

O Certiface disponibiliza um componente e código fonte exemplo, de como efetuar vídeo captura via Web. Este componente tem como principal objetivo demonstrar algumas de diversas maneiras de como instanciar um dispositivo de captura.

A figura a seguir exibe a parte operacional do componente exemplo. O primeiro botão recorta a imagem e à exibe ao lado do box com vídeo ao vivo. Já o segundo botão invoca uma função interna em Javascript denominada “image” e passando como parâmetro uma imagem em base64. Como resultado, teremos a exibição da imagem recém-capturada.



A seguir um exemplo da função que invoca a chamada javascript, extraída do código fonte do componente de captura escrito em Apache Flex:

```
/* Método de envio para o servidor */
public function sendImage():void
{
   Security.allowDomain("*");
   var Encoder:JPEGEncoder = new JPEGEncoder(85);
   try
   {
      if (capturaFoto)
      {
         btnCapturar.enabled = false;
         load = new Loading();
         load.Message = MessageSend;
         PopUpManager.addPopUp(load, this, true);
         PopUpManager.centerPopUp(load);
         imageArray = Encoder.encode(snapshotCrop);
         if (isSendToScript) {
            if (ExternalInterface.available) {
               ExternalInterface.call("image",
Base64.encodeByteArray(imageArray));
            } else {
               Alert.show("Impossivel o acesso a pagina.", "CertiFace");
            }
            load.closePopUp();
            btnCapturar.enabled = true;
            btnEnviar.enabled = true;
         } else {
            var myFileReference:FileReference = new FileReference();
            var loader:URLLoader = new URLLoader();
            var header:URLRequestHeader = new URLRequestHeader("Content-type",
"application/octet-stream");
            var saveJPG:URLRequest = new URLRequest(ServletSendImageURL);
            saveJPG.requestHeaders.push(header);
            saveJPG.method = URLRequestMethod.POST;
            saveJPG.data = imageArray;
            loader.addEventListener(Event.COMPLETE, handleComplete);
            loader.addEventListener(IOErrorEvent.IO_ERROR, onIOError);
            loader.load(saveJPG);
         }
      }
      else
      {
         mess = MessagePleaseCap;
         Alert.show(mess, "CertiFace");
      }

   }
   catch (obj:Error)
   {
      Alert.show(obj.message, "CertiFace");
      load.closePopUp();
   }
}
```

A seguir um exemplo da função que invoca a chamada javascript, extraída do

código fonte do componente de captura escrito em HTML5:

```
function sendImagePost_cam()
{
    var data = new Object();

    if (img !== null) {

        setTimeout(function(){ lockCamUI(); }, 0);

        data = img.src.replace(/^data:image\/(png|jpeg);base64,/, "");

        jQuery.ajax({
            contentType: 'image/jpeg',
            type: "POST",
            async: true,
            url: url,
            data: data,
            cache: false,
            dataType: "json",
            success: function (json) {

                alert('Altere a url no arquivo \'camera.js\'');

            },
            error: function (json) {

                        alert('Altere a url no arquivo \'camera.js\'');

            }

        });
        //stop();
    } else {
        alert(messagePleaseCap);
    }
}


function successCallback(stream) {
    if (video.mozSrcObject !== undefined) {
        video.mozSrcObject = stream;
    } else {
        video.src = (window.URL && window.URL.createObjectURL(stream)) ||
stream;
    }
    ;
    localMediaStream = stream;
    video.play();
}
```

Componentes, Arquivos fontes e exemplos

Os códigos fontes utilizados neste documento pode ser obtido através de um link de internet ou média previamente solicitado para o email *contato@oititec.br*. Na pasta examples contém os códigos fontes em Flex, iOS, Java e dotNet. A seguir a estrutura de diretórios:

```
├── examples
│   ├── example.Flex
│   │   ├── Flash.Componente
│   │   └── source
│   ├── example.iOS
│   ├── example.Java
│   │   ├── axis
│   │   ├── JCertifaceConsole
│   │   ├── md5
│   │   └── run
│   └── example.Net
└── Outros
```

Estrutura WSDL

A seguir a estrutura XML do arquivo WSDL:

```
<?xml version='1.0' encoding='UTF-8'?><!-- Published by JAX-WS RI at http://jax-
ws.dev.java.net. RI's version is JAX-WS RI 2.2-hudson-740-. --><!-- Generated by
JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-WS RI 2.2-hudson-
740-. --><definitions xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-wssecurity-utility-1.0.xsd" xmlns:wsp="http://www.w3.org/ns/ws-
policy" xmlns:wsp1_2="http://schemas.xmlsoap.org/ws/2004/09/policy"
xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tns="http://service.certifacews.com.br/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="http://schemas.xmlsoap.org/wsdl/"
targetNamespace="http://service.certifacews.com.br/" name="CertiFaceWSService">
<types>
<xsd:schema>
<xsd:import namespace="http://service.certifacews.com.br/"
schemaLocation="http://10.0.1.103:8080/CertiFaceWS/CertiFaceWS?xsd=1" />
</xsd:schema>
<xsd:schema>
<xsd:import namespace="http://jaxb.dev.java.net/array"
schemaLocation="http://10.0.1.103:8080/CertiFaceWS/CertiFaceWS?xsd=2" />
```

```
</xsd:schema>
</types>
<message name="login">
<part name="parameters" element="tns:login" />
</message>
<message name="loginResponse">
<part name="parameters" element="tns:loginResponse" />
</message>
<message name="listDocuments">
<part name="parameters" element="tns:listDocuments" />
</message>
<message name="listDocumentsResponse">
<part name="parameters" element="tns:listDocumentsResponse" />
</message>
<message name="listOtherDocuments">
<part name="parameters" element="tns:listOtherDocuments" />
</message>
<message name="listOtherDocumentsResponse">
<part name="parameters" element="tns:listOtherDocumentsResponse" />
</message>
<message name="about">
<part name="parameters" element="tns:about" />
</message>
<message name="aboutResponse">
<part name="parameters" element="tns:aboutResponse" />
</message>
<message name="certiface">
<part name="parameters" element="tns:certiface" />
</message>
<message name="certifaceResponse">
<part name="parameters" element="tns:certifaceResponse" />
</message>
<message name="assessment">
<part name="parameters" element="tns:assessment" />
</message>
<message name="assessmentResponse">
<part name="parameters" element="tns:assessmentResponse" />
</message>
<message name="findId">
<part name="parameters" element="tns:findId" />
</message>
<message name="findIdResponse">
<part name="parameters" element="tns:findIdResponse" />
</message>
<message name="addDocuments">
<part name="parameters" element="tns:addDocuments" />
</message>
<message name="addDocumentsResponse">
<part name="parameters" element="tns:addDocumentsResponse" />
</message>
<message name="updateDocuments">
<part name="parameters" element="tns:updateDocuments" />
</message>
<message name="updateDocumentsResponse">
<part name="parameters" element="tns:updateDocumentsResponse" />
</message>
<message name="addDocumentsBySFTP">
```

```
<part name="parameters" element="tns:addDocumentsBySFTP" />
</message>
<message name="addDocumentsBySFTPResponse">
<part name="parameters" element="tns:addDocumentsBySFTPResponse" />
</message>
<message name="addDocumentsByObjectArray">
<part name="parameters" element="tns:addDocumentsByObjectArray" />
</message>
<message name="addDocumentsByObjectArrayResponse">
<part name="parameters" element="tns:addDocumentsByObjectArrayResponse" />
</message>
<message name="addOtherDocumentsByObjectArray">
<part name="parameters" element="tns:addOtherDocumentsByObjectArray" />
</message>
<message name="addOtherDocumentsByObjectArrayResponse">
<part name="parameters" element="tns:addOtherDocumentsByObjectArrayResponse" />
</message>
<message name="updateDocumentsBySFTP">
<part name="parameters" element="tns:updateDocumentsBySFTP" />
</message>
<message name="updateDocumentsBySFTPResponse">
<part name="parameters" element="tns:updateDocumentsBySFTPResponse" />
</message>
<message name="DocumentsExist">
<part name="parameters" element="tns:DocumentsExist" />
</message>
<message name="DocumentsExistResponse">
<part name="parameters" element="tns:DocumentsExistResponse" />
</message>
<portType name="CertiFaceWS">
<operation name="login">
<input
wsam:Action="http://service.certifacews.com.br/CertiFaceWS/loginRequest"
message="tns:login" />
<output
wsam:Action="http://service.certifacews.com.br/CertiFaceWS/loginResponse"
message="tns:loginResponse" />
</operation>
<operation name="listDocuments">
<input
wsam:Action="http://service.certifacews.com.br/CertiFaceWS/listDocumentsRequest"
message="tns:listDocuments" />
<output
wsam:Action="http://service.certifacews.com.br/CertiFaceWS/listDocumentsResponse
" message="tns:listDocumentsResponse" />
</operation>
<operation name="listOtherDocuments">
<input
wsam:Action="http://service.certifacews.com.br/CertiFaceWS/listOtherDocumentsReq
uest" message="tns:listOtherDocuments" />
<output
wsam:Action="http://service.certifacews.com.br/CertiFaceWS/listOtherDocumentsRes
ponse" message="tns:listOtherDocumentsResponse" />
</operation>
<operation name="about">
```

```
<input wsam:Action="http://service.certifacews.com.br/CertiFaceWS/aboutRequest"
message="tns:about" />
<output
wsam:Action="http://service.certifacews.com.br/CertiFaceWS/aboutResponse"
message="tns:aboutResponse" />
</operation>
<operation name="certiface">
<input
wsam:Action="http://service.certifacews.com.br/CertiFaceWS/certifaceRequest"
message="tns:certiface" />
<output
wsam:Action="http://service.certifacews.com.br/CertiFaceWS/certifaceResponse"
message="tns:certifaceResponse" />
</operation>
<operation name="assessment">
<input
wsam:Action="http://service.certifacews.com.br/CertiFaceWS/assessmentRequest"
message="tns:assessment" />
<output
wsam:Action="http://service.certifacews.com.br/CertiFaceWS/assessmentResponse"
message="tns:assessmentResponse" />
</operation>
<operation name="findId">
<input wsam:Action="http://service.certifacews.com.br/CertiFaceWS/findIdRequest"
message="tns:findId" />
<output
wsam:Action="http://service.certifacews.com.br/CertiFaceWS/findIdResponse"
message="tns:findIdResponse" />
</operation>
<operation name="addDocuments">
<input
wsam:Action="http://service.certifacews.com.br/CertiFaceWS/addDocumentsRequest"
message="tns:addDocuments" />
<output
wsam:Action="http://service.certifacews.com.br/CertiFaceWS/addDocumentsResponse"
message="tns:addDocumentsResponse" />
</operation>
<operation name="updateDocuments">
<input
wsam:Action="http://service.certifacews.com.br/CertiFaceWS/updateDocumentsReques
t" message="tns:updateDocuments" />
<output
wsam:Action="http://service.certifacews.com.br/CertiFaceWS/updateDocumentsRespon
se" message="tns:updateDocumentsResponse" />
</operation>
<operation name="addDocumentsBySFTP">
<input
wsam:Action="http://service.certifacews.com.br/CertiFaceWS/addDocumentsBySFTPReq
uest" message="tns:addDocumentsBySFTP" />
<output
wsam:Action="http://service.certifacews.com.br/CertiFaceWS/addDocumentsBySFTPRes
ponse" message="tns:addDocumentsBySFTPResponse" />
</operation>
<operation name="addDocumentsByObjectArray">
<input
```

```
wsam:Action="http://service.certifacews.com.br/CertiFaceWS/addDocumentsByObjectA
rrayRequest" message="tns:addDocumentsByObjectArray" />
<output
wsam:Action="http://service.certifacews.com.br/CertiFaceWS/addDocumentsByObjectA
rrayResponse" message="tns:addDocumentsByObjectArrayResponse" />
</operation>
<operation name="addOtherDocumentsByObjectArray">
<input
wsam:Action="http://service.certifacews.com.br/CertiFaceWS/addOtherDocumentsByOb
jectArrayRequest" message="tns:addOtherDocumentsByObjectArray" />
<output
wsam:Action="http://service.certifacews.com.br/CertiFaceWS/addOtherDocumentsByOb
jectArrayResponse" message="tns:addOtherDocumentsByObjectArrayResponse" />
</operation>
<operation name="updateDocumentsBySFTP">
<input
wsam:Action="http://service.certifacews.com.br/CertiFaceWS/updateDocumentsBySFTP
Request" message="tns:updateDocumentsBySFTP" />
<output
wsam:Action="http://service.certifacews.com.br/CertiFaceWS/updateDocumentsBySFTP
Response" message="tns:updateDocumentsBySFTPResponse" />
</operation>
<operation name="DocumentsExist">
<input
wsam:Action="http://service.certifacews.com.br/CertiFaceWS/DocumentsExistRequest
" message="tns:DocumentsExist" />
<output
wsam:Action="http://service.certifacews.com.br/CertiFaceWS/DocumentsExistRespons
e" message="tns:DocumentsExistResponse" />
</operation>
</portType>
<binding name="CertiFaceWSPortBinding" type="tns:CertiFaceWS">
<soap:binding transport="http://schemas.xmlsoap.org/soap/http"
style="document" />
<operation name="login">
<soap:operation soapAction="" />
<input>
<soap:body use="literal" />
</input>
<output>
<soap:body use="literal" />
</output>
</operation>
<operation name="listDocuments">
<soap:operation soapAction="" />
<input>
<soap:body use="literal" />
</input>
<output>
<soap:body use="literal" />
</output>
</operation>
<operation name="listOtherDocuments">
<soap:operation soapAction="" />
<input>
<soap:body use="literal" />
</input>
```

```
<output>
<soap:body use="literal" />
</output>
</operation>
<operation name="about">
<soap:operation soapAction="" />
<input>
<soap:body use="literal" />
</input>
<output>
<soap:body use="literal" />
</output>
</operation>
<operation name="certiface">
<soap:operation soapAction="" />
<input>
<soap:body use="literal" />
</input>
<output>
<soap:body use="literal" />
</output>
</operation>
<operation name="assessment">
<soap:operation soapAction="" />
<input>
<soap:body use="literal" />
</input>
<output>
<soap:body use="literal" />
</output>
</operation>
<operation name="findId">
<soap:operation soapAction="" />
<input>
<soap:body use="literal" />
</input>
<output>
<soap:body use="literal" />
</output>
</operation>
<operation name="addDocuments">
<soap:operation soapAction="" />
<input>
<soap:body use="literal" />
</input>
<output>
<soap:body use="literal" />
</output>
</operation>
<operation name="updateDocuments">
<soap:operation soapAction="" />
<input>
<soap:body use="literal" />
</input>
<output>
<soap:body use="literal" />
</output>
```

```
</operation>
<operation name="addDocumentsBySFTP">
<soap:operation soapAction="" />
<input>
<soap:body use="literal" />
</input>
<output>
<soap:body use="literal" />
</output>
</operation>
<operation name="addDocumentsByObjectArray">
<soap:operation soapAction="" />
<input>
<soap:body use="literal" />
</input>
<output>
<soap:body use="literal" />
</output>
</operation>
<operation name="addOtherDocumentsByObjectArray">
<soap:operation soapAction="" />
<input>
<soap:body use="literal" />
</input>
<output>
<soap:body use="literal" />
</output>
</operation>
<operation name="updateDocumentsBySFTP">
<soap:operation soapAction="" />
<input>
<soap:body use="literal" />
</input>
<output>
<soap:body use="literal" />
</output>
</operation>
<operation name="DocumentsExist">
<soap:operation soapAction="" />
<input>
<soap:body use="literal" />
</input>
<output>
<soap:body use="literal" />
</output>
</operation>
</binding>
<service name="CertiFaceWSService">
<port name="CertiFaceWSPort" binding="tns:CertiFaceWSPortBinding">
<soap:address location="http://10.0.1.103:8080/CertiFaceWS/CertiFaceWS" />
</port>
</service>
</definitions>
```

Integração com o serviço Certiface REST API

Resumidamente, REST é a abstração da arquitetura da World Wide Web (Web), o REST tem como objetivo simplificar, isto acontece em função da não relevância dos detalhes da implementação de componente e a sintaxe de protocolo com o objetivo de focar nos papéis dos componentes, nas restrições sobre sua interação com outros componentes e na sua interpretação de elementos de dados significantes.

Baseado na Wikipédia, o termo transferência de estado representacional foi introduzido e definido no ano de 2000 por Roy Fielding, um dos principais autores da especificação do protocolo HTTP que é utilizado por sites da Internet, em uma tese de doutorado (PHD) na UC Irvine. REST tem sido aplicada para descrever a arquitetura web desejada, identificar problemas existentes, comparar soluções alternativas e garantir que extensões de protocolo não violem as principais restrições que fazem da Web um sucesso. Fielding desenvolveu a REST em colaboração com seus colegas enquanto trabalhava no HTTP 1.1 e nos Identificadores de Recursos Uniformes. O estilo arquitetural de REST também é aplicado no desenvolvimento de serviços Web. Pode-se caracterizar os web services como "RESTful" se eles estiverem em conformidade com as restrições descritas na seção restrição arquiteturais.

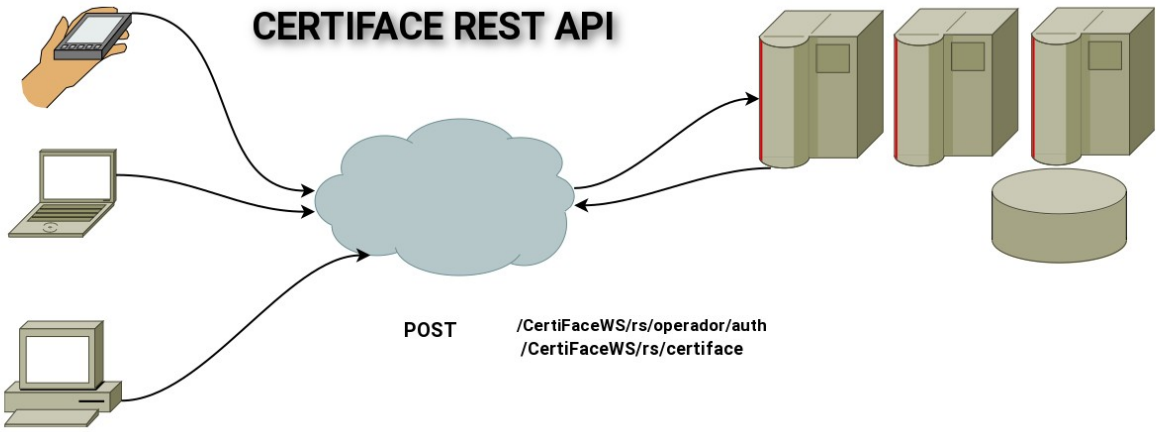
O termo REST se referia, originalmente, a um conjunto de princípios de arquitetura (descritos mais abaixo), na atualidade se usa no sentido mais amplo para descrever qualquer interface web simples que utiliza XML (ou YAML, JSON, ou texto puro) e HTTP, sem as abstrações adicionais dos protocolos baseados em padrões de trocas de mensagem como o protocolo de serviços Web SOAP. É possível desenhar sistemas de serviços Web de acordo com o estilo arquitetural REST descrito por Fielding, e também é possível desenhar interfaces XMLHTTP de acordo com o estilo de RPC mas sem utilizar SOAP. Estes usos diferentes do termo REST causam certa confusão em discussões técnicas, onde RPC não é um exemplo de REST.

Consumo do WebService com REST

O consumo do WebService Certiface REST API, acontece com praticamente com duas chamadas. Primeiramente devemos solicitar a API de autenticação para obter o token acompanhado da data e hora de expiração.

Posteriormente utilizamos o token na API de cadastro precedido dos

dados cadastrais (nome, data de nascimento e CPF).



Guia de utilização do Certiface Rest service API

A API Certiface é uma aplicação REST. Toda a comunicação utiliza no resposta os dados no formato JSON, tanto para request quanto para response.

É necessário informar no cabeçalho as requisições, sem exceção. A API tem recursos orientados pela URL e utiliza os Status Codes para indicar os erros. Como mencionado anteriormente o Certiface trabalha comente com 2 URL para prover o serviço de biometria facial.

A API REST do Certiface pode ser consumida por qualquer plataforma desktop (Windows, Linux e Mac) ou mobile (Android, iOS, FirefoxOS e outros). Caso as informações deste documento não seja suficiente para integração no ambiente legado, entre imediatamente em contato com o departamento de desenvolvimento e suporte da OITI TECNOLOGIA. A seguir as informações referentes as chamadas dos Certiface.

Durante a integração a OITI disponibiliza login de testes para utilizar a ferramenta com testes operacionais, como também para testes de integração e chamadas da API. Sendo assim a URL de homologação teste e desenvolvimento está disponível em: https://comercial.certiface.com.br. Já a URL oficial de consumo do certiface encontra-se em

https://www.certiface.com.br .

Método: AUTENTICAÇÃO

Na URL de autenticação, o principal objetivo é autenticar o usuário do sistema, como também obter o token de acesso a chamada biométrica do sistema. A resposta dos dados no formato JSON é composto do token e data/hora de expiração.

Requisição do Método para Autenticação:

Requisição: POST /CertiFaceWS/rs/operador/auth HTTP/1.1
Parâmetros: login e senha.
Resposta: Token e data/hora de expiração, para acessar o método de cadastro e certificação biométrica.

Exemplo da chamada do método POST:

```
POST /CertiFaceWS/rs/operador/auth HTTP/1.1
Host: certiface.com.br
Cache-Control: no-cache
Content-Type: application/x-www-form-urlencoded
login=usuario&senha=senha_em_md5
```

Após executado o método POST o retorno é o token que deve ser utilizado para acessar o método de cadastro e certificação biométrica. O token é sua chave privada de acesso é não deverá ser reenviada em nenhum momento. Abaixo um exemplo referente ao retorno do token:

```
{ "token" : "cKaEzJ9G4F-AX5XE0fKesVsuI0sZZkwC9Hm7z3n5PPg" ,  "expires" :
                          "11/06/2015 18:01:22" }
```

*** O token possui tempo de expiração seguindo o UTC (Coordinated Universal Time).

Método: CADASTRO E CERTIFICAÇÃO

Na URL de Cadastro, podemos efetuar o cadastro do usuário enviando a foto em Base64 ou ByteArray acompanhado dos dados cadastrais (nome, cpf e data

de nascimento). Cada empresa e/ou cadastro pode operar com retorno diferentes em função de sua necessidade operacional. Sou seja, retorno apenas referente a conclusão do cadastro (Retorno Simples), como também o retorno baseado no processamento de um provável fraude (Retorno Detalhado).

Requisição do Método de Cadastro

Requisição: POST /CertiFaceWS/rs/certiface HTTP/1.1

Authorization: usuário;**assinatura**;data e hora de expiração
Cache-Control: no-cache
Content-Type: multipart/form-data

Parâmetros:

CPF Sem pontuação
Nome No mínimo com um sobrenome
Nascimento Data de nascimento no formato (dd/mm/yyyy).
Imagem byte array (byte[]) ou base64 contendo a face para cadastramento
Resposta: Resposta Simples ou Detalhada referente ao cadastro efetuado

Atenção especial ao cabeçalho da requisição no Authorization utilizado na autenticação. Esta autenticação é composta do login do usuário, assinatura, data e hora de expiração retornado na requisição do método de login. Estes campos utilizam o carácter “;” como separador.

A assinatura/hash deve ser gerada utilizando o algoritmo SHA256 utilizando o método+pathURL+data_e_hora_de_expiração como texto base e o token como chave. Ex:

```
var textToSign = (verb + pathCertiface + expires);
var hash = CryptoJS.HmacSHA256(textToSign, token);
```

Exemplo da chamada do método POST:

```
POST /CertiFaceWS/rs/certiface HTTP/1.1
Host: certiface.com.br
```

```
Authorization: usuario;5S9wfiTrbvFQfCgSLy7tMN_EUHzAiQAE9EN-cMTvX9Y;11/06/2015
18:24:35
Date: 11/06/2015 17:55:13

Cache-Control: no-cache

Content-Type: multipart/form-data;
boundary=----WebKitFormBoundary7MA4YWxkTrZu0gW




----WebKitFormBoundary7MA4YWxkTrZu0gW
Content-Disposition: form-data; name="imagem"

/9j/4AAQSkZJRgABAQEBKwErAAD/2wBDAAMCAgMCAgMDAwMEAwMEBQgFBQQEBQoHBwYID……
----WebKitFormBoundary7MA4YWxkTrZu0gW
Content-Disposition: form-data; name="cpf"

00000000000
----WebKitFormBoundary7MA4YWxkTrZu0gW
Content-Disposition: form-data; name="nome"

NOME DO CLIENTE
----WebKitFormBoundary7MA4YWxkTrZu0gW
Content-Disposition: form-data; name="nascimento"

06/06/1987
----WebKitFormBoundary7MA4YWxkTrZu0gW--
```

Após a chamada do método, a resposta é enviada no formato JSON conforme previamente configurada no contrato do serviço. Este retornos atende as mais diversas necessidades mercadológicas.

Retorno Simples

O Retorno Simples apresenta apenas uma resposta de ERRO somente se algum requisito da imagem enviada, não atenda especificações mínimas do cadastro biométrico (Face não encontrada, olhos fechados e face não frontal). Estas mensagens são configuradas e parametrizadas por empresa. As mensagens podem serem configuras para atender um determinado público-alvo.

O Retorno Simples geralmente é utilizado em empresas onde as informações referentes a fraude (face com vários similares na base) não deve ser exibida no

cadastro presencial ou cadastro não assistido onde o usuário final ou atendente de loja não tem a necessidade de obter informação referente sobre a(s) face(s) encontrada(s) na base (sendo uma vez que esta informação é geralmente enviada para mesa de aprovação). Na imagem a seguir um exemplo de operação com o Retorno Simples configurado.

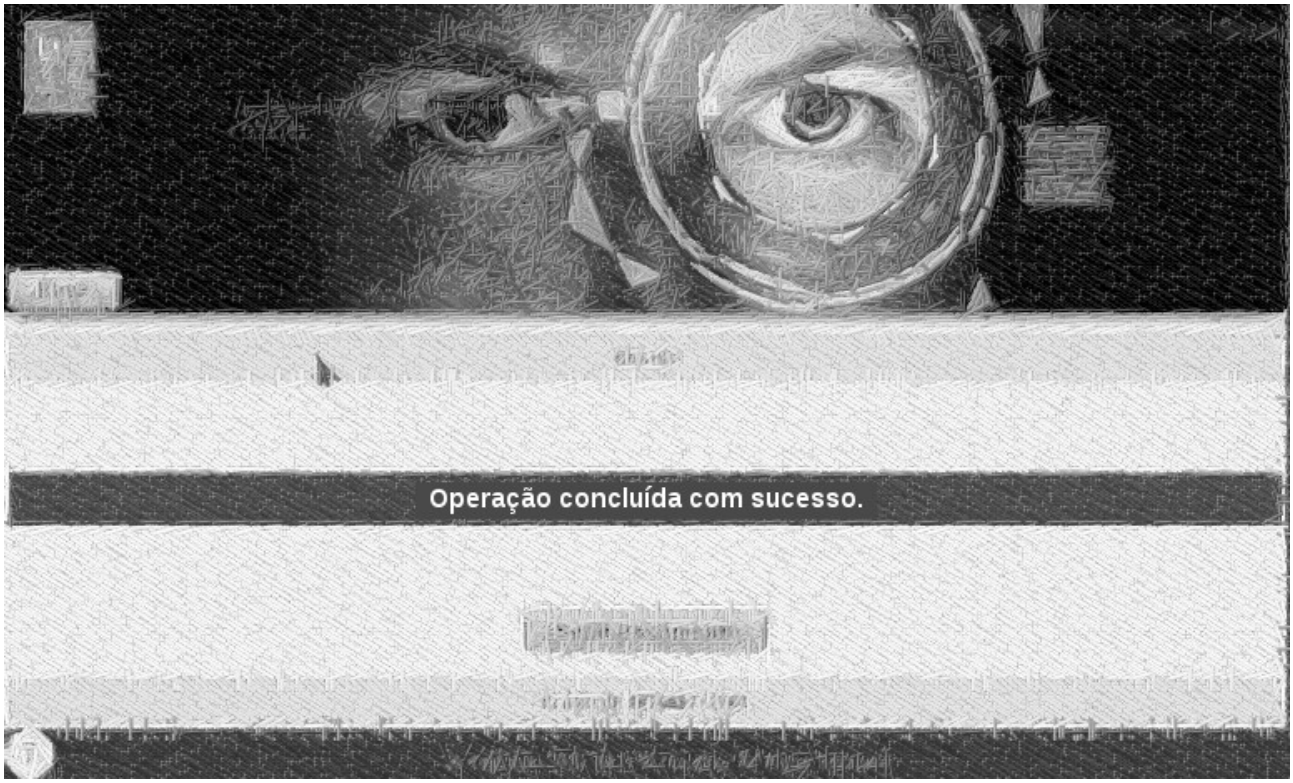


figura 10 : Cadastro com Retorno Simples.

Após executado o método POST o retorno como mencionado anteriormente apresentam dois modos. Abaixo o exemplo do primeiro modo denominado **Retorno Simples,** os detalhamentos dos campos está disponível na página seguinte.

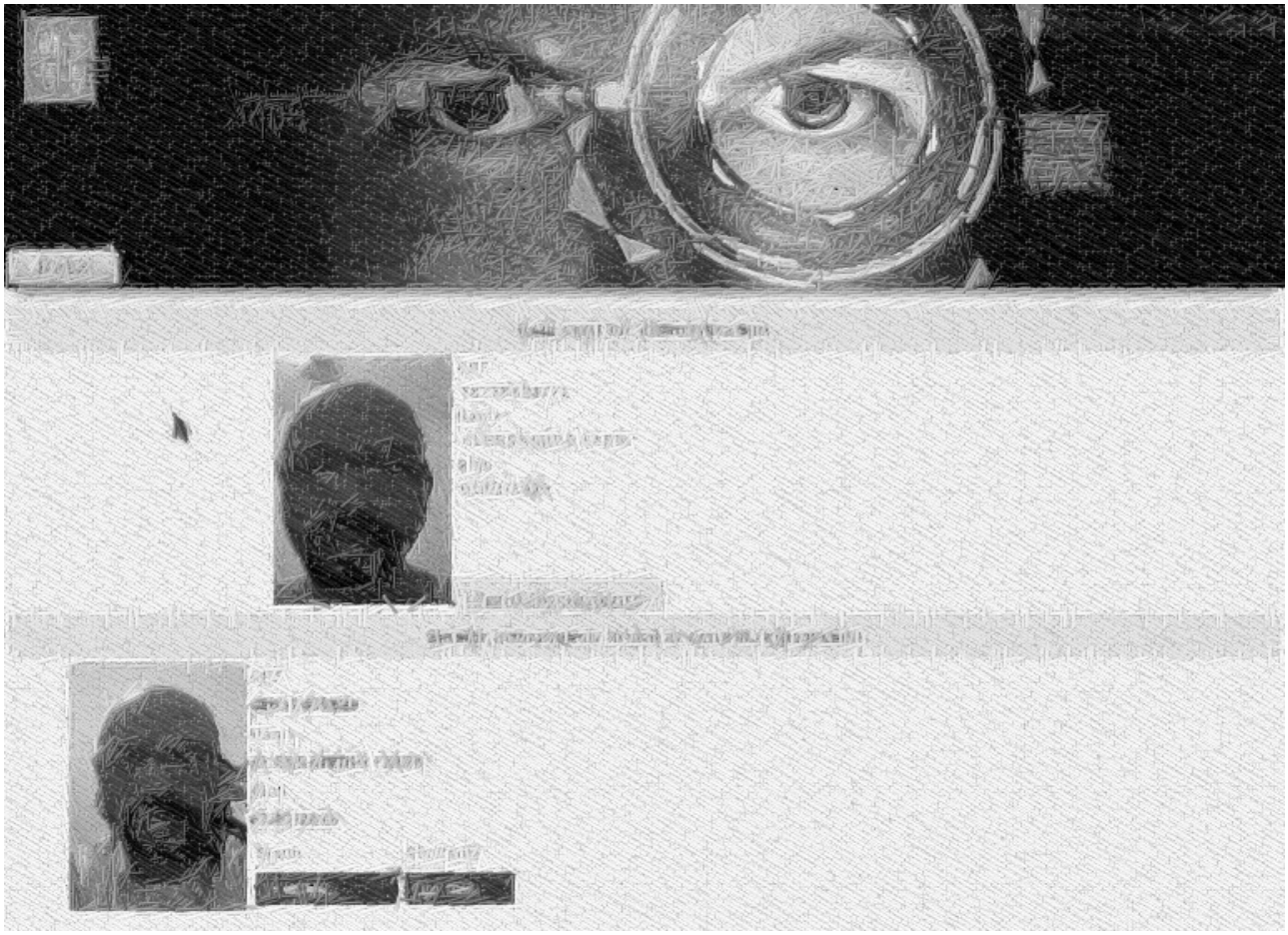
```
{"protocolo":"201600771280","status":"IDE","resultado":"SUC","similares":[]}
```

*** O resultado simples não retorna as fases similares (quando existir)

Retorno Detalhado

O Retorno Detalhado ao contrário do retorno simples apresenta como resultado da operação as faces similares acompanhado do nome, cpf e data de nascimento. Geralmente útil quando o processo operacional decide diante o cliente aprovação ou reprovação cadastral.

Como no Retorno Simples, o Retorno detalhado retorna uma mensagem de erro quando a imagem enviada apresenta algum erro ou não apresenta propriedades suficientes para processar o processo de cadastramento biométrico. Na imagem a seguir podemos compreender a real diferenças entre o Retorno Simples e Detalhado.



Após executado o método POST o retorno como mencionado anteriormente apresentam dois modos. Abaixo o exemplo do modo denominado **Retorno Detalhado:**


```
{
     "protocolo": "201500706654",
     "status": "IDE",
     "resultado": "PEN",
     "similares": [{
           "cpf": "21270705261",
```

```
            "nome": "TESTE CERTIFACE 21270705261",
            "nascimento": "06/06/1987",
            "status": "P",
            "score": "1.0",
            "foto": "/9j/4AAQSkZJRgABAwMEAwMEBQgFBQQEBQoHBwYID……",
            "tipo": "B",
            "cadastro": "2015-08-12 09:31:21.0"
      },
      {
            "cpf": "65143506824",
            "nome": "TESTE CERTIFACE 65143506824",
            "nascimento": "06/06/1987",
            "status": "P",
            "score": "1.0",
            "foto": "/9j/4AAQSkZJRgABAwMEAwMEBQgFBQQEBQoHBwYID……",
            "tipo": "B",
            "cadastro": "2015-08-12 09:35:21.0"
      }]
}
```


*** O resultado detalhado retorna as fases similares (quando existir)

Especificação dos campos de retorno:

Vale a pena ressaltar que todo retorno pode ser customizado mediante a necessidade do cliente. Mas segue o retorno simples padrão:

PROTOCOLO*:* Identificador único do processo.

STATUS*:* Identificador do serviço realizado. Abaixo os possíveis valores.

> **IDE*:*** Novo cadastro. O CPF não foi encontrado no Certiface.
> **VAL*:*** CPF já existente, a imagem foi comparada com a imagem preexistent
> **ERR*:*** Erro no processo.

RESULTADO: Resultado do serviço/status do serviço realizado. Abaixo os possíveis valores.

SUC: No caso de STATUS *IDE*(novo cadastro), o Certiface não encontrou nenhum ou similar(es) com baixa probabilidade de fraude.

PEN: No caso de STATUS *IDE*(novo cadastro), o Certiface encontrou similar(es) com alta probabilidade de fraude.

POS: No caso de STATUS *VAL*(CPF já existente), o Certiface compara a face enviada com a imagem existente na base de dados, e resultado da similaridade apresentou resultado positivo.

NEG: No caso de STATUS *VAL*(CPF já existente), o Certiface compara a face enviada com a imagem existente na base de dados, e o resultado da simlaridade apresentou um resultado negativo.

FAS: No caso de STATUS *ERR*(Erro no processo cadastral), a imagem enviada não atende os atributos mínimos de qualidade exigida pelo Certiface.

SIMILARES: Quando configurado no modo Resultado Detalhado, os dados dos similiares

serão retornados quando existir. A seguir os campos dos registros similares

CPF: Documento do usuário;
NOME: Nome do usuário;
NASCIMENTO: Data de nascimento do usuário;
STATUS: **(A)**tivo, **(P)**endente ou **(R)**estrito;
SCORE: Similaridade Biométrica;
TIPO: Tipo da origem da imagem;
CADASTRO: Data e Hora do cadastramento.

MSG: No caso de *ERR*(Erro no processo), este campo é adicionado com o descritivo do erro, no caso citado acima, onde o erro foi causado por falta de qualidade na imagem. A mensagem apresentada retorna texto como *"OLHOS NÃO ENCONTRADOS", "FACE NÃO ENCONTRADA", "NÃO FRONTAL"* etc. Outras mensagens de erro também são adicionadas neste campo.

Exemplos

A seguir alguns trechos de exemplo nas linguagens mais populares, a OITI trabalhar com a entregar de exemplo sob demanda em virtude dos inúmeros tipos de linguagens e/ou tecnologia disponíveis. Mas como o uso de consumo de API via webservice é uma prática do dia a dia de qualquer serviço em nuvem, acreditamos não existem impacto em qualquer linguagem de programação.

Exemplo em JavaScript

A seguir uma sugestão/exemplo de uma função para obter token/assinatura para utilizarmos no método de autenticação. Reparem que esta função recebe alguns parâmetros como o login, senha, URL, path da autenticação

```
function getSignature(login, senha, url, pathAuth , pathCertiface, verb) {

    var tokenDate = getToken(login, senha, url, pathAuth);
    if (tokenDate) {
        console.log("TOKEN: " + tokenDate.token + "," + tokenDate.expires);

        var token = tokenDate.token;
        var expires = tokenDate.expires;
        var textToSign = (verb + pathCertiface + expires);

        var hash = CryptoJS.HmacSHA256(textToSign, token);
        var hashInBase64 = CryptoJS.enc.Base64.stringify(hash);
        var hashInBase64Safe = hashInBase64.replace(/\+/g, '-').replace(/\//g,
'_').replace(/\=+$/, '');
```

```
        var signature = login + ';' + hashInBase64Safe + ';' + expires;

        return {token: signature, date: moment(new
Date().toISOString()).format('DD/MM/YYYY hh:mm:ss')};
    }

}
```

Enviado os dados do formulário da página para o Certiface. Ressaltamos que os códigos disponíveis têm como objetivo demonstras a simplicidade de integração. Fica a cargo de cada desenvolvedor criar o seu processo de envio.

```
function setRequest() {

    /* Carregando o dados inseridos */
    var login = document.getElementById('login').value;
    var senha = document.getElementById('senha').value;
    if (login == "" || senha == ""){
        alert("Insira o login e senha.");
        return;
    }
    var senhaHash = md5(senha);


    var nome = document.getElementById('nome').value;
    var cpf = document.getElementById('cpf').value;
    var dataDeNascimento = "01/01/2001";

    var imagemSrc = document.getElementById('imagemPrev');

    /* Formatando o base64 */
    data = imagemSrc.src.replace(/^data:image\/(png|jpeg);base64,/, "");

    /* Gerando assinatura */
    setResposta("Enviando token request, aguarde....");
    var signature =
getSignature(login,senhaHash,url,pathAuth,pathCertiface,verb);

    if (signature) {
        /* Enviando para o certiface */
        setResposta("Enviando certiface request, aguarde....");
        submitForm(cpf, nome, dataDeNascimento, data, url, pathCertiface,
signature);
    }

}
```

Exemplo em PHP

A seguir uma sugestão/exemplo de uma função para obter token/assinatura para utilizarmos no método de autenticação. Reparem que esta função tem o objetivo de mostra a facilidade do consumo do WebService utiliza o pacote CURL, existem diversas técnicas o que tornaria inviável mencioná-las todas neste documento.

```
function login() {
  $curl = curl_init();

  curl_setopt_array($curl, array(
    CURLOPT_PORT => "8443",
    CURLOPT_URL =>
"https://comercial.certiface.com.br:8443/CertiFaceWS/rs/operador/auth",
    CURLOPT_RETURNTRANSFER => true,
    CURLOPT_SSL_VERIFYPEER => false,
    CURLOPT_USERAGENT => "Php request",
    CURLOPT_ENCODING => "",
    CURLOPT_MAXREDIRS => 10,
    CURLOPT_TIMEOUT => 30,
    CURLOPT_HTTP_VERSION => CURL_HTTP_VERSION_1_1,
    CURLOPT_CUSTOMREQUEST => "POST",
    CURLOPT_POSTFIELDS => "login=usuario&senha=senha-em",
    CURLOPT_HTTPHEADER => array(
      "cache-control: no-cache",
      "content-type: application/x-www-form-urlencoded"
    ),
  ));

  $response = curl_exec($curl);
  $err = curl_error($curl);

  curl_close($curl);

}
```

Exemplo em C# dotNET

A seguir uma sugestão/exemplo de uma função para obter token/assinatura para utilizarmos no método de autenticação. O código é apenas uma prova de conceito para demonstrar a facilidade de integração com o serviço Certiface.

Em breve disponibilizaremos a api de acesso em dotNet core, a iniciativa openSouce da Microsoft para utilização desta linguagem e plataforma de desenvolvimento na plataforma GNU/Linux.

```
public Token GetToken(string usuario, string senha)
```

```
{
      // cria o Objeto Token de Retorno
      Token token = new Token();

      // metodo de autenticação
      string metodo = "CertiFaceWS/rs/operador/auth";
      // Cria o MD5 da senha
      string pass = Crypto.CreateMD5Hash(senha);

      // cria o endereço de URL de conexão
      string adress = BASE_URI + metodo;
      // Cria o Handler para descomprimir automaticamente
      HttpClientHandler handler = new HttpClientHandler();
      handler.AutomaticDecompression = System.Net.DecompressionMethods.GZip |
System.Net.DecompressionMethods.Deflate;

      // Cria o Cliente HTTPClient com o Handler
      HttpClient httpClient = new HttpClient(handler);
      KeyValuePair<string, string> loginPair = new KeyValuePair<string,
string>("login", usuario);
      KeyValuePair<string, string> senhaPair = new KeyValuePair<string,
string>("senha", pass);

      // Cria paremetros do tipo: application/x-www-form-urlencoded
      FormUrlEncodedContent form = new FormUrlEncodedContent(new[] { loginPair,
senhaPair });

      // Envia o post e aguarda o resultado
      var result = httpClient.PostAsync(adress, form).Result;
      Console.WriteLine(": " + result.StatusCode);

      // verifica o StatusCode = OK (200)
      if (result.StatusCode == System.Net.HttpStatusCode.OK)
      {
         // Cria o Json de Resposta.
         string contentJson = result.Content.ReadAsStringAsync().Result;
         dynamic json = JsonConvert.DeserializeObject(contentJson);
         token.TokenID = json.token;
         token.Expires = json.expires;
      }
      else
      {
         return null;
      }

     return token;
  }
```

Liveness / Prova de Vida API

O serviço de prova de vida, tem como principal objetivo processar a autenticidade do ser humano. Assim evitando fraudes convencionais para “burlar” sistemas de biometria facial como utilização de máscaras, e vídeos diante a webcam. Deferentemente das soluções encontradas no mercado, o serviço é processado 100% em nuvem.

Com todo o processamento matemático em nuvem, é possível processar a prova de vida em celulares de baixo custo, bastando apenas uma conexão com a internet suficientes para enviar alguns quadros do vídeo ao vivo exibido no dispositivo de captura. Resumidamente a prova de vida pode ser processada em qualquer dispositivo com câmera conectado na internet.

A operação da prova de vida consiste no processamento dos desafios solicitados após o pressionamento do botão iniciar. Automaticamente os quadros são processados, e os desafios são lançados de acordo com a amostragem disponível no vídeo ao vivo. Por exemplo, se a imagem apresentar insuficiência nos cálculos da geometria dos lábios (ou seja baixa confiabilidade dos pontos estruturais da boca), não será enviado um desafio de sorriso ao usuário.

Os desafios são processados por diversos algoritmos de visão computacional, a seguir os principais desafios do componente prova de vida.

- Olhar para cima;
- Olhar para baixo;
- Olhar para esquerda
- Olhar para direita;
- Sorrir;
- Piscar os olhos;
- Erguer as sobrancelhas.
- Inclinar o rosto para esquerda.
- Inclinar o rosto para direita.

Integração do componente Liveness/Prova de Vida em JA

A API de integração da prova de vida, **consiste apenas em obter a chave de aplicativo e inseri-la na chamada em javascript.** A Seguir o modelo do arquivo HTML e JS:

```
<html>
	<head>
	</head>
	<body>
	</body>
	<script src="jquery.min.1.5.2.js"></script>
	<script src="faceCaptchaApp.js"></script>
</html>
```

*** exemplo HTML

```
var CPF = "0000004141";
var nome = "Usuario teste";
var dtNasc = "10/10/1999";
$().ready(function(){
	getAndStartFaceCaptcha();
});

function getExtras(){
		var info =
		  "cpf," + CPF + ",nome," + nome + ",nascimento," + dtNasc;
    return info;
}
function getAndStartFaceCaptcha(){
	var captchaObj = $('<div style="text-align: center;padding-top:
25px;padding-left: 22px;"><div id="faceCaptcha" accesskey="oKIZ1jjpRyXCDDNiR--
_OPNGiNmraDZIuGE1rlUyZwOGJJzDtJR7BahJ4MqnobwetlmjXFsYFeze0eBRAGS2KWmjFUp08HYsv6p
yI3KZklISJVmKJDSgfmkRmPaBR9ZJP3wtVWFDwNR9kS_vecameg"
onFinish="onFinishFaceCaptcha" autorestart="false"></div></div>');
```

```
      $('body').append(captchaObj);
$.getScript('https://comercial.certiface.com.br:8443/facecaptcha/service/
captcha/getit' + '?nc=' + new Date().getTime(), function(script, textStatus,
jqXHR){
            initFaceCaptcha();
      });
}
function onFinishFaceCaptcha(response){

      var msg = "Falha na valida\u00e7\u00e3o";
      if(response.valid){
            msg = "Validado com sucesso";
      }
      faceCaptchaShowCustomMsg(msg);
      setTimeout(function () {
            initFaceCaptcha();
      }, 7000);
}
```

*** exemplo Javascript

Na função onFinishFaceCaptcha obtemos a resposta referente ao processamento biométrico. Com esta resposta podemos obter o resultado para então decidir o fluxo da aplicação. A seguir os possíveis resultados obtido via java script :

```
response.valid [true / false]
```

Descrição: Retorna o resultado do processamento do algoritmo prova de vida do Certiface.

`True` **=** Significa que todos os desafios foram concluídos com sucesso.

`False` **=** Indique que um ou mais desafios não foram processados com sucesso.

```
response.cause “String”
```

Descrição: Retorna uma string representando o motivo do processamento, quando a prova de vida for concluída com sucesso o resultado é (null).

`“PROVA DE VIDA”` **=** Indica que o motivo da falha foi devido ao processamento dos desafios da prova de vida.

`“BIOMETRIA”` **=** Informa que o motivo da falha foi devido ao processamento biométrico.

`“IMAGEM”` **=** Informa que o motivo da falha foi devido a

qualidade insuficiente da image para processamento biométrico (ex: face não encontrada, olhos não encontrado, iluminação insuficiente).

```
response.hash “String”
```

Descrição: Retorna o Hash utilizado para a confirmação do resultado biométrico via servidores via Webservice em REST Full.

```
hash: "Q3MBn_inWozoCj5HJbC.........…"
```

```
response.protocol “String”
```

Descrição: Protocolo do log da prova de vida, geralmente utilizado para consultas posteriores.

```
Protocol:"2017***"
```

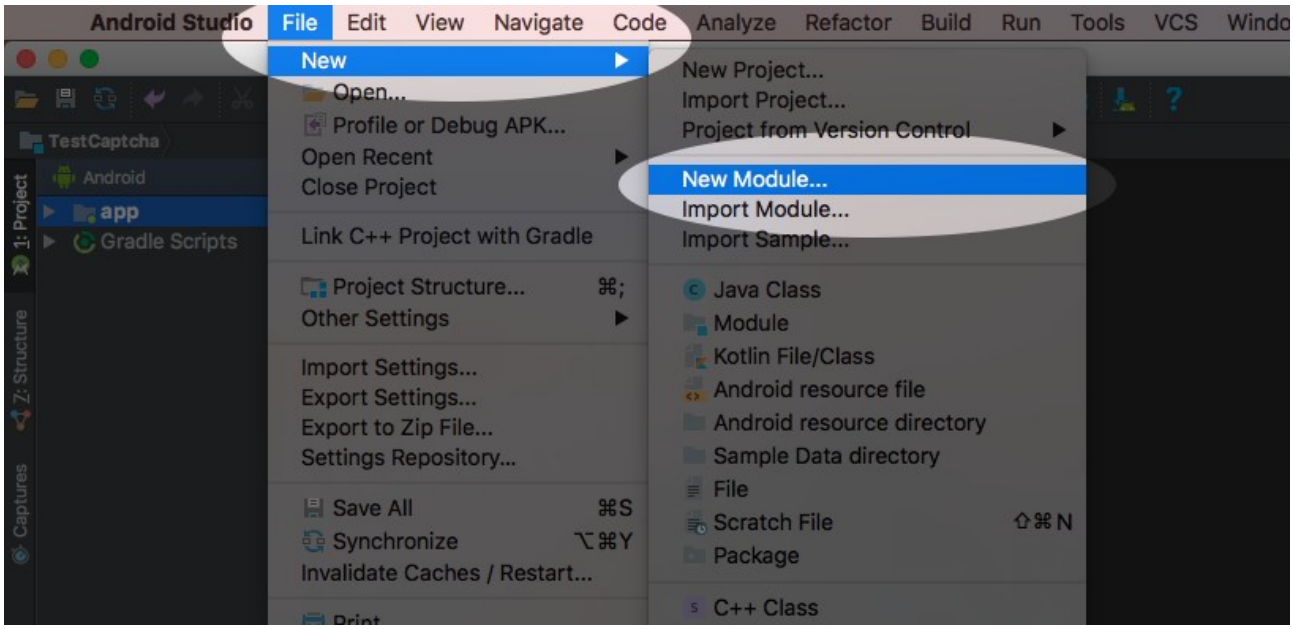
```
response.snap “String”
```

Descrição: Retorna a imagem JPG em base64, utilizada quando necessário armazenar a imagens cadastral localmente ou no sistema legado do cliente final.

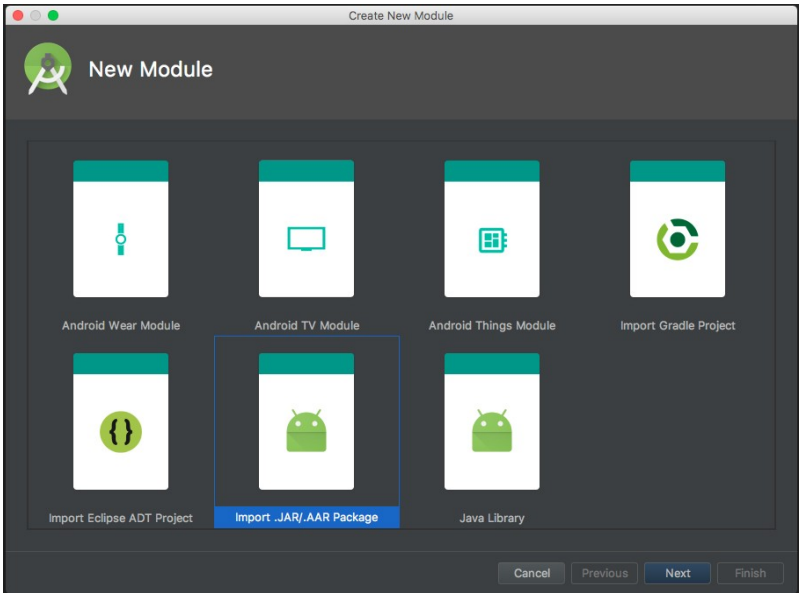
```
snap:".....…"
```

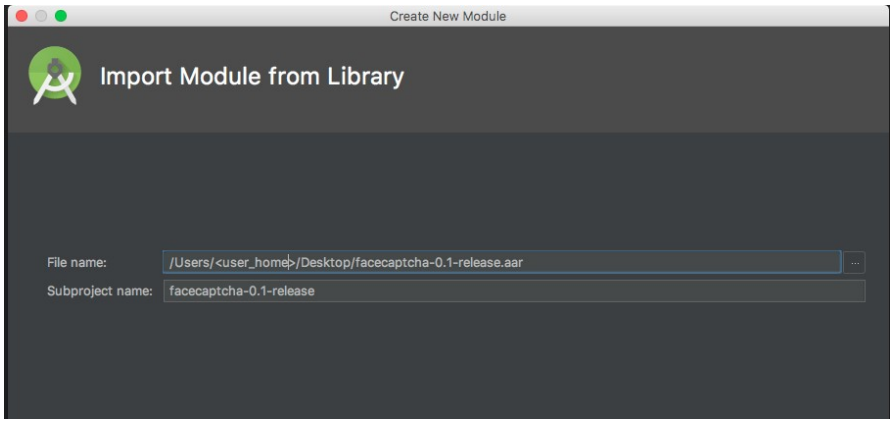
Integração do componente Liveness/Prova em Android

Com o componente facecaptcha-X.Y-release.aar salvo em disco, inicie a ferramente de desenvolvimento Android Studio, em seguida selecione no menu principal a opção FILE → NEW → New Module. Conforme a ilustração a seguir:

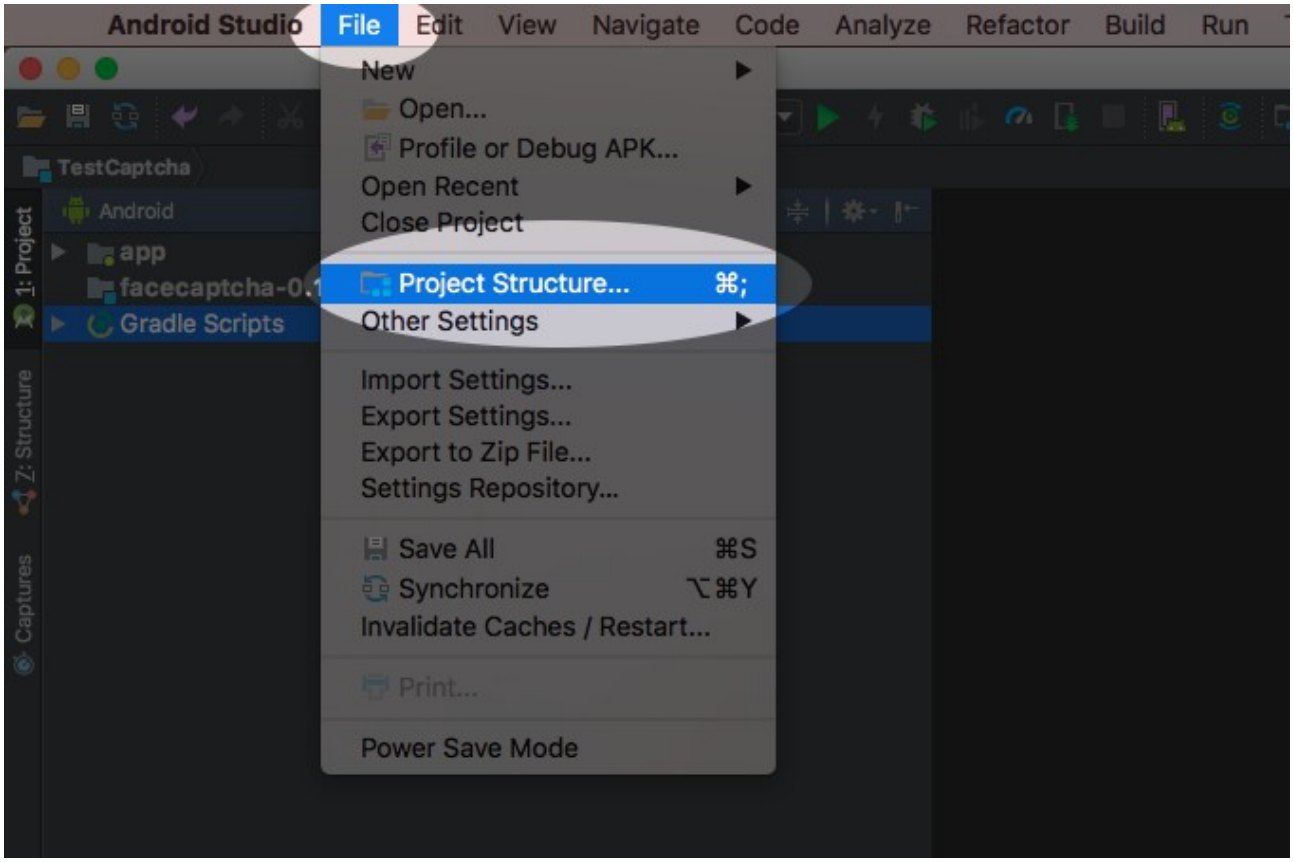


Agora na janela de dialogo “CREATE NEW MODULE”, selecione o item “import .JAR//AAR Package”. Após, informe o caminho do arquivo .AAR e clique em NEXT, e para finalizar clique em FINISH.

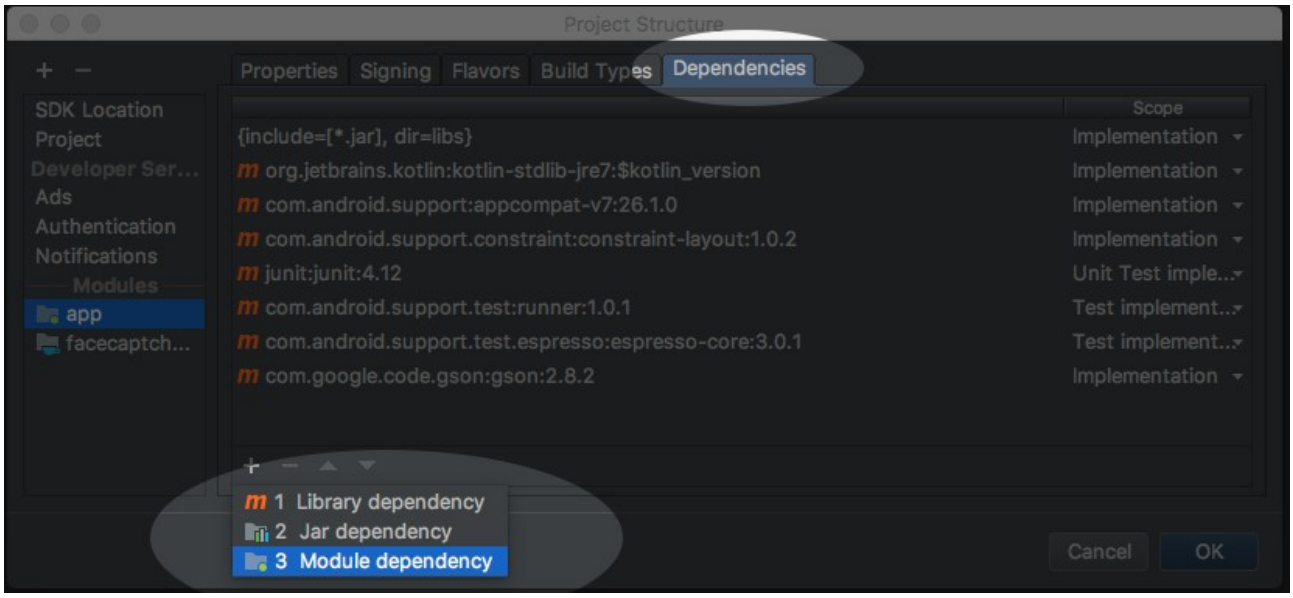




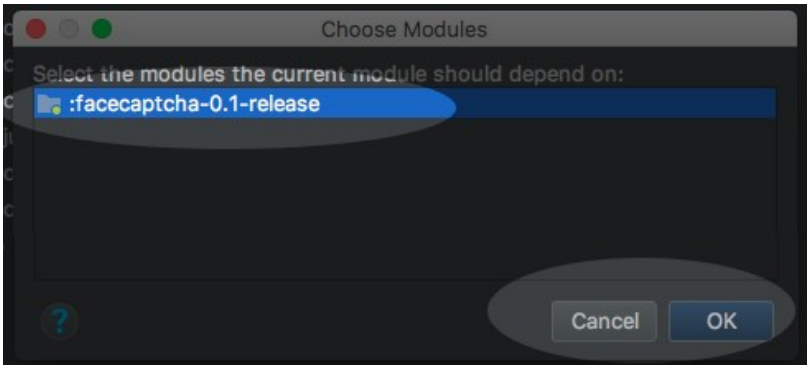
Para configurar a dependência do módulo, selecione no menu o item File → Project Structure conforme a imagem a seguir:



No formulário “Project Structure”, selecione a aba “Dependencies”. Na parte inferior da janela atual, clique no botão “+” e selecione “3 Module dependency” como no exemplo abaixo:



Selecione o módulo “facecaptcha” recém-criado, e clique em OK.



Agora com o ambiente configurado, ao criar um novo projeto, devemos incluir as dependências em build.gradle:

```
implementation "com.squareup.retrofit2:retrofit:2.3.0"
implementation "com.squareup.retrofit2:converter-gson:2.3.0"
implementation "org.bouncycastle:bcprov-jdk16:1.45"
implementation "com.android.support:design:26.1.0"
implementation "org.jetbrains.kotlin:kotlin-stdlib-jre7:1.2.0"
```

A seguir um exemplo:

```
dependencies {
    implementation fileTree(include: ['*.jar'], dir: 'libs')
    implementation project(':facecaptcha-0.1-release')

    implementation 'com.android.support:appcompat-v7:26.1.0'
    implementation 'com.android.support.constraint:constraint-layout:1.0.2'

    implementation "org.jetbrains.kotlin:kotlin-stdlib-jre7:1.2.0"
    implementation 'com.android.support:design:26.1.0'
    implementation 'com.squareup.retrofit2:retrofit:2.3.0'
    implementation 'com.squareup.retrofit2:converter-gson:2.3.0'
    implementation 'org.bouncycastle:bcprov-jdk16:1.46'

    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'com.android.support.test:runner:1.0.1'
    androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.1'
}
```

Agora na Activity desejada, declare a importação do pacote FaceCaptchaActivity com o comando import:

```
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.widget.Toast;

import br.com.oiti.certiface.FaceCaptchaActivity;
```

O Face CaptchaActivity deve receber o seguintes parâmetros:

```
FaceCaptchaActivity.PARAM_ENDPOINT: Server endpoint
FaceCaptchaActivity.PARAM_APP_KEY: User application key (static ou dinamic)
FaceCaptchaActivity.PARAM_USER_INFO: User sensible data
FaceCaptchaActivity.PARAM_OVERLAY_IMAGE: [optional] PNG image drawable id
```

A seguir os possíveis retorno a Activity:

Activity.RESULT_CANCELED: Operação cancelada pelo usuário.

Activity.RESULT_OK: Operação concluída com sucesso. Neste caso, a activity retornará 2 parâmetros extras:

FaceCaptchaActivity.RESULT_HASH: Hash utilizado nos servidores backend para validação e/ou obter informações da validação efetuada.

FaceCaptchaActivity.RESULT_PROTOCOL: Número do protocolo gerado no servidor durante o processo de validação.

A seguir um exemplo de implementação.

```java
public class JavaActivity extends AppCompatActivity {

    private static final int CAPTCHA_RESULT_REQUEST = 1;
    private static final String ENDPOINT = "https://comercial.certiface.com.br:8443";
    private static final String APP_KEY = "oKIZ1jjpRyXCDDNiR--_OPNGiNmraDZIuGE1JVmKJDSgfmkRmPaBR9ZJP3wtVWFDwNR9kS_vecameg";
    private static final String USER_INFO = "user,comercial.token,cpf,123456789,nome,JOAO DA SILVA,nascimento,11/01/1900";

    protected void onCreate(@org.jetbrains.annotations.Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        this.setContentView(R.layout.activity_main);

        startChallenge( context: this);
    }

    private void startChallenge(Context context) {
        Intent intent = new Intent(context, FaceCaptchaActivity.class);
        intent.putExtra(FaceCaptchaActivity.PARAM_ENDPOINT, ENDPOINT);
        intent.putExtra(FaceCaptchaActivity.PARAM_APP_KEY, APP_KEY);
        intent.putExtra(FaceCaptchaActivity.PARAM_USER_INFO, USER_INFO);
        intent.putExtra(FaceCaptchaActivity.PARAM_OVERLAY_IMAGE, R.drawable.overlay);

        this.startActivityForResult(intent, CAPTCHA_RESULT_REQUEST);
    }

    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if(requestCode == CAPTCHA_RESULT_REQUEST) {
            switch(resultCode) {
                case RESULT_OK:
                    this.resultSuccess(data);
                    break;
                case RESULT_CANCELED:
                    this.resultCanceled();
            }
        }
    }

    private void resultSuccess(Intent data) {
        Boolean result = data != null ? data.getBooleanExtra(FaceCaptchaActivity.PARAM_RESULT,  defaultValue: false) : null;
        String cause = data != null ? data.getStringExtra(FaceCaptchaActivity.PARAM_RESULT_CAUSE) : null;

        Toast.makeText( context: this,  text: "Valid: " + result + "\nCause: " + cause, Toast.LENGTH_LONG).show();
    }

    private void resultCanceled() {
        Toast.makeText( context: this,  text: "Ação cancelada", Toast.LENGTH_SHORT).show();
    }
}
```

Integração do componente Liveness/Prova em Cordova

Crie um novo projeto com a tecnologia Cordova e importe os arquivos a

seguir javascrip e css:

```
<script type="text/javascript" src="cordova.js"></script>
<script src="js/aes.js"></script>
<script src="js/jquery.min.js"></script>
<script src="js/cripto.js" charset="UTF-8"></script>
<script src="js/facecaptcha.js" charset="UTF-8"></script>
<script type="text/javascript" src="js/index.js"></script>
</body>
```

```
<meta name="viewport" content="user-scalable=no, initial-scale=1, maxim
<link rel="stylesheet" type="text/css" href="css/facecaptcha.css">
<title>Hello World</title>
```

O requisito para o desenvolvimento do FaceCaptcha na tecnologia Cordova, é o plugin cordova-plugin-camera-preview. A seguir o comando para adicionar o respectivo plugin no projeto recém-criado.

```
$ cordova plugin add https://github.com/luispimenta/cordova-plugin-camera-preview.git
```

Após a inclusão do plugin, efetue a chamada da câmera para inicializá-la conforme o exemplo abaixo:

```
// Update DOM on a Received Event
receivedEvent: function (id) {

    let options = {
        x: 0,
        y: 0,
        width: window.screen.width,
        height: window.screen.height - 50,
        camera: CameraPreview.CAMERA_DIRECTION.FRONT,
        toBack: true,
        tapPhoto: true,
        tapFocus: false,
        previewDrag: false,
        shutterSound: false
      };
    CameraPreview.startCamera(options);
}
```

Podemos criar uma DIV similar ao modelo abaixo, para exibir a imagem ao vivo.

```
<div id="fc_imgaux_p" style="height: 40px; position: absolute; z-index: 999999999; margin-left: 5px;">
```

```
    <img id="fc_imgaux" style="display: none;">
</div>
<div style="text-align: center; width: 80%">
    <img id="fc_imgaux_b" style="display: none; position: fixed; bottom: 10px;">
</div>
    <span id="counter" style="position: fixed; bottom: 10px; height: 25px; left: 30px; color:
#FFDD67"></span>
    <button value="none" type="button" id="start" style="position: fixed; bottom: 0px; width: 100%;
height: 40px;">Start</button>
</div>
```

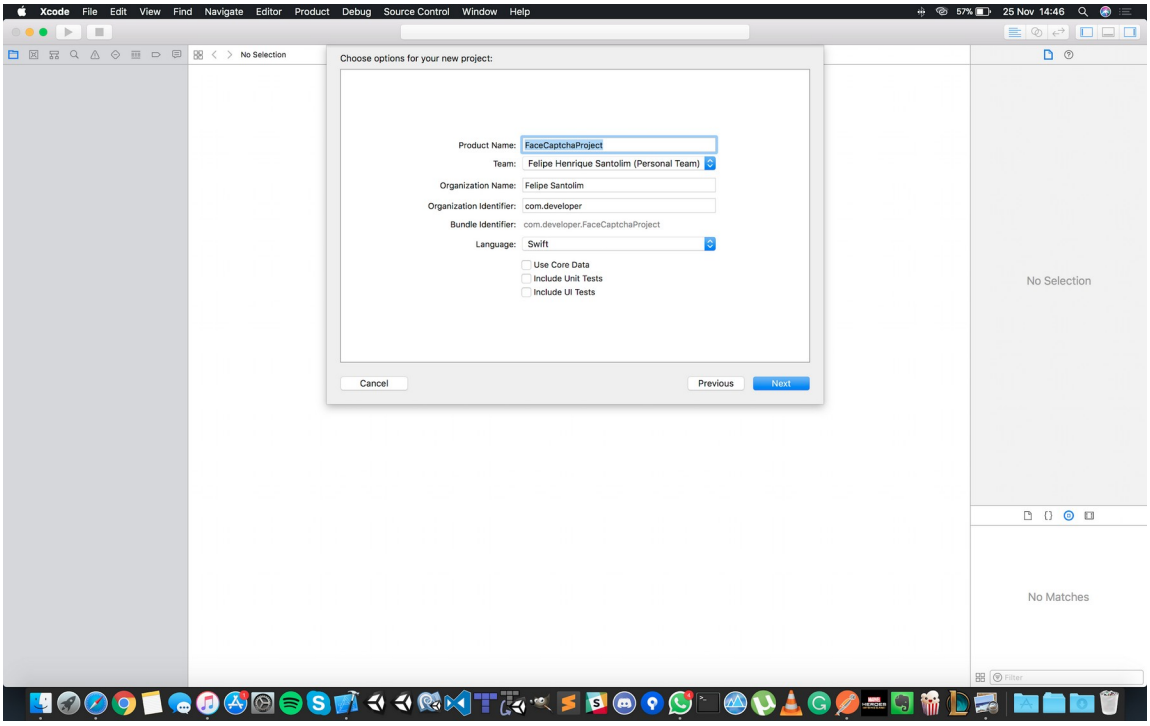
Agora devemos criar uma função cujo principal objetivo é inicia o fluxo da vídeo captura.

```
$(function () {
   $("#start").click(function () {
       start();
   });
});

function start() {
   appkey = 'chave';
   fcvarUrlbase = 'https://comercial.certiface.com.br:8443';
   userInfo = cripto_user('comercial.token', 'cpf', 'nome, '29/08/1989')
   //recebe os desafios
   getChallenge(encodeURIComponent(userInfo), function(){
       // começa as capturas da camera
       startCapture(function(){
           var enc = encChData(images);
           // valida desafio
           //return false;
           validChallenge(encodeURIComponent(enc), function(data){
               //console.log(data)
               $('#counter').hide()
               $('#start').show()
               if(data.valid){
                   alert(data.valid)
               }else{
                   alert("Valido : " + data.valid + "\nMotivo : " +  data.cause )
               }
           })
       })
   })
}
```

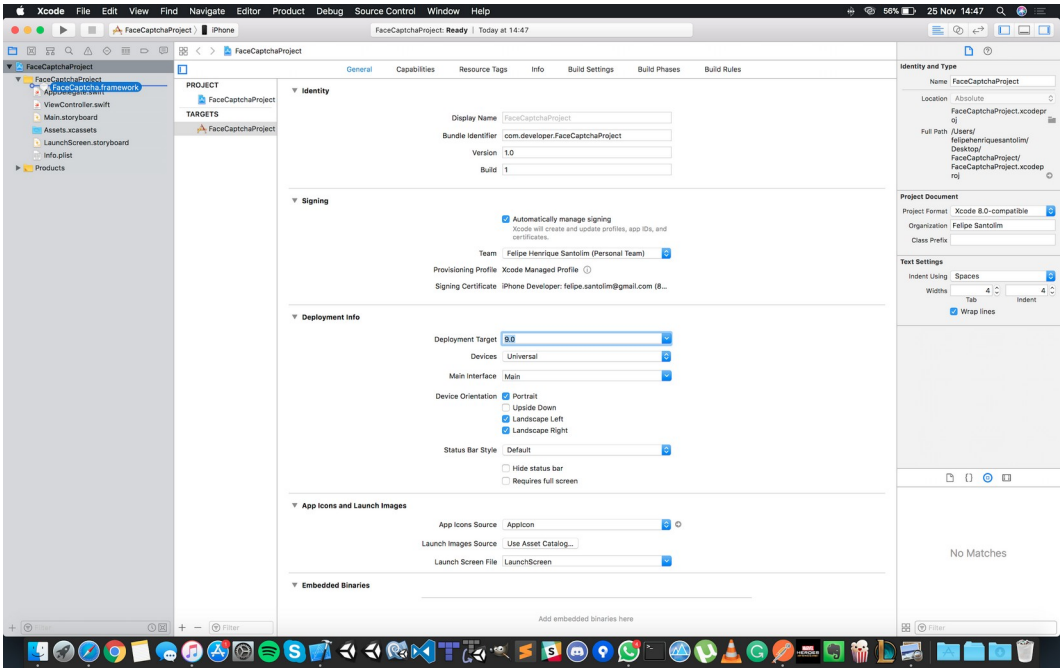
Será enviado ao JS o resultado do processamento Liveness, para um posterior tratamento do fluxo lógico da aplicação. Vale a pena ressaltar que este pluguin requer permissão de acesso ao hardware da câmera, tanto na plataforma Android como na plataforma iOS. Para adiciona uma imagem como Layer (sobre o vídeo ao vivo), basta inseri-la na pasta IMG com o nome overlay.png.

Integração do componente Liveness/Prova em iOS

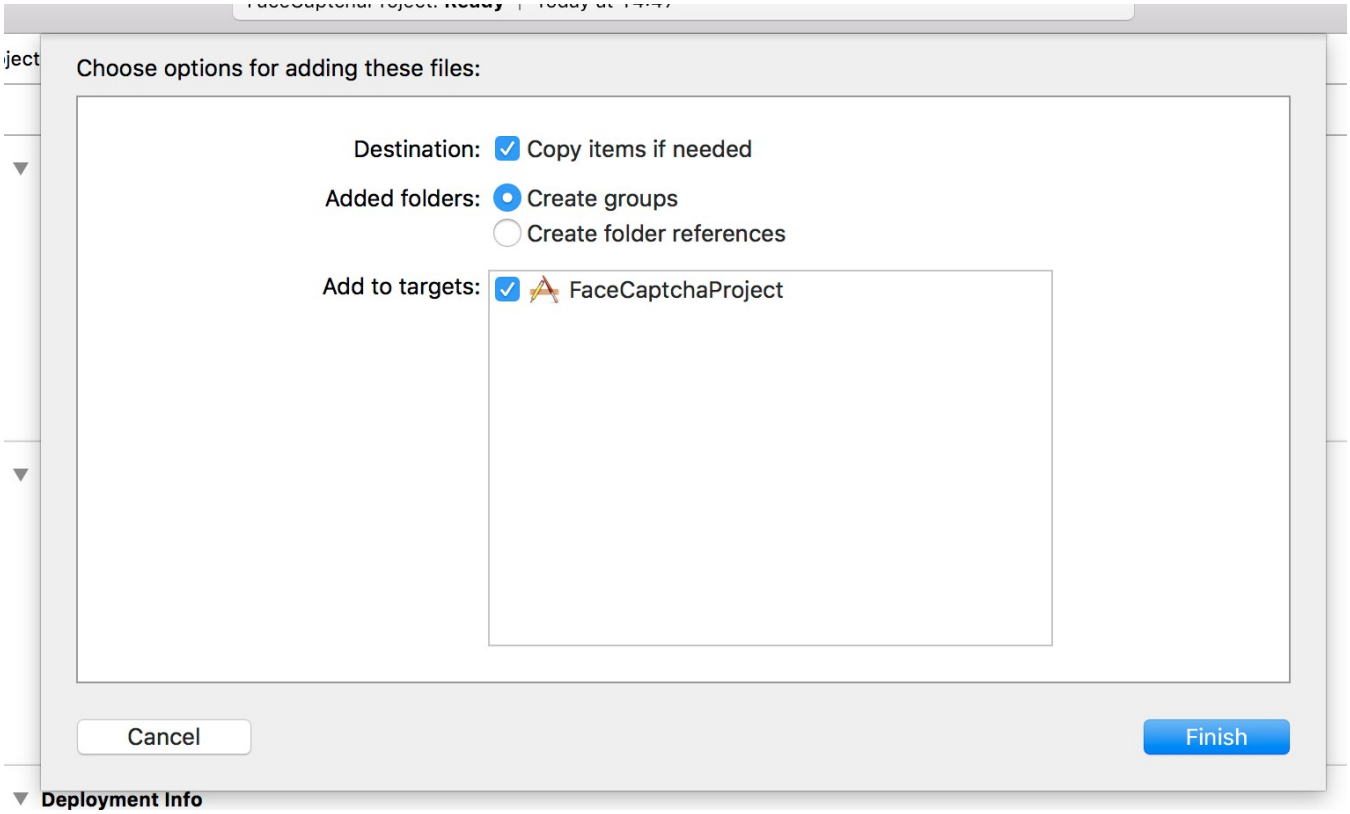
Crie um novo projeto no Xcode, insira os dados referente ao projeto e seleciona a linguagem Swift.



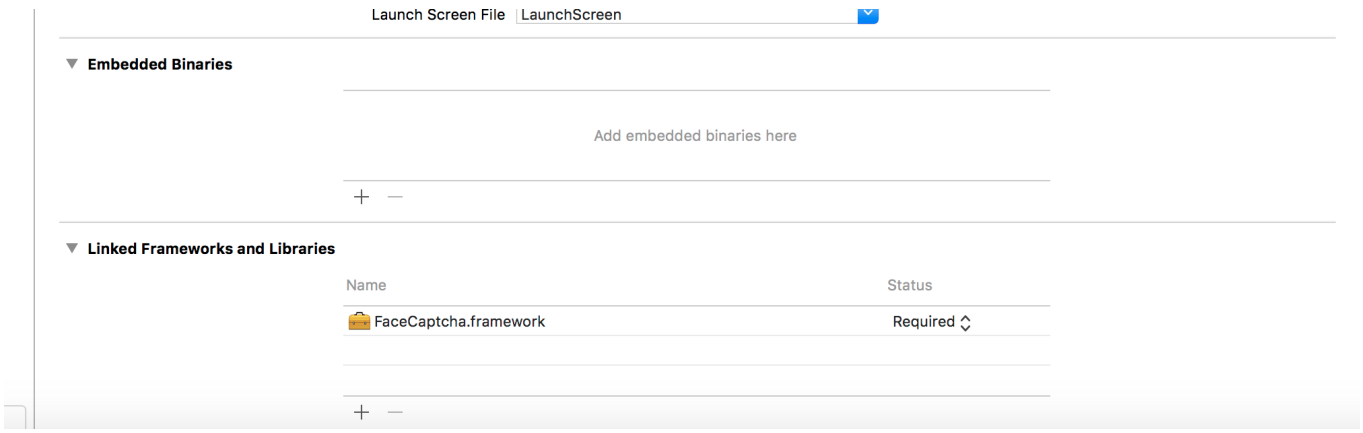
Com o recurso drag-in-drop, arraste o componente FaceCaptcha.framework para o projeto recém-criado conforme o exemplo a seguir:



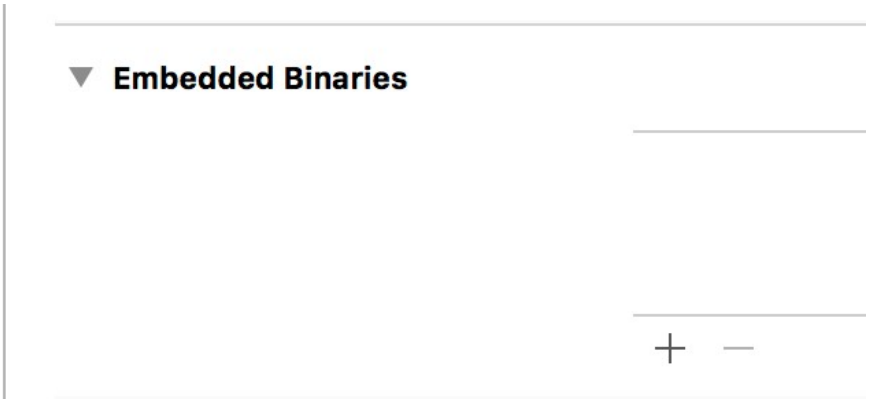
Como exibido na ilustração a seguir, selecione os campos seguindo o exemplo a seguir:

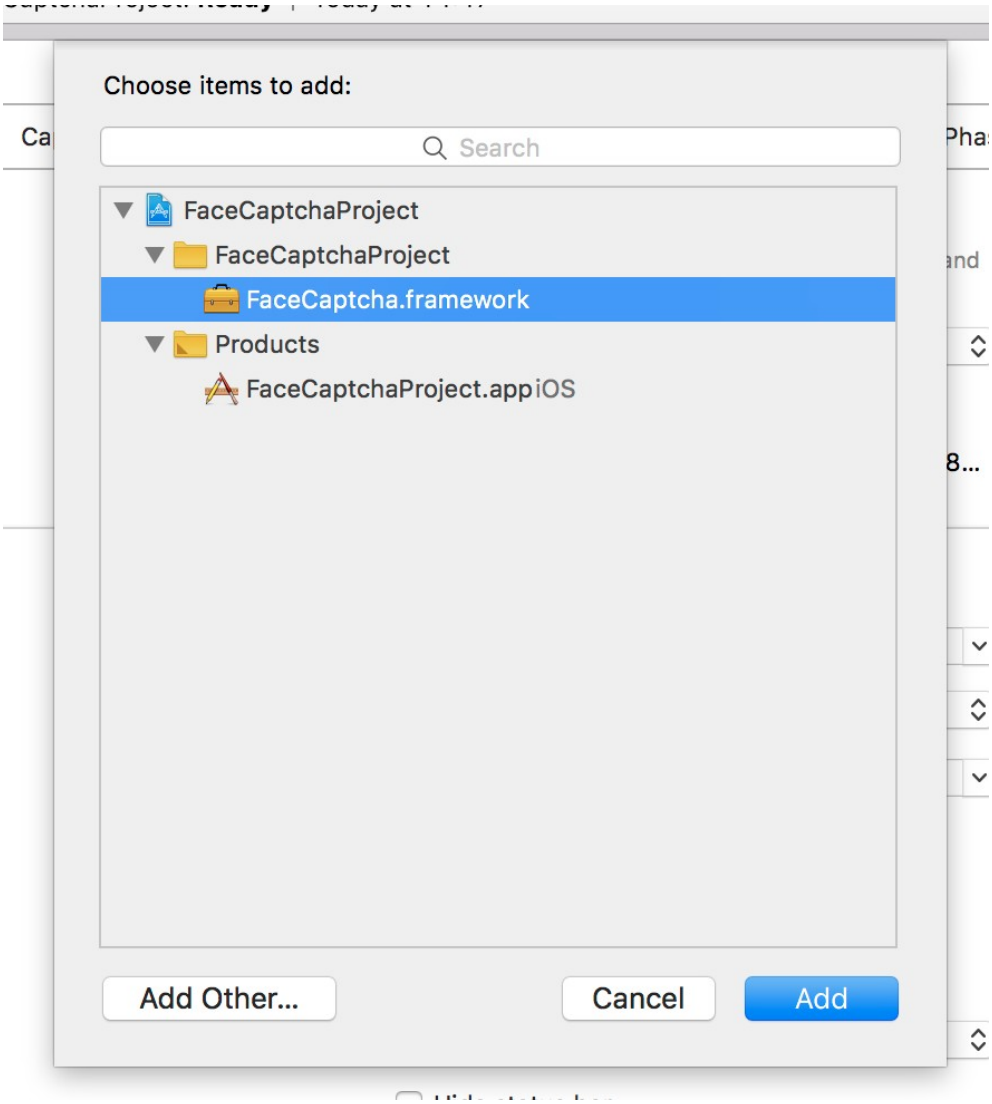


Configure o projeto conforme as opções ilustrações a seguir, clicando no ícone contendo o sinal “+”:

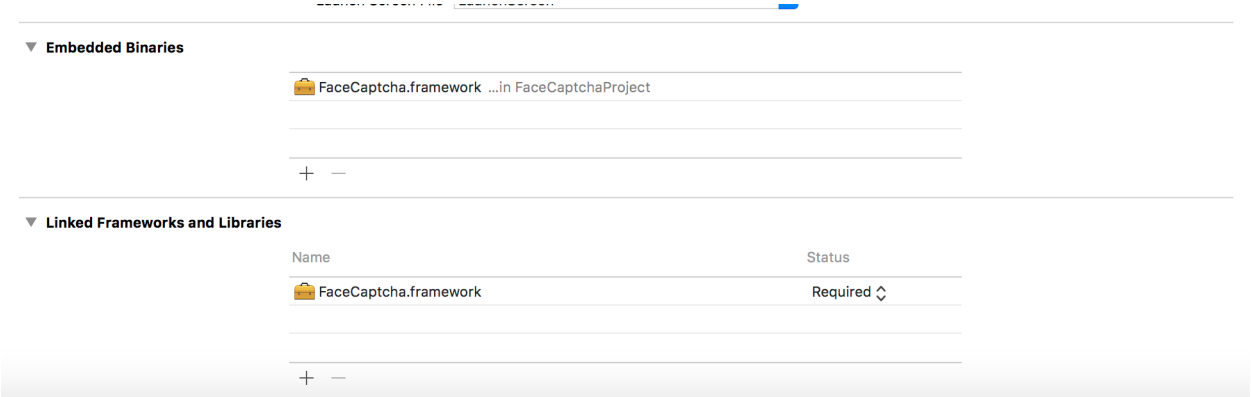


Em “Embedded Binaries” adicione o FaceCaptcha.framework:

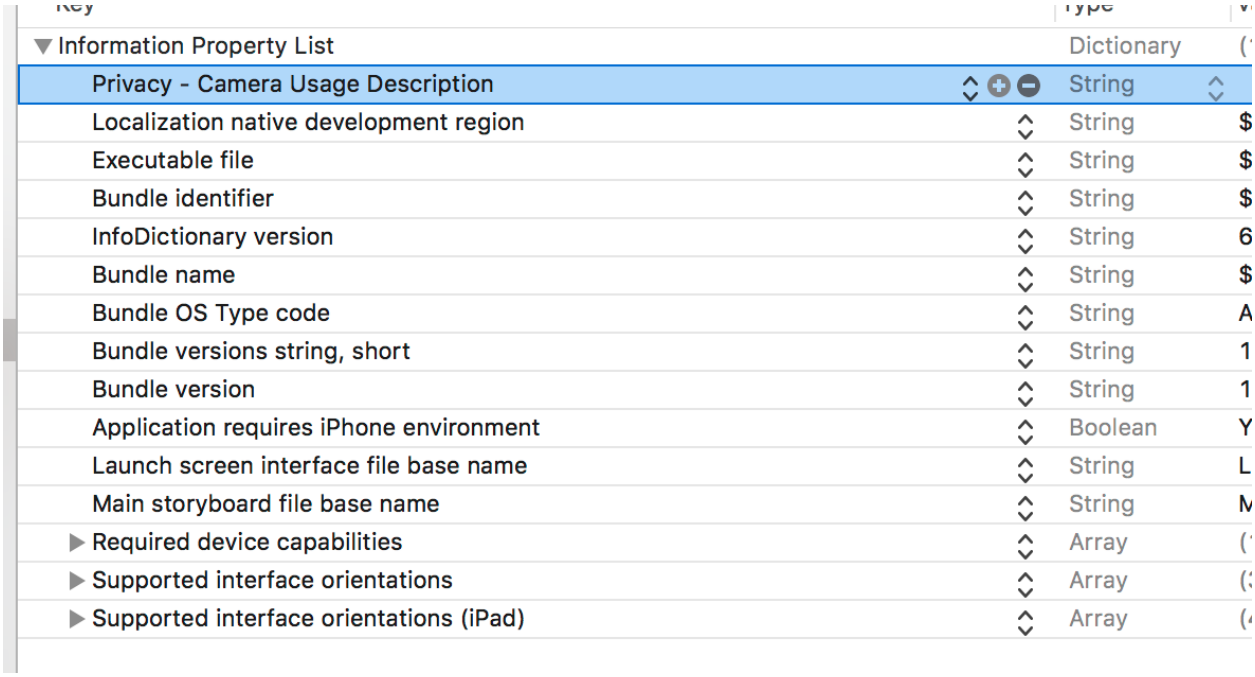




Se tudo estiver configurado corretamente, teremos uma interface similar a imagem a seguir:



Devemos adicionar a PERMISSÃO DE ACESSO A CÂMERA em info.plist do projeto (PROCESSO OBRIGATÓRIO PARA O FUNCIONAMENTO DO COMPONENTE). “Privacy - Camera Usage Description“.



A seguir um exemplo de implementação de código, primeiramente conforme o exemplo a abaixo na linha 10, insira o comando ***import FaceCaptcha.***

```
7  //
8
9  import UIKit
10 import FaceCaptcha
11
12 class ViewController: UIViewController {
13
14     override func viewDidLoad() {
15         super.viewDidLoad()
16         // Do any additional setup after loading the view, typically from a nib.
17     }
18
19     override func didReceiveMemoryWarning() {
20         super.didReceiveMemoryWarning()
21         // Dispose of any resources that can be recreated.
22     }
23
24
25 }
26
27
```

Agora implementaremos os dados e outros recursos no método openCamera()

```
@IBAction func openCamera() {

    let model = FCUserModel("felipesantolim",
                            "01234567890",
                            "FELIPE HENRIQUE SANTOLIM",
                            "11/11/1991")
    let fcCamera = FCCameraCapture(model, self, self)
    fcCamera.show()
}
```

Delegate (callback)

A implementação é simples, após criar um modelo “FCUserModel", o mesmo deve ser enviado como parâmetro para “FCCameraCapture(model, self, self)”, os demais parâmetros “self” e “self”, são referentes à UIViewController e a conformidade com o delegate “FCCameraCaptureDelegate".

Como de conhecimento para os desenvolvedore iOS, “FCCameraCaptureDelegate" obriga a implementação do método “handleCaptureVideo", no qual recebemos os frames/quadros como também o objeto. Bastando apenas invocar a closure validate enviando os parametros frames “fcImages”. O retorno do processamento , em seu retoré composto do objeto “FCValidCaptchaModel" contendo todas as informações necessárias.

```
36
37  // MARK: - FCCameraCaptureDelegate
38  extension ViewController: FCCameraCaptureDelegate {
39
40      func handleCaptureVideo(_ fcCameraCapture: FCCameraCapture, _ fcImages: [UIImage]) {
41          /* LOAD / PROGRESS HUD da sua escolha */
42          _ = fcCameraCapture.validate(fcImages, { [weak self] validResult, error in
43              /* RECEBE O VALID */
44          })
45      }
46  }
47
48
```