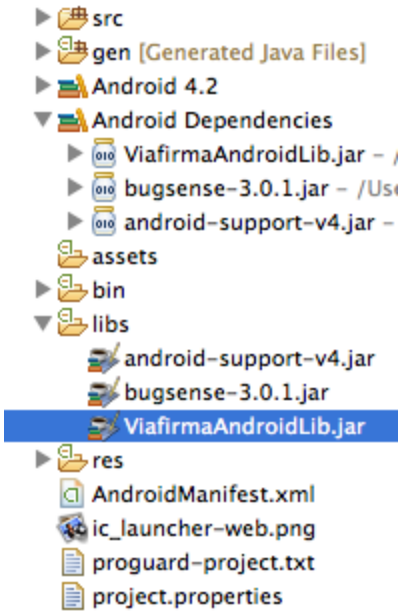
Viafirma SDK inApp API Android

versión 1.0

1. ÍNDICE

2. INSTALACIÓN DE LA LIBRERÍA (Android)

Para incluir la librería de Viafirma en su desarrollo, comience añadiendo ViafirmaAndroidLib.jar y bugsense-3.0.1.jar en su proyecto Android. Debe mostrarse entre las añadidas al proyecto, de forma parecida a la siguiente captura:



Ejemplo de proyecto con la librería ViafirmaAndroidLib

Necesitará los siguientes permisos en su AndroidManifest.xml:

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

3. INICIALIZACIÓN DE PARÁMETROS

Viafirma provee un único método de inicialización al que podremos indicarle las opciones de las que querremos hacer uso.

Si desea inicializar Viafirma con los valores por defecto, tan sólo deberá disponer de la url y de sus credenciales.

El método de inicialización es:

```
public static ViafirmaAndroidLib initWithOptions(Activity context, Properties params, String urlViafirma, String apiKey, String apiPass)
```

Donde:

- context : Contexto desde el cual ejecutaremos la inicialización.
- params : El conjunto de parámetros opcionales de inicialización de la librería. Valor por defecto `null`.
- urlViafirma : La URL a la cual se solicitarán las peticiones.
- apiKey y apiPass : Las credenciales de uso de licencia.

Ejemplo:

```
String urlViafirmaString = "http://testservices.viafirma.com/viafirma";
String viafirmaApiPass = "dev_servinbox";
String viafirmaApiKey = "12345";
ViafirmaAndroidLib api = ViafirmaAndroidLib.initWithOptions(this, null, urlViafirmaString, viafirmaApiKey, viafirmaApiPass);
```

4. OBTENCIÓN DE CERTIFICADOS INSTALADOS EN MEMORIA

Para poder acceder a los certificados, basta con arrastrarlos a la tarjeta sd o memoria del dispositivo. Una vez allí, el siguiente método del objeto creado en el apartado anterior nos provee la lista de rutas de todos los .pdf o .pfx encontrados.

public *List<CertificateEntity> loadCertificatesInstalled()*

Esto nos devolverá una lista de objetos de tipo CertificateEntity, cuyos campos principales son:

- title : Nombre del fichero .p12 o .pfx guardado en la aplicación.
- p12 : Ruta completa al fichero .p12 o .pfx guardado en la aplicación.
- certificateFullName : Nombre y apellidos completos del dueño del certificado.
- certificateName : Nombre del dueño del certificado.
- certificateSurnames : Apellidos del dueño del certificado.
- certificateNationalId : Número de identificación nacional.
- certificateEmail : Correo electrónico.
- certificateCa : Autoridad de certificación del certificado en formato corto.
- certificateCaShort : Autoridad de certificación del certificado.
- type : Tipo de certificado.

Nota: Los campos de datos personales, como el nombre o el email (es decir, todos aquellos que comienzan con la palabra "*certificate*", y el campo de "*type*"), tienen un valor nulo hasta que se haya ejecutado la autenticación. Por tanto, los CertificateEntity cargados en *api*.loadCertificatesInstalled() sólo tendrán completos los campos "*title*" y "*p12*".

Ejemplo:

```
// Imprime por logcat los nombres de los certificados instalados
List<CertificateEntity> certificates = api.loadCertificatesInstalled();
for (CertificateEntity certificateEntity : certificates) {
	Log.i("tag", certificateEntity.getTitle());
}
```

4.1. OBTENCIÓN DE CERTIFICADOS INSTALADOS EN ALMACÉN ANDROID

Android 4.0 Ice Cream Sandwich y superior permite la instalación de certificados en el propio sistema operativo.

Incluye un almacén que, cuando intentamos abrir un archivo reconocido como certificado, nos solicita la contraseña una sola vez para poder hacer uso del mismo en futuras ocasiones de forma más cómoda, rápida, y segura gracias a la administración del propio Android.

Viafirma es compatible con este almacén, que llamaremos “Keychain”. Por lo que es perfectamente válido el uso de certificados mediante este método.

En primer lugar, ofrecemos un método que nos permitirá saber si el dispositivo Android actual ofrece o no soporte para Keychain.

public boolean *isKeyChainSupported()*

Si la condición devuelta por el método previo se cumple, podremos hacer uso de la siguiente operación:

public *CertificateEntity getCertificateKeyChainRef()*

Nota: La mayoría de los campos del objeto CertificateEntity devuelto por Keychain serán nulos, hasta que se realice la autenticación.

Ejemplo:

```
if (api.isKeyChainSupported()) {
        CertificateEntity certificate = api.getCertificateKeyChainRef();
}
```

5. AUTENTICACIÓN Y CARGA DE LOS DATOS DEL CERTIFICADO

Sea cual sea el método usado para obtener el certificado, cabe recordar que hasta este punto no es más que un archivos .p12 cuyo contenido permanece oculto, ya que aún no nos hemos autenticado. Para poder hacerlo y ejecutar el resto de funciones, necesitaremos usar el siguiente método:

```
public void login(final CertificateEntity cert, final String pass, final ViafirmaAPILoginCallBack apiCallBack)
```

Si la contraseña introducida se corresponde con el pin válido del certificado, se ejecutará el siguiente método de la interfaz ViafirmaAPILoginCallback:

```
public void loginOk(CertificateEntity cert)
```

Donde el certificado devuelto ya estaría completo, con todos sus campos rellenos de forma correcta, tras haberse autenticado con éxito.

En caso de introducir mal la contraseña, o si ha habido algún error, se ejecutará el siguiente:

```
public void fail(String message)
```

Los valores devueltos en el argumento "*message*" nos informarán sobre el tipo de error ocurrido. El error por defecto es el de contraseña incorrecta, pero para poder controlar el resto, disponemos de una serie de constantes que representan distintos errores y que podremos usar para comparar.

- ViafirmaAndroidLib.*ERROR_CA_NOT_SUPPORTED*;
  - Autoridad de certificación no soportada por Viafirma.
- ViafirmaAndroidLib.*ERROR_EXPIRED_CERTIFICATE*;
  - El certificado ha caducado.
- ViafirmaAndroidLib.*ERROR_VIAFIRMA_CONNECTION*;
  - Error en la conexión con el servidor.
- ViafirmaAndroidLib.*ERROR_WITH_CERTIFICATE*;
  - Error con el certificado.
- Otro:
  - Contraseña incorrecta.

Ejemplo autenticación desde keychain, o memoria / tarjeta SD si no está soportado:

```
CertificateEntity selectedCertificate = null;
if (!api.isKeyChainSupported()) {
        // Nos autenticaremos con el primer certificado encontrado en memoria.
        List<CertificateEntity> certificates = api.loadCertificatesInstalled();
        selectedCertificate = certificates.get(0);
} else {
        // O nos autenticaremos con el certificado seleccionado en keychain.
        selectedCertificate = api.getCertificateKeyChainRef();
}

api.login(selectedCertificate, "password_test", new ViafirmaAPILoginCallBack() {

        public void loginOk(CertificateEntity cert) {
                Log.i("VIAFIRMA", "Usuario autenticado:" + cert.getCertificateFullName() + " - " +
        cert.getCertificateNationalId());
        }

        public void fail(String message) {
                if (message.equals(ViafirmaAndroidLib.ERROR_CA_NOT_SUPPORTED)){
                        Log.e("Viafirma", "Error: CA no soportada)");
                } else if (message.equals(ViafirmaAndroidLib.ERROR_EXPIRED_CERTIFICATE)){
                        Log.e("Viafirma", "Error: Certificado caducado");
                } else if (message.equals(ViafirmaAndroidLib.ERROR_VIAFIRMA_CONNECTION)) {
                        Log.e("Viafirma", "Error: Fallo de conexión");
                } else if (message.equals(ViafirmaAndroidLib.ERROR_WITH_CERTIFICATE)) {
                        Log.e("Viafirma", "Error: Fallo al leer el certificado");
                } else {
                        Log.e("Viafirma", "Error: Contraseña incorrecta");
                }
        }
});
```

6. FIRMA DE DOCUMENTOS CON CERTIFICADO DIGITAL

Para realizar una firma, y a partir del certificado del usuario que vaya a realizar la firma, tan sólo tendremos que ejecutar el siguiente método:

```
public void sign(
        final List<DocumentVO> documents,
        final CertificateEntity cert,
        final String pass,
        final Map<? extends String, ? extends String> policyParams,
        final ViafirmaAPILoginCallBack apiCallBack)
```

El primer parámetro "*documents*" deberá contener la lista de documentos a firmar, como una lista de objetos de tipo DocumentVO, este objeto consta de los siguientes campos básicos:

- Atributos de construcción:
- String filename : Nombre del documento.
- **byte**[] dataToSign : Documento en formato array de bytes.
- TypeFile typeFile : Tipo del documento. Será un enumerable con distintos tipos de formato. Los más usuales serán: "XML, TXT, PDF, PNG, JPG, HTML, ..."
- TypeFormatSign typeFormatSign : Formato de la firma del documento. Será un enumerable con distintos tipos de formato. Algunos de ellos serán: "XADES_EPES_ENVELOPED, CMS, PDF_PCKS7, CMS_ATTACHED, ..."
- Policy : Parámetros adicionales sobre el modo de firma empleado. Ver ejemplos para más información.

El segundo y tercer atributo se corresponderán con el certificado obtenido en los pasos previos, junto con el pin del mismo.

El tercer parámetro habilita una serie de opciones adicionales para casos concretos, en caso de duda aplicarlo tal como aparece los ejemplos adjuntos.

Análogamente para el caso de la autenticación, una vez el proceso esté completado se ejecutarán los correspondientes métodos de la interfaz ViafirmaAPILoginCallback. Para este caso de firma, deberemos fijarnos en dos:

```
public void signOk(CertificateEntity cert, String idSign)
```

Para el caso de ejecución exitosa. Se nos devolverá el certificado usado, así como el id de firma del proceso. En caso de error, se ejecutará:

public void fail(String message)

Ejemplo:

```
List<DocumentVO> documentsToSign = new ArrayList<DocumentVO>();

// Primer documento
String xmlDocument = "<test>Hola mundo! Documento 1</test>";
byte[] dataToSign =xmlDocument.getBytes();
DocumentVO document = new DocumentVO("test", dataToSign, TypeFile.XML);
document.setTypeFormatSign(TypeFormatSign.XADES_EPES_ENVELOPED);

Policy policy=new org.viafirma.client.sign.Policy();
policy.setTypeSign(TypeSign.ENVELOPED);
policy.setTypeFormatSign(TypeFormatSign.XADES_EPES_ENVELOPED);
document.setPolicy(policy);
documentsToSign.add(document);

// Segundo documento
String xmlDocument2 = "<test>Documento 2</test>";
byte[] dataToSign2 =xmlDocument2.getBytes();
DocumentVO document2 = new DocumentVO("test2", dataToSign2, TypeFile.XML);
document2.setTypeFormatSign(TypeFormatSign.XADES_EPES_ENVELOPED);

Policy policy2=new org.viafirma.client.sign.Policy();
policy2.setTypeSign(TypeSign.ENVELOPED);
policy2.setTypeFormatSign(TypeFormatSign.XADES_EPES_ENVELOPED);
document2.setPolicy(policy2);
documentsToSign.add(document2);

Map<String, String> policyParams=new HashMap<String, String>();

api.sign(documentsToSign, certificate, password, policyParams, new ViafirmaAPILoginCallBack(){

        public void signOk(CertificateEntity entity, String idSign){
                Log.i("Viafirma","Proceso de firma completado con id: "+idSign);
        }

        public void fail(String message){
                Log.i("Viafirma","Error : "+message);
        }
});
```