IES RIBERA DE CASTILLA

MÓDULO PROYECTO

CFGS Desarrollo de Aplicaciones Multiplataforma

Informática y Comunicaciones

Aplicación 'VarSign'

Tutor individual: *José María Rojo Zumel*

Tutor colectivo: *M.ª Cristina Silván Pardo*

Año: *2023 - 2024*

Fecha de presentación: *17/06/2024*

Nombre y Apellidos: Raúl Sastre Martín

Email: rasamadev@gmail.com

INDICE

1. DESCRIPCION GENERAL DEL PROYECTO

En este documento se recoge toda la información referente a la presentación de un trabajo de fin de grado para el ciclo formativo de grado superior "Desarrollo de Aplicaciones Multiplataforma" realizado en el I.E.S Ribera de Castilla durante el curso 2023 – 2024. El proyecto surge como propuesta de mi tutor de proyecto y profesor José María Rojo Zumel.

La idea es realizar una aplicación que sea capaz de aplicar una firma digital a uno o varios documentos de tipo PDF mediante el uso de un certificado digital importado en el dispositivo móvil, o el DNI electrónico.

Por tanto, se trata de un proyecto de desarrollo de una aplicación para Android realizada con el lenguaje de programación Kotlin, y a su vez, de investigación experimental en cuanto al estudio de las librerías utilizadas para la importación del certificado, la lectura del DNI electrónico y el proceso de implantación de la firma en el documento.

Los objetivos de este proyecto son tanto el cumplimiento de los requisitos funcionales de la aplicación como el desarrollo del aprendizaje del lenguaje de programación y entorno que se han utilizado.

2. OBJETIVOS

Desarrollar una aplicación móvil para el sistema operativo Android que sea capaz de realizar las siguientes acciones:

- Importar certificados digitales mediante un archivo de copia de seguridad '.pfx' o '.p12'.
- Reconocer y seleccionar documentos PDF desde el explorador de archivos, ya sea uno solo o varios documentos incluidos dentro de una carpeta.
- Leer un DNI electrónico mediante el uso de la tecnología NFC y extraer sus datos.
- Aplicar una firma digital en dichos documentos mediante el uso de certificado digital o DNI.
- Ofrecer un histórico de documentos firmados por el usuario con datos como el nombre del documento, la fecha y hora de la firma, el autor y el método de firma utilizado.

Con estas funcionalidades implementadas conseguimos una aplicación funcional y versátil para una persona que tenga la necesidad de firmar documentos en cualquier lugar sin necesidad de tener que disponer exactamente de un ordenador ni conexión a internet, utilizando únicamente su dispositivo móvil.

3. CUESTIONES METODOLOGICAS

Para este proyecto no se ha seguido ninguna metodología concreta de desarrollo de software. El primer paso que se dio después de tomar la decisión de realizarlo fue investigar si era posible llevar a cabo dichos objetivos descritos anteriormente, lo que poco a poco fue llevando a descubrir la existencia de distintas librerías encargadas de realizar diferentes funcionalidades necesarias para la elaboración del proyecto.

Después de encontrar que existen librerías con funciones capaces de realizar los objetivos que requiere el proyecto, se procede a realizar varias pruebas de funcionamiento para cada funcionalidad por separado con el fin de encontrar los métodos necesarios.

Por último, después de conseguir un funcionamiento óptimo de dichas librerías, se empieza a desarrollar la aplicación implementando las pantallas, menús y recursos a utilizar.

A medida que se va desarrollando la aplicación, se van encontrando diferentes dificultades las cuales se solventan mediante prueba y error y revisión de código, y según se van realizando esas pruebas va surgiendo la necesidad de añadir más comprobaciones con el fin de realizar una aplicación lo más robusta posible frente a errores y excepciones.

4. ENTORNO DE TRABAJO (Tecnologías de desarrollo y herramientas)

Para el desarrollo del proyecto se han utilizado las siguientes tecnologías de desarrollo y herramientas:

- Kotlin: Es un lenguaje de programación multiplataforma creado por la empresa JetBrains y diseñado para ser totalmente interoperable con Java. Kotlin se dirige principalmente a la 'Java Virtual Machine' pero también puede compilar a JavaScript para hacer aplicaciones front-web con React.

- Android Studio: Es el entorno de desarrollo integrado oficial para Android y el utilizado para el desarrollo de la aplicación 'VarSign'. Fue lanzados a finales de 2014 y desde entonces su desarrollo no ha parado de crecer.

- GitHub: Es una plataforma de desarrollo colaborativo que utiliza el sistema de control de versiones 'Git'. A medida que se han ido desarrollando diferentes funcionalidades para la aplicación, se han ido subiendo a esta plataforma con el fin de almacenarla y llevar el control de las diferentes ediciones del código.

- Dispositivos móviles Android: Se han utilizado diferentes modelos con distintas versiones de Android para realizar pruebas del funcionamiento de la aplicación.

5. VISION GENERAL DEL SISTEMA

Las funcionalidades básicas de la aplicación son poder realizar firmas digitales en uno o varios documentos a la vez mediante el uso de certificado digital o DNI electrónico. También se ofrece la funcionalidad de llevar un histórico de los documentos firmados con anterioridad.

La funcionalidad total de la aplicación puede estar limitada a ciertos usuarios que cumplan el requisito de tener un lector de NFC incorporado en su dispositivo, ya que es el canal de comunicación utilizado para la lectura del DNI electrónico, y por tanto, realizar operaciones de firma con el mismo. Aunque no tengan esa opción, existe la posibilidad de realizar firmas con un certificado digital instalado en el dispositivo.

La aplicación está pensada para usuarios que dispongan de la posibilidad de utilizar los métodos de firma disponibles, ya sea habiendo solicitado su certificado en la sede electrónica de la Fábrica Nacional de Moneda y Timbre, o conociendo la contraseña asociada a su DNI electrónico.

6. LIBRERIAS UTILIZADAS

Se han utilizado las siguientes librerías y métodos para desarrollar el funcionamiento de los objetivos de la aplicación:

- Android: Se han importado y utilizado las siguientes librerías:

- **androidx.biometric:** Proporciona una forma de confirmar la identidad del usuario mediante el uso de uno o varios métodos de autenticación, como PIN, patrón, contraseña, huella dactilar, entre otros. Esto supone la implementacion de una capa de seguridad para la aplicación frente a posibles firmas no deseadas.

- **androidx.datastore:** Es una solución de almacenamiento de datos que permite almacenar datos en pares clave-valor de manera asíncrona y coherente mediante el uso de corrutinas en Koltin. De esta forma se guarda dentro de la aplicación el histórico de documentos firmados con esta.

- **androidx.lifecycle:** Librería implementada para la utilización de corrutinas dentro de la aplicación requeridas para ciertos métodos que funcionan de forma asíncrona.

- **DNIeDROID:** Es un middleware, es decir, un software que gestiona la conexión entre el dispositivo móvil y el DNI electrónico. Ofrece una API que nos permite implementar las operaciones de lectura del DNI, firmado de documentos y control de posibles fallos durante el proceso de firma.

- Bouncy Castle: Es una colección de APIs dedicadas a la criptografía, de las cuales utilizamos algoritmos de cifrado y protocolos para la realización de la firma en un documento. También se implementa una versión denominada 'SpongyCastle' diseñada exclusivamente para Android.

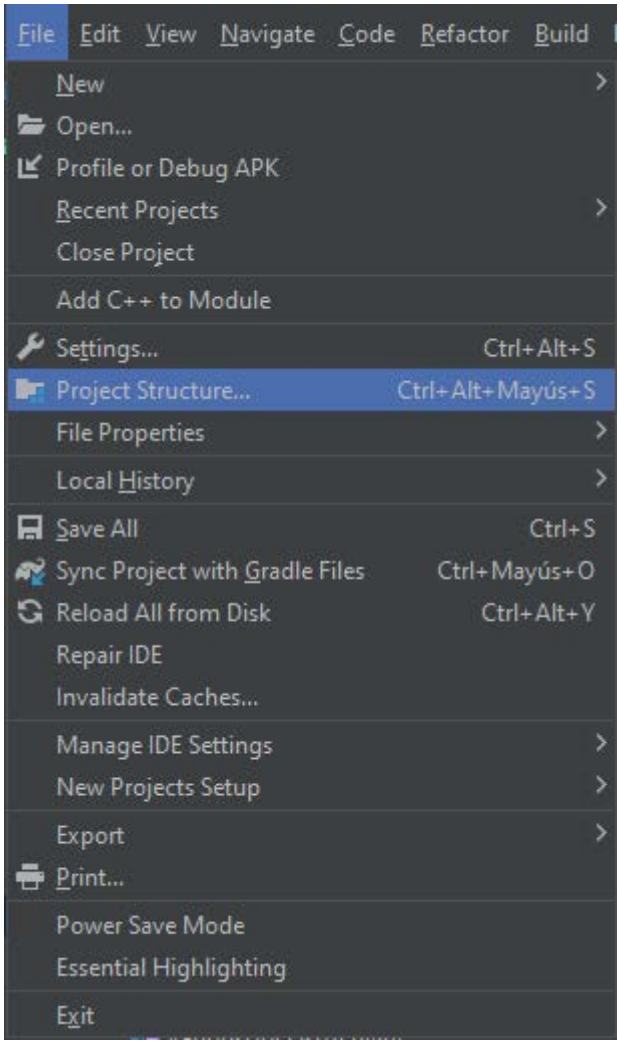
- **iText**: Es una librería OpenSource que nos permite realizar varias operaciones con archivos PDF, entre las cuales esta añadirle una firma digital. Principalmente está pensada para proyectos que usen Java o C#, pero en este proyecto se usa una versión llamada 'iText G' pensada para el sistema operativo Android, que elimina ciertos paquetes que solo pueden ser utilizados por el JDK de Java.

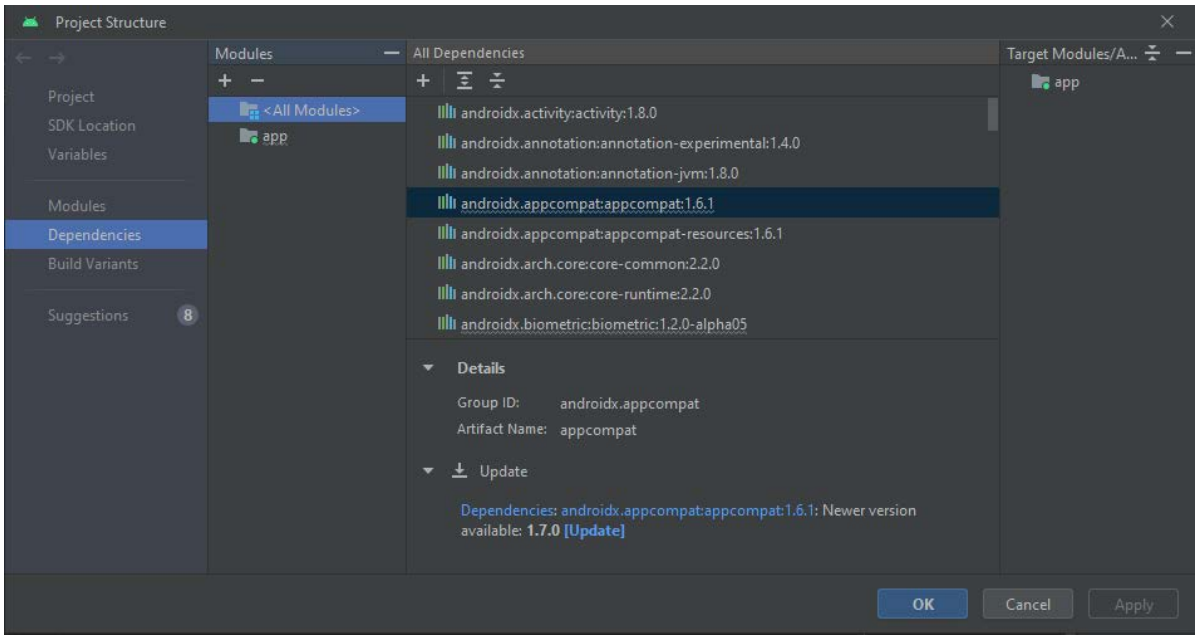
A continuación, se muestra una captura de una parte del archivo 'build.gradle.kts', en el cual se implementan las dependencias utilizadas mediante el gestor de dependencias 'Maven':

```
dependencies { this: DependencyHandlerScope

    implementation("androidx.core:core-ktx:1.9.0")
    implementation("androidx.appcompat:appcompat:1.6.1")
    implementation("com.google.android.material:material:1.12.0")
    implementation("androidx.constraintlayout:constraintlayout:2.1.4")

    // AUTENTICACION BIOMETRICA
    implementation("androidx.biometric:biometric:1.2.0-alpha05")
    // DATASTORE
    implementation("androidx.datastore:datastore-preferences:1.1.1")
    // USO DE CORRUTINAS
    implementation("androidx.lifecycle:lifecycle-runtime-ktx:2.8.1")

    // LIBRERIA 'iText G' PARA ANDROID
    implementation("com.itextpdf:itextg:5.5.10")

    // BOUNCE CASTLE Y SPONGY CASTLE
    implementation("org.bouncycastle:bcprov-jdk15on:1.65")
    implementation("org.bouncycastle:bcpkix-jdk15on:1.65")
    implementation("org.bouncycastle:bctls-jdk15on:1.65")
    implementation("com.madgag:scpkix-jdk15on:1.47.0.2")
    implementation("com.madgag:scprov-jdk15on:1.47.0.2")

    // LIBRERIA 'DNIeDROID'
    implementation(files( ...paths: "libs\\dniedroid-release.aar"))

    testImplementation("junit:junit:4.13.2")
    androidTestImplementation("androidx.test.ext:junit:1.1.5")
    androidTestImplementation("androidx.test.espresso:espresso-core:3.5.1")
}
```

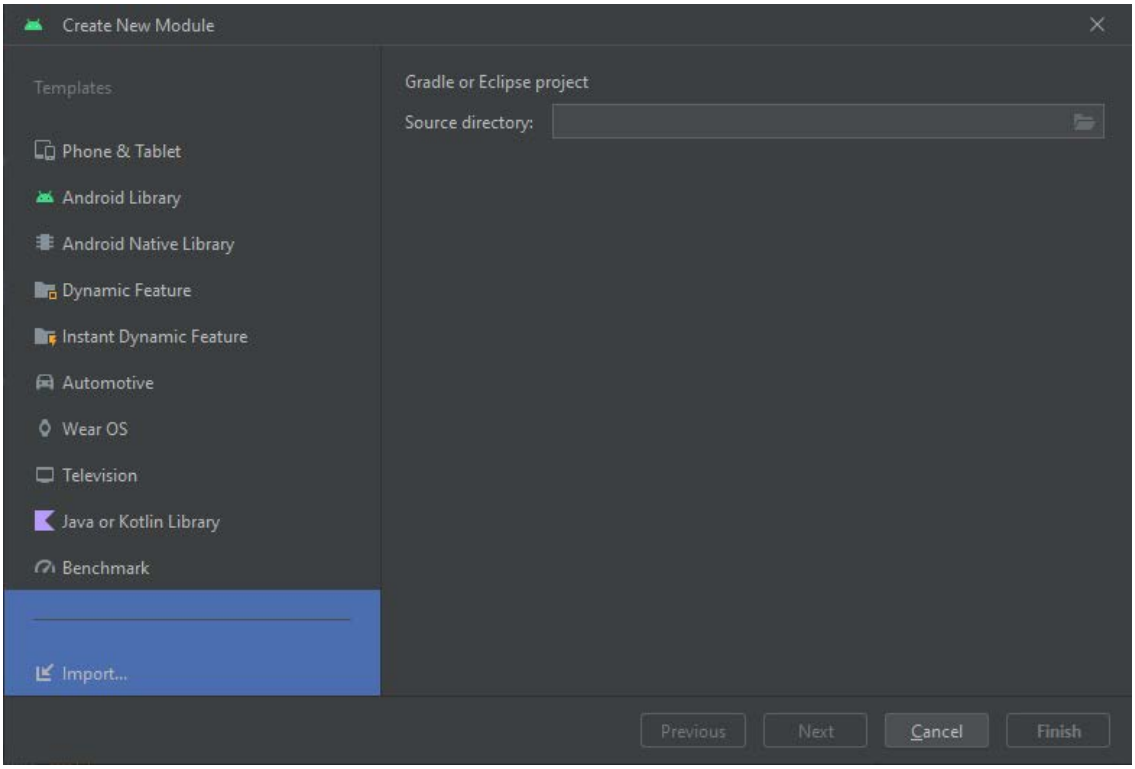
Para la importación de la librería DNIeDROID se realiza un proceso distinto antes de importarla en el archivo 'build.gradle.kts', el cual consiste en añadir primero el archivo de librería 'dniedroid-release.aar' en una carpeta dentro del proyecto. En este proyecto esta añadido en el directorio 'app/libs'.

Después de añadir el archivo en una carpeta dentro del proyecto, lo importaremos en Android Studio entrando, en la barra de menus, en File > Project Structure > Dependencies.

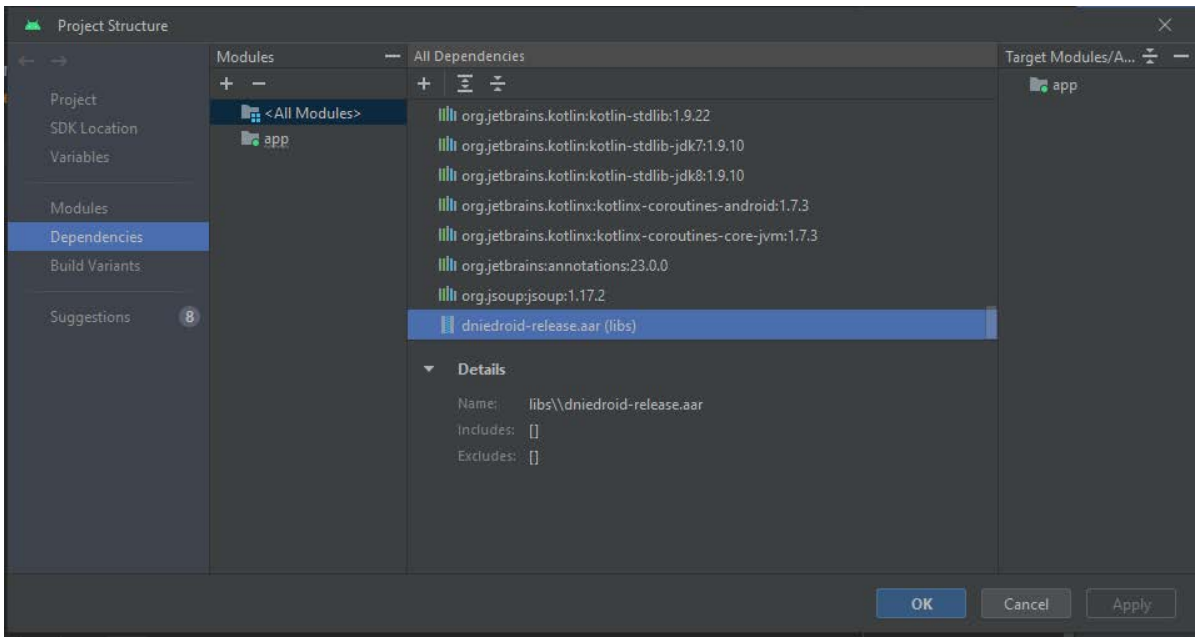




Una vez estemos en ese menú, pulsamos en el icono '+' de la columna de modules y se abrirá otra pantalla donde podremos directamente importar el archivo pulsando en la opción "Import" de abajo del todo.



Cuando lo hayamos seleccionado, en la lista de todas las dependencias aparecerá la librería que acabamos de seleccionar, por lo que ya podemos pulsar en 'OK' e implementarla mediante el archivo 'build.gradle.kts' del proyecto.



7. DOCUMENTACION TECNICA

7.1. ESPECIFICACION DE REQUISITOS

Para que la aplicación se pueda instalar en nuestro dispositivo móvil, es necesario que la versión mínima del mismo sea 'Android 7.0'. Esto es debido a que en el archivo 'build.gradle.kts' de la aplicación se ha establecido que la versión mínima del SDK (*Software Development Kit*) utilizado por Android sea 25, ya que algunos métodos así lo requieren.

También, como hemos indicado anteriormente, para tener una experiencia de uso completa de la aplicación, se requiere que el dispositivo móvil cuente con lector de NFC incorporado.

7.2. DISEÑO DEL ALMACENAMIENTO DE DATOS

Debido a la funcionalidad de la aplicación es firmar documentos con métodos agregados de forma externa, como el certificado digital y el DNI electrónico, la aplicación no cuenta ni necesita un sistema de gestión de usuarios ni bases de datos complejas que almacenen una gran cantidad de datos.

Respecto al almacenamiento de datos, la librería 'DNIeDROID' utiliza una clase propia denominada 'CANSpecDOStore', que se encarga de crear una pequeña base de datos que guarda los números CAN que hayamos añadido para después utilizarlos en el proceso de firma con DNI electrónico. También, al realizar una firma con un CAN concreto, se guardan los datos del nombre del certificado y DNI de ese CAN.

Esta base de datos utiliza el sistema gestor 'SQLite' y es lo suficientemente ligera para ser almacenada en los datos de la aplicación.

También, como hemos comentado anteriormente, almacenamos los registros de los documentos que hemos firmado en un string que se guarda dentro de los datos de la aplicación a través de la librería 'Datastore'.

Cada vez que se firma un documento, se guarda un registro con los siguientes datos y en el siguiente orden:

- Nombre del documento
- Fecha y hora de la firma
- Método de firma utilizado
- Alias del certificado utilizado.

Cada registro se almacena en una única cadena con el formato:

"firmado_NombreDocumento?dd-MM-yyyy HH:mm:ss<MetodoFirma>AliasCertificado"

Estos registros están separados por el símbolo '|' para después hacer un 'split' de todo el string y separarlos usando ese símbolo a modo de delimitador.

Una vez separados, se vuelve a 'splitear' cada registro usando los símbolos '?', '<' y '>' para obtener los diferentes datos y mostrarlos, todo gracias a la clase 'Adapter' que se usa en el histórico.

Cada registro ocupa aproximadamente 45 - 49 caracteres (45 firmando con DNI, 49 con certificado digital) más lo que ocupen el nombre del documento y alias del certificado.

Teniendo en cuenta que un string puede almacenar 2^31 – 1 caracteres, es decir, 2.147.483.647 caracteres, es probable que entren más de tres millones de registros.

7.3. DISEÑO DE LA APLICACIÓN E INTERFAZ

A continuación, se especifican las diferentes pantallas implementadas en la aplicación y sus características. Para ver una demostración grafica de las mismas, revise el manual de usuario.

7.3.1. PANTALLA DE INICIO

Es la primera pantalla que se muestra nada más iniciar la aplicación. Si es la primera vez que se usa, podremos utilizarla después de comprobar que tenemos un método de bloqueo de seguridad configurado en nuestro dispositivo y de aprobar los permisos de almacenamiento necesarios.

A través de ella se pueden acceder a todas las funcionalidades, como instalar nuestro certificado digital, añadir un código CAN, firmar uno o varios documentos o ver el historial de documentos firmados.

Cuando hayamos firmado uno o varios documentos con el DNI electrónico, o si hemos firmado con el certificado digital mediante la opción de "Varios documentos", se nos llevara de nuevo a esta pantalla y se mostrara un mensaje de confirmación de que se ha realizado la firma con éxito.

También cuenta con dos apartados de ayuda al usuario para quien no sepa instalar un certificado digital en el dispositivo, o que sepa que es el código CAN del DNI.

7.3.2 INTERFACES DE INSTALACION DE CERTIFICADOS DIGITALES

Durante la instalación de un certificado digital se irán mostrando diferentes diálogos de solicitud de contraseña, introducción de un alias para el certificado y seleccionar el almacén donde que se va a guardar. En algunos dispositivos, la introducción del alias y selección del almacén van en el mismo dialogo, dependiendo de la versión de Android que se ejecute.

También existe un dialogo con una lista de los certificados que tengamos instalados en nuestro dispositivo de los cuales podremos seleccionar uno de ellos para utilizarlo en el proceso de firma.

De esto se encarga la clase 'KeyChain' de Android, que nos brinda dos métodos distintos:

- createInstallIntent: Nos devuelve los distintos diálogos para la introducción de la contraseña del certificado y selección de un almacén y alias mediante un objeto de tipo 'Intent', que usaremos como parámetro en el método 'startActivity()', el cual muestra los diálogos.

- choosePrivateKeyAlias: A través de este método se muestra el dialogo de selección de uno de los certificados instalados en nuestro dispositivo móvil. Como parámetro se introduce un objeto de tipo 'KeyChainAliasCallback', el cual utiliza un método denominado 'alias()' que nos permite controlar la acción a realizar después de seleccionar el certificado.

7.3.3. PANTALLA DE SELECCIÓN DE UN DOCUMENTO

Cuando seleccionemos la opción de firma de "Un documento" en el menú que se nos muestra después de pulsar en el botón de "Firmar documento(s)" del menú principal, se nos mostrara una pantalla que contiene los siguientes apartados:

- Selección de documento: Arriba del todo a la derecha se encuentra un botón "Seleccionar" que, al pulsarlo, abre el explorador de archivos del dispositivo con el objetivo de

elegir exclusivamente un documento de tipo PDF, ya que se nos impedirá seleccionar cualquier otro tipo de archivo. Una vez seleccionado, se cargará su nombre en el elemento 'EditText' de su derecha o, si es un archivo protegido con contraseña, se mostrará un dialogo de introducción para introducirla.

- Selección de pagina de la firma: El apartado del medio mostrara por defecto que no se ha seleccionado aun un documento. Después de cargar un documento, se mostrará el número de páginas de este junto con un apartado donde podremos introducir un valor numérico que representará la página donde aplicaremos la firma. Si intentamos firmar habiendo introducido un numero incorrecto, se mostrará un error.

- Selección de posición de la firma: Por último, se nos dará la opción de seleccionar donde ira posicionada la firma mediante un grupo de 'RadioButtons', cada uno con una posición. Por defecto, aparece seleccionada la posición 'Arriba a la izquierda'

- Botón de 'Firmar documento': Al pulsarlo, la aplicación comprobará si hemos seleccionado antes un documento y si hemos introducido un numero de pagina de firma que entre en el rango del total de paginas del documento. En el caso de que alguna de estas dos comprobaciones no se cumpla, se nos mostrara un error.

  También comprueba si ya existe un documento en la carpeta 'VarSign' del dispositivo móvil con el nombre "firmado_NombreDelDocumento" y, en el caso de que exista, muestra una advertencia de si queremos seguir con el proceso de firma.

Cuando todo este correctamente, se nos pedirá que indiquemos el método de firma que queremos usar.

7.3.4. PANTALLA DE SELECCIÓN DE VARIOS DOCUMENTOS

Cuando seleccionemos la opción de firma de "Varios documentos" en el menú que se nos muestra después de pulsar en el botón de "Firmar documento(s)" del menú principal, se abrirá el explorador de archivos del dispositivo para que seleccionemos una carpeta.

Una vez seleccionada, primero comprobará si contiene documentos PDF dentro de la misma. En el caso de no tener ninguno, se mostrará un mensaje de error.

Si la carpeta contiene documentos, se guardarán todos en una lista y se eliminarán los que estén corrompidos y los que tengan contraseña. Después, pasaremos a una pantalla donde se mostrara esa lista de documentos y, en el caso de haber encontrado algún documento con contraseña, se mostrara un mensaje indicando que estos documentos se han omitido y que deben firmarse mediante la opción de "Un documento".

Los documentos que contenga la carpeta se mostraran en una lista de 'CheckBox', los cuales podremos seleccionar y deseleccionar para elegir cuales de esos documentos queremos firmar. Por último, tendremos un botón de "Cancelar" para volver al menú de inicio y un botón "Aceptar" para proceder con la firma.

Si pulsamos en el botón de "Aceptar" sin haber seleccionado ningún documento, se nos mostrara un mensaje de error. Cuando hayamos

seleccionado uno o varios documentos de la lista, comprobara con cada uno si en la carpeta "VarSign" del dispositivo móvil existe una copia ya firmada con el mismo nombre (igual que cuando se va a firmar un documento) y en el caso de que coincidan uno o varios, se mostrara una advertencia que nos permitirá elegir si continuar con la firma o cancelarla.

Cuando todo este correcto, nos aparecerá un dialogo con un grupo de 'RadioButtons' que nos permitirá indicar la posición en la que ira la firma. La posición seleccionada se aplicará a todos los documentos por igual. Después de seleccionarla, se nos preguntara por el método de firma a utilizar.

7.3.5 PANTALLA DE FIRMA CON DNI ELECTRONICO

Después de indicar que queremos realizar la firma mediante el DNI electrónico y haber seleccionado el CAN, nos aparecerá una pantalla con una imagen de un DNI colocado de forma vertical a modo de referencia de como colocarlo para que sea detectado por el lector NFC, un mensaje que indica que aproximemos el DNI al lector y un botón de cancelar, que volverá a la pantalla de uno o varios documentos, dependiendo de la opción que hayamos seleccionado.

Cuando el lector NFC detecte nuestro DNI, lo sabremos por el cambio del mensaje de la pantalla a "Leyendo datos, no aparte el DNI…" y el cambio de color de la imagen del DNI.

El primer paso que comprobara la aplicación es si el numero CAN seleccionado coincide con el que viene en el DNI. En el caso de que no sea así, se actualizara el mensaje de la pantalla con un error.

Cuando haya detectado que ambos números coinciden, se mostrara un dialogo en la pantalla que solicita la contraseña del certificado para poder realizar la firma. Si cancelamos la solicitud, se actualizará el mensaje de la pantalla indicando dicha acción. También, si introducimos una contraseña incorrecta, nos saldrá de nuevo el mismo dialogo, pero indicando el número de intentos que nos quedan para volver a introducir la contraseña.

Una vez introducida la contraseña correcta, saldrá un último dialogo de confirmación de la firma. Cuando confirmemos la operación, volveremos al menú de inicio de la aplicación y se mostrará un mensaje indicando que la firma del documento (o documentos) seleccionado(s) se ha realizado correctamente.

Si en algún momento alejamos el DNI del lector o se pierde el acceso al mismo, se cancelará el proceso de firma y se indicará mediante la actualización del mensaje de la pantalla.

Todo el proceso de lectura del CAN, solicitud de la contraseña del DNI y extracción del certificado de firma del mismo se lleva a cabo a través de los métodos de la librería 'DNIeDROID'.

7.3.6. PANTALLA DE HISTORICO DE DOCUMENTOS

En el menú de inicio de la aplicación tenemos un apartado de "Documentos firmados" que, al pulsarlo, comprobara si hemos firmado anteriormente algún documento con la aplicación. En el caso de no haber firmado nada, se mostrará un mensaje de error.

Cuando hayamos firmado algún documento, podremos abrir el histórico y ver todos los registros de los documentos que hemos firmado, la fecha y hora en la que se firmaron, el método que se utilizo para firmarlo (certificado o DNI) y la persona que realizo dicha firma.

Al pulsar en un registro, si ese documento sigue en la carpeta "VarSign" del dispositivo, se abrirá con la aplicación predeterminada del dispositivo para abrir documentos de tipo PDF. En el caso de que el documento no se encuentre porque se haya borrado, movido o cambiado de nombre, se mostrara un mensaje de error.

7.3.7. MENUS Y DIALOGOS.

A lo largo del uso de la aplicación iremos viendo menús y diálogos para seleccionar diferentes opciones o indicar distintos errores o advertencias.

Al entrar en la aplicación, en la pantalla del menú de inicio, se podrían mostrar diálogos de solicitud de permisos de almacenamiento o de configuración de un método de bloqueo de seguridad en el

dispositivo si no se cumplen las condiciones. Ambos diálogos ofrecen un acceso directo a los ajustes para configurar estas opciones.

Otros diálogos que se pueden mostrar a lo largo de la aplicación son:

- Cuando no puedes añadir un CAN debido a que el dispositivo no cuenta con lector NFC.

- Cuando no has firmado ningún documento con la aplicación.

- Cuando has seleccionado una carpeta que no contiene documentos PDF.

- Cuando seleccionamos un archivo que esta erróneo.

- Cuando pulsamos el botón "Firmar documento" de la pantalla de firma de "Un documento" sin haberlo seleccionado.

- Cuando indicamos un numero incorrecto para la pagina donde aplicar la firma en un documento.

- Advertencia de cuando existen copias ya firmadas y con el mismo nombre de uno o varios documentos seleccionados que vamos a firmar.

- Cuando vamos a firmar mediante el método de firma del DNI y no hemos añadido un código CAN en la aplicación.

- Solicitud de confirmación cuando se procede a firmar un documento, ya sea con DNI electrónico o certificado digital.

- Solicitud de confirmación cuando se procede a firmar varios documentos junto con una lista de los mismos.

- Confirmación en el menú de inicio de que se han firmado uno o varios documentos con el método de firma seleccionado.

- Cuando pulsamos en un registro del historial de documentos firmados y ese archivo ya no se encuentra en la carpeta 'VarSign'.

- Mensaje de confirmación de que se han firmado uno o varios documentos con el método de firma seleccionado. Aparece en el menú de inicio de la aplicación.

Respecto a los menús, pueden ser de introducción de un dato o de elección de una opción:

- Menú para introducir un código CAN y añadirlo a los datos de la aplicación.

- Menú para seleccionar firmar uno o varios documentos.

- Menú para introducir la contraseña de un documento protegido.

- Menú para seleccionar el método de firma a utilizar (certificado digital o DNI electrónico)

- Menú para seleccionar la posición de la firma, habiendo escogido anteriormente la opción de firmar varios documentos.

- Menú que aparece después de haber firmado un único documento utilizando un certificado digital. Se nos

muestran las opciones de abrir el documento firmado, volver al menú de inicio o simplemente cerrar ese menú.

7.4. ESTRUCTURA DEL PROYECTO

El proyecto esta dividido en varios archivos:

7.4.1. ARCHIVOS DE CONFIGURACION

Para que una aplicación desarrollada para Android funcione correctamente, necesita unos archivos de configuración concretos. A continuación, se exponen los más importantes:

- **AndroidManifest.xml:** Se encuentra en el directorio 'app/src/main' del proyecto. En el se incluyen los permisos que necesita la aplicación, un proveedor de contenido que administra el acceso a datos del dispositivo, las pantallas que va a utilizar la aplicación ('Activitys') y algunos detalles como el icono, el nombre a mostrar, el tema a utilizar, entre otros.

- **build.gradle.kts:** Se encuentra en la carpeta 'app' del proyecto. En el se establecen algunas configuraciones como el id de la aplicación, la versión mínima del SDK que necesita para funcionar, entre otras muchas. También es donde añadimos las dependencias que va a usar nuestra aplicación mediante Gradle, un sistema de compilación que automatiza y gestiona el proceso de construcción de la app sobre los conceptos de Apache Maven.

7.4.2. ARCHIVOS DE CODIGO

Podemos clasificar los archivos del proyecto en diferentes utilidades:

- Adapters: Están incluidos en el paquete 'com.rasamadev.varsign.adapter'. Son unos archivos de código que acoplan unos datos concretos a un elemento de una actividad, como por ejemplo, una lista. En este proyecto se encuentran dos adapters:

  - AdapterCanList.java: Adapter que se utiliza para recoger los códigos CAN almacenados en la base de datos 'CANSpecDOStore' e introducirlos en una lista que se mostrará cuando vayamos a firmar un documento con DNI electrónico. También introduce al lado de cada código CAN un botón con un icono de una papelera que al pulsarlo elimina ese código de la base de datos.

  - AdapterSignedDocsHistory.kt: Adapter que se utiliza para mostrar el histórico de documentos firmados. Recoge el string del histórico guardado en el 'Datastore' de la aplicación y lo va 'spliteando' a través de distintos delimitadores para mostrar los distintos datos en apartados distintos de cada elemento que se mostrará en la lista. También incorpora en cada elemento un 'clickListener' para controlar la acción que ocurrirá al pulsarlo.

- Utilidades: Están incluidos en el paquete 'com.rasamadev.varsign.utils'. Contienen archivos con diferentes clases y métodos que proporcionan utilidades al proyecto:

  - Dialogs.kt: Contiene la clase 'Dialogs' en la cual se encuentran dos métodos de tipo 'Companion object', es decir, similares a los métodos estáticos en Java:

    - mostrarMensaje(): Muestra un 'AlertDialog' en pantalla con el mensaje que hayamos pasado por parámetro y un botón de 'Aceptar'.

    - dialogNoCans(): Muestra un 'AlertDialog' con un titulo y un mensaje que indican que debemos añadir un numero CAN en el menú de inicio. Se mostrará si el usuario va a firmar con el método de firma del DNI electrónico y no ha añadido un código CAN anteriormente. También contiene un botón de 'Aceptar' para descartar el dialogo.

  - ModeloDatos.kt: Contiene una clase de tipo 'data class', es decir, una clase que solo guarda atributos a modo de modelo de datos. En este proyecto, se utiliza a la hora de extraer los datos guardados en el 'DataStore' de la aplicación, ya que necesita que le indiquemos que dato queremos extraer exactamente. Únicamente contiene una variable de tipo String denominada 'path'.

  - Utils.kt: Contiene la clase 'Utils' que contiene diferentes métodos de tipo 'companion object' que proporcionan diferentes utilidades al proyecto:

- isPasswordProtected(): Comprueba si un documento PDF pasado por parámetro está protegido o no con contraseña.

- isPasswordValid(): Recoge por parámetros una contraseña y un documento PDF. Comprueba en dicho documento si la contraseña indicada es correcta.

- NFCExists(): Comprueba si el dispositivo móvil en el que se está ejecutando la aplicación tiene un lector NFC incorporado.

- NFCActivated(): Comprueba si el lector NFC del dispositivo esta activado o desactivado.

- signDocuments(): Método que aplica la firma al documento (o documentos) seleccionado(s) con la clave privada del método de firma seleccionado.

■ Clases 'DNIeDROID': Se incluyen los paquetes 'graphic' con el archivo 'CanvasView.java' y 'pki' con los archivos 'OCSP.java' y 'Tool.java' y el archivo 'Common.java' que contienen distintos métodos para el funcionamiento de la firma con DNI electrónico. Han sido directamente extraídos e importados del kit de desarrollo de DNIeDROID, aunque se ha modificado un método del archivo 'Common' para obligar a que, cuando leamos el DNI, recoja directamente el certificado de firma en vez de preguntar por ese o por el de autenticación.

- Activitys: Se encuentran directamente en el paquete 'com.rasamadev.varsign'. Contienen los archivos de las diferentes pantallas de la aplicación y el control de lo que sucede en cada una de ellas:

  - MainActivity.kt: Es la pantalla que se muestra por defecto al iniciar la aplicación y desde la que se accede a todas las funcionalidades de esta.

    Sus atributos son:

    - btnInstalarCertificado: Boton de instalar certificado de la pantalla.
    - btnFirmarDocs: Boton de firmar documentos de la pantalla.
    - btnDocsFirmados: Boton del histórico de documentos de la pantalla.
    - btnAddCan: Boton de la pantalla para añadir un numero CAN.
    - biometricPrompt, executor, promptInfo: Atributos necesarios para la autenticación biométrica.
    - inicioApp: Variable para comprobar si es la primera vez que se ejecuta la aplicación.
    - PICK_PFX_REQUEST_CODE: Código de confirmación de solicitud de un archivo '.pfx'.
    - PICK_DIRECTORY: Código de confirmación de solicitud de un directorio.

- REQUEST_CODE_PERMISSIONS: Código de confirmación de permisos de almacenamiento.

- sdh: Siglas de 'Signed Docs Historic', recoge el historial de documentos firmados del 'DataStore'.

- _canStore: Instancia a la base de datos de CAN´s.

Sus métodos son:

- onCreate(): Método que se ejecuta al iniciar la pantalla. Inicia los elementos de la pantalla, comprueba los permisos y configuración de métodos de bloqueo de seguridad, controla la acción del botón de atrás del dispositivo, muestra un dialogo si hemos accedido a ella después de firmar documentos, recoge el historial de documentos firmados desde 'DataStore' e inicializa la base de datos de CANs.

- onResume(): Se ejecuta cuando el estado de la aplicación pasa a "Resumed" (por ejemplo, cuando se navega a otra aplicación distinta)

- onClick(): Controla las acciones de los botones de la pantalla al pulsarlos.

- onActivityResult(): Controla el resultado de elegir un archivo concreto cuando el dispositivo lo solicite.

- onRequestPermissionsResult(): Controla la respuesta a la solicitud del permiso "WRITE_EXTERNAL_STORAGE".

- initView(): Inicializa los elementos de la pantalla y los listener de los botones.

- getPaths(): Devuelve un string del histórico de documentos firmados por la aplicación.

- createFolderVarSign: Crea la carpeta 'VarSign' en el almacenamiento del dispositivo (si existe, no la sobrescribe).

- checkSecurityAndPermissionsConfig(): Comprueba primero si el dispositivo tiene configurado un método de bloqueo de seguridad para después comprobar si la aplicación tiene permisos de almacenamiento.

- checkPermissions(): Comprueba si la aplicación tiene permisos de almacenamiento.

- requestBiometricAuthentication(): Muestra el dialogo de solicitud de autenticación biométrica y controla si esta se ha realizado o no con éxito.

- openPfxPicker(): Abre el explorador de archivos para seleccionar un archivo de tipo '.pfx'.

- getPdfFileNamesInDirectory(): Recoge un objeto de tipo 'Uri' que contendrá un directorio con archivos de tipo PDF, de tamaño mayor de 0 bytes y que no sean erróneos.

- getDirectoryPath(): Devuelve la ruta relativa de un directorio seleccionado.

- openVariousDocListActivity: Abre la Activity de firma de varios documentos.

- installPfxCertificate(): Muestra los diálogos de instalación de certificado digital.

- dialogDocOptions(): Dialogo de menú para seleccionar firmar uno o varios documentos.

- dialogSecurityConfig(): Dialogo que se mostrará si el dispositivo no tiene un método de autenticación configurado.

- dialogPermissionsRequest(): Dialogo que solicitara los permisos de almacenamiento para la aplicación.

- dialogExitApplication(): Dialogo que nos preguntara si queremos salir de la aplicación.

- dialogAddCan(): Dialogo de menú para añadir un código CAN.

- dialogDudaCert(): Muestra un dialogo de ayuda al usuario para instalar un certificado digital.

- dialogDudaCAN(): Muestra un dialogo de ayuda al usuario para añadir un código CAN.

■ SingleDocActivity: Pantalla que contiene un formulario para seleccionar un documento e indicarle en que página se añadirá la firma y la posición de esta.

Sus atributos son:

- btnSelDoc: Boton para seleccionar un documento.

- btnFirmar: Boton "Firmar documento(s)".

- etNombreArchivo: Campo de nombre del documento seleccionado.

- etNumPagDoc: Campo de numero de página de la firma.
- txtNumPagsDoc: Texto que indica el número de páginas del documento.
- txtSignPage: Texto que indica si se ha seleccionado o no un documento.
- rgLugarFirma: RadioGroup de opciones del lugar de la firma.
- Grupo de RadioButtons, cada uno para una posición de la firma.
- rec: Rectángulo de la firma.
- idLugarFirma: Lugar de firma seleccionado.
- docSeleccionado: Documento seleccionado.
- docName: Nombre del documento seleccionado.
- docPath: Ruta del documento seleccionado.
- docPages: Número de páginas del documento seleccionado.
- numPageSign: Numero de la página donde se insertará la firma.
- signPosition: String que guardará la posición de la firma.
- aliasCert: Alias del certificado seleccionado.

- pdfReader: Lector del documento seleccionado.
- passwordDoc: Posible contraseña del documento a firmar.
- docPasswordExists: Variable booleana que guarda si el documento seleccionado tiene contraseña o no.
- _canStore: Instancia a la base de datos de CAN´s.
- CANSpecDO: Numero CAN seleccionado.
- getPdf: Recoge el documento PDF seleccionado mediante el explorador de archivos.

Sus métodos son:

- onCreate(): Inicializa la pantalla y recoge los datos de la base de datos de CANs.
- initView(): Inicializa los elementos de la pantalla y los listener de los botones.
- onClick(): Controla las acciones de los botones de la pantalla al pulsarlos.
- clearElements(): Devuelve los elementos dinámicos de la pantalla al estado de cuando se inició la pantalla.
- getFileName(): Recoge el nombre de un archivo.
- getFilePath(): Recoge la ruta de un archivo.
- continueDocSelected(): Continua con el proceso de firma. Carga el documento seleccionado, guarda

algunos de sus datos y modifica elementos de la pantalla.

- continueSign(). Continua con el proceso de firmado. Recoge la posición de la firma y comprueba la orientación de la página en la que se va a firmar.

- guardarPath(): Guarda en 'DataStore' la información del documento firmado.

- dialogRequestPassword(): Dialogo de menú de solicitud de contraseña de un documento.

- dialogDocumentAlreadyExists(): Dialogo de advertencia de que ya se encuentra una copia con el mismo nombre y firmada del documento seleccionado en la carpeta 'VarSign'.

- dialogSignMethods(): Dialogo de menú de selección de firma (certificado digital o DNI electrónico).

- dialogConfirmSign(): Dialogo de solicitud de confirmación para firmar el documento.

- dialogDocSigned(): Dialogo de confirmación de la firma que muestra un menú con las opciones de abrir el documento, volver al menú de inicio o cerrar el dialogo.

- dialogNFCConfig(): Dialogo que se muestra cuando el dispositivo móvil tiene el lector NFC deshabilitado. Nos da un acceso directo a los ajustes de conectividad del dispositivo para habilitarlo.

- VariousDocListActivity: Pantalla que se muestra después de haber seleccionado una carpeta mediante el explorador de archivos. Proporciona una lista de los documentos que contiene dicha carpeta y nos permite seleccionar cuales de ellos queremos firmar.

  Sus atributos son:

  - toolBarVariousDocs: Toolbar superior de la pantalla.
  - cbDocsContainer: Contenedor de la lista de documentos.
  - btnCancelar: Boton "Cancelar". Al pulsarlo, vuelve al menu de inicio.
  - btnAceptar: Boton "Aceptar".
  - docsSelected: Lista de documentos seleccionados para firmar.
  - aliasCert: Alias del certificado seleccionado.
  - directorySelected: Nombre de la carpeta seleccionada.
  - signPosition: String que guarda la posición de la firma.
  - _canStore: Instancia a la base de datos de CAN´s.
  - can: Numero CAN seleccionado.

  Sus métodos son:

  - onCreate(): Recoge los extras que provienen de la pantalla 'MainActivity', cargamos un CheckBox con

un nombre en el contenedor de la lista por cada documento que contenga la carpeta, mostramos un dialogo de advertencia si se han encontrado documentos con contraseña, e instanciamos a la base de datos de CAN´s.

- initView(): Inicializa los elementos de la pantalla, los listener de los botones y establece la Toolbar.

- onClick(): Controla las acciones de los botones de la pantalla al pulsarlos.

- guardarPath(): Guarda en 'DataStore' la información del documento firmado.

- dialogDocumentsAlreadyExists(): Dialogo de advertencia que se muestra cuando se encuentran uno o varios documentos ya firmados y con el mismo nombre en la carpeta 'VarSign'.

- dialogSignPosition(): Dialogo de menú para establecer la posición de la firma.

- dialogNFCConfig(): Dialogo que se muestra cuando el dispositivo móvil tiene el lector NFC deshabilitado. Nos da un acceso directo a los ajustes de conectividad del dispositivo para habilitarlo.

- dialogConfirmSign(): Dialogo de solicitud de confirmación para firmar los documentos seleccionados. Se muestran en una lista.

■ DNISignActivity(): Muestra la pantalla de proceso de firma por DNI electrónico. Utiliza varios de los métodos de la librería 'DNIeDROID' para el proceso.

Sus atributos son:

- tv: TextView de información principal.
- tvInfo: TextView de información adicional.
- ivDni: ImageView de la foto de un DNI.
- btnCancelar: Boton 'Cancelar'.
- _executor y _handler: Ejecutor y controlador de tareas secundarias.
- privateKey: Clave privada.
- chain: Clave pública.
- rec: Rectángulo de la firma.
- aliasCert: Alias del certificado
- can: Código CAN a utilizar para la firma.
- docPath: Ruta del documento o carpeta seleccionada.
- docsSelected: Lista de documentos seleccionados para firmar.
- signPosition: String que guarda la posición de la firma.
- numPageSign: Numero de la pagina donde se insertará la firma.
- pdfReader: Lector del documento seleccionado.

- del: String que guardará el nombre del documento erroneo que borraremos si ha habido un problema en el proceso de firmado.

Sus métodos son:

- onCreate(): Recogemos los extras que provengan de otras activitys, inicializamos el executor y el handler, el dialogo de solicitud de contraseña del DNI, establecemos la imagen del DNI e inicializamos la lectura por NFC.

- initView(): Inicializa los elementos de la pantalla y los listener de los botones.

- updateInfo(): Método que actualiza los TextView de información y actualiza la foto del prototipo de DNI.

- onTagDiscovered(): Método que se ejecuta cuando el lector NFC detecta el DNI. Contiene todo el codigo de firmado del documento.

- signDocuments(): Método que aplica la firma a uno o varios documentos.

- guardarPath(): Guarda en 'DataStore' la información del documento firmado.

- doInBackground(): Método que realiza en segundo plano el proceso de firmado.

- SignedDocsHistoricActivity: Muestra la pantalla del histórico de documentos firmados por el usuario y controla cuando pulsamos en uno de ellos.

  Sus atributos son:

  - vrSignedDocsHistoric: RecyclerView que contendrá la lista de documentos firmados.
  - toolBarSignedDocs: Toolbar superior.
  - signedDocsHistoric: String del histórico de documentos firmados.
  - splitSDH: Lista ya spliteada de los distintos documentos.

  Sus métodos son:

  - onCreate(): Inicializa la pantalla, recoge como extra de la activity 'MainActivity' el string del histórico de documentos firmados, lo splitea y lo aplica al RecyclerView de la pantalla.
  - initView(): Inicializa los elementos de la pantalla y establece la toolbar.
  - onItemClick(): Método de configuración de la pulsación de un documento en la lista.

7.4.3 ARCHIVOS DE PANTALLAS Y RECURSOS

Todos estos recursos se almacenan dentro de la carpeta 'res' en el directorio 'app/src/main' del proyecto. Se dividen en varias carpetas:

- drawable: Contiene algunos archivos importados en el proyecto para su uso en algunas pantallas u otros apartados.

  En este proyecto, están incluidas las imágenes del prototipo del DNI en blanco y negro y a color, una imagen de un ejemplo del DNI para la ayuda del código CAN, el logo del DNIe, el logo de la aplicación y el icono del documento PDF que se muestra en cada registro de la lista de documentos firmados.

- layout: Contiene todos los archivos de diseño de las pantallas y de algunos ítems para las mismas, todo en formato XML. Concretamente, los archivos son:

  - activity_dni_sign: Diseño de la pantalla de firma mediante DNI.

  - activity_main: Diseño de la pantalla de inicio de la aplicación.

  - activity_signed_docs_historic: Diseño de la pantalla del histórico de documentos firmados.

  - activity_singledoc: Diseño de la pantalla de firma de un documento.

  - activity_variousdoclist: Diseño de la pantalla de firma de varios documentos.

- can_list: Diseño del dialogo con la lista de CAN´s a seleccionar.
- dialog_duda_can: Diseño del dialogo que muestra la ayuda al usuario sobre el numero CAN.
- dialog_variousdoc_signposition: Diseño del menú de posición de la firma para la pantalla de firma de varios documentos.
- item_signed_docs_historic: Item de cada documento firmado para la lista del histórico.
- list_mrtd_row: Item para cada numero CAN con los datos del mismo y el botón de borrar.
- sample_can: Diseño del dialogo de menú para introducir un código CAN y guardarlo.

- values: En esta carpeta se guardan variables de datos que se usan en algunas pantallas, como strings de frases concretas, paletas de colores en formato hexadecimal, o estilos y temas para diferentes elementos.

8. PRUEBAS

A continuación, se ofrece una tabla de casos de prueba realizados en la aplicación con sus resultados, indicando en color verde las pruebas realizadas con éxito y en rojo las pruebas fallidas y los motivos:

DESCRIPCION DE LA PRUEBA	CONDICIONES PREVIAS	DATOS DE PRUEBA	PASOS A EJECUTAR	RESULTADO FINAL
Instalar un certificado digital.	Disponer del archivo '.pfx' de un certificado	Archivo '.pfx' de un certificado	1- Menú de inicio > Instalar certificado 2- Seleccionar archivo '.pfx' en el explorador de archivos 3- Escribir la contraseña del certificado 4- Cumplimentar el resto de los pasos	El certificado se instala correctamente.
Añadir un código CAN.	Disponer de lector NFC en nuestro dispositivo móvil.	Código CAN de 6 dígitos.	1- Menú de inicio > Añadir CAN 2- Introducir el código CAN de 6 dígitos y aceptar.	El código CAN se añade correctamente.
Añadir un código CAN.	Disponer de lector NFC en nuestro dispositivo móvil.	Código CAN de 3 dígitos.	1- Menú de inicio > Añadir CAN 2- Introducir el código CAN de 3 dígitos y aceptar.	Se nos muestra un dialogo de error indicando que el CAN debe tener 6 dígitos numéricos.
Ver el histórico de documentos firmados.	Haber firmado al menos un documento con la aplicación.	Ninguno.	1- Instalamos la aplicación y la ejecutamos por primera vez 2- Menú de inicio > Documentos firmados	Se nos muestra un dialogo indicando que debemos firmar al menos un documento.
Ver el histórico de documentos firmados.	Haber firmado al menos un	Ninguno.	1- Firmamos un documento con la aplicación.	Se nos muestra el historial de

	documento con la aplicación.		2- Menú de inicio > Documentos firmados	documentos firmados.
Firmar un documento.	Seleccionar un documento PDF desde el explorador de archivos e indicar la página y posición de la firma.	Ninguno.	1- Menú de inicio > Firmar documentos > Autentificación biométrica > Un documento. 2- Pulsamos en "Firmar documento".	Se nos muestra un mensaje de error indicando que no hemos seleccionado un documento.
Firmar un documento.	Seleccionar un documento PDF desde el explorador de archivos e indicar la página y posición de la firma.	Documento PDF.	1- Menú de inicio > Firmar documentos > Autenticación biométrica > Un documento. 2- Seleccionamos un documento. 3- Indicamos un numero de página superior al número total de páginas del documento. 4- Pulsamos en "Firmar documento".	Se nos muestra un mensaje de error indicando que establezcamos un numero de página correcto.
Firmar un documento.	Seleccionar un documento PDF desde el explorador de archivos e indicar la página y posición de la firma.	Documento PDF.	1- Menú de inicio > Firmar documentos > Autenticación biométrica > Un documento. 2- Seleccionamos un documento. 3- Indicamos que se firme en la página 1 del documento. 4- Pulsamos en "Firmar documento".	Continuamos con el proceso de firmado correctamente.
Firmar un documento.	Seleccionar un documento PDF desde el explorador de archivos e indicar la página y posición de la firma.	Documento PDF erróneo.	1- Menú de inicio > Firmar documentos > Autenticación biométrica > Un documento.	Se nos muestra un mensaje de error indicando que el documento no se puede abrir

			2- Seleccionamos un documento erróneo.	
Firmar varios documentos.	**Seleccionar una carpeta que contenga documentos PDF e indicar los que queramos firmar.**	**Carpeta del dispositivo.**	**1- Menú de inicio > Firmar documentos > Autenticación biométrica > Varios documentos.** **2- Seleccionamos una carpeta vacía.**	**Se nos mostrará un mensaje de error indicando que la carpeta no contiene documentos.**
Firmar varios documentos.	**Seleccionar una carpeta que contenga documentos PDF e indicar los que queramos firmar.**	**Carpeta del dispositivo.**	**1- Menú de inicio > Firmar documentos > Autenticación biométrica > Varios documentos.** **2- Seleccionamos una carpeta con documentos PDF.**	**Se nos llevara a una pantalla que muestra la lista de documentos que contiene la carpeta, pudiendo seleccionar cuales queremos firmar.**
Firmar varios documentos.	**Seleccionar una carpeta que contenga documentos PDF e indicar los que queramos firmar.**	**Carpeta del dispositivo.**	**1- Menú de inicio > Firmar documentos > Autenticación biométrica > Varios documentos.** **2- Seleccionamos una carpeta con documentos PDF.** **3- Deseleccionamos todos los documentos de la lista.** **4- Pulsamos en "Aceptar".**	**Se nos mostrará un mensaje de error indicando que seleccionemos al menos un documento.**
Firmar varios documentos.	**Seleccionar una carpeta que contenga documentos PDF e indicar los que queramos firmar.**	**Carpeta del dispositivo.**	**1- Menú de inicio > Firmar documentos > Autenticación biométrica > Varios documentos.** **2- Seleccionamos una carpeta con documentos PDF.**	**Continuaremos con el proceso de firma correctamente.**

			3- Seleccionamos algunos documentos de la lista. **4- Pulsamos en "Aceptar".**	
Firmar con DNI electrónico.	**Haber añadido un código CAN en el menú de inicio. También tener el lector NFC activado.**	**Ninguno.**	**1- Instalamos la aplicación.** **2- Menú de inicio > Firmar documento > Autentificación biométrica > Cualquiera de las dos opciones** **2- Seleccionamos el documento o documentos a firmar.** **3- Indicamos el método de firma por DNI electrónico.**	**Se nos mostrará un mensaje de error indicando que debemos añadir un código CAN en el menú de inicio.**
Firmar con DNI electrónico.	**Haber añadido un código CAN en el menú de inicio. También tener el lector NFC activado.**	**Código CAN de nuestro DNI.** **Lector de NFC activado.**	**1- Instalamos la aplicación.** **2- Añadimos un código CAN en menú de inicio > Añadir CAN.** **3- Menú de inicio > Firmar documento > Autentificación biométrica > Cualquiera de las dos opciones** **4- Seleccionamos el documento o documentos a firmar.** **5- Indicamos el método de firma por DNI electrónico.**	**Se nos solicitara que seleccionemos un numero CAN de la lista.**
Firmar con DNI electrónico.	**Haber añadido un código CAN en el menú de inicio. También tener el lector NFC activado.**	**Código CAN de nuestro DNI.** **Lector de NFC desactivado.**	**1- Menú de inicio > Firmar documento > Autentificación biométrica >**	**Se nos mostrara un mensaje indicando que debemos activar el lector DNI.**

			Cualquiera de las dos opciones **2- Seleccionamos el documento o documentos a firmar.** **3- Indicamos el método de firma por DNI electrónico.**	
Firmar con DNI electrónico.	**Haber añadido un código CAN en el menú de inicio. También tener el lector NFC activado.**	**Código CAN de nuestro DNI.** **Lector de NFC desactivado.**	**1- Menú de inicio > Firmar documento > Autentificación biométrica > Cualquiera de las dos opciones** **2- Seleccionamos el documento o documentos a firmar.** **3- Indicamos el método de firma por DNI electrónico.**	**Se nos mostrara un mensaje indicando que debemos activar el lector DNI.**
Firmar con DNI electrónico.	**Haber añadido un código CAN en el menú de inicio. También tener el lector NFC activado.**	**Código CAN de nuestro DNI.** **Lector de NFC activado.**	**1- Menú de inicio > Firmar documento > Autentificación biométrica > Cualquiera de las dos opciones** **2- Seleccionamos el documento o documentos a firmar.** **3- Indicamos el método de firma por DNI electrónico.** **4- En la pantalla de firma por DNI, aproximamos el DNI al lector, pero al rato lo apartamos.**	**Se nos mostrara un mensaje de error indicando que se ha perdido la conexión con el DNIe.**

Firmar con DNI electrónico.	**Haber añadido un código CAN en el menú de inicio. También tener el lector NFC activado.**	**Código CAN de nuestro DNI.** **Lector de NFC activado.** **Contraseña incorrecta.**	**1- Menú de inicio > Firmar documento > Autentificación biométrica > Cualquiera de las dos opciones** **2- Seleccionamos el documento o documentos a firmar.** **3- Indicamos el método de firma por DNI electrónico.** **4- En la pantalla de firma por DNI, aproximamos el DNI al lector.** **5- En el momento de introducir la contraseña, introducimos una que es incorrecta.**	**Se mostrará de nuevo el dialogo de solicitud de contraseña, pero indicando el número de intentos restantes.**
Firmar con DNI electrónico.	**Haber añadido un código CAN en el menú de inicio. También tener el lector NFC activado.**	**Código CAN de nuestro DNI.** **Lector de NFC activado.** **Contraseña incorrecta.**	**1- Menú de inicio > Firmar documento > Autentificación biométrica > Cualquiera de las dos opciones** **2- Seleccionamos el documento o documentos a firmar.** **3- Indicamos el método de firma por DNI electrónico.** **4- En la pantalla de firma por DNI,**	**La firma del documento se realiza correctamente.**

			aproximamos el DNI al lector. **5- En el momento de introducir la contraseña, introducimos la contraseña y confirmamos la firma.**	

9. CONCLUSIONES Y POSIBLES AMPLIACIONES

En este proyecto hemos visto como realizar una aplicación para Android utilizando el lenguaje Kotlin, un lenguaje bastante demandado hoy en día para este tipo de desarrollo, hemos investigado acerca de la firma digital y la criptografía a través de teoría y distintas librerías desarrolladas por terceras personas.

Las posibles ampliaciones que añadiría en un futuro son:

- Mejora de la interfaz de usuario: Teniendo en cuenta la evolución del lenguaje de diseño 'Material' de Google y la implementación del kit de herramientas 'Jetpack Compose' de Android en sustitución a los archivos XML de pantallas e ítems, se podría realizar una interfaz más visual e intuitiva para el usuario.

- Mejora en la lectura del DNI electrónico: Debido a las limitaciones de la librería 'DNIeDROID', el proceso de firma con el DNI puede ser un poco complicado debido a la lentitud a la hora de leer los datos de este y el tener que escribir la contraseña sin apartar el DNI del dispositivo. Pero gracias a que se trata de un kit de desarrollo de

código abierto, se puede investigar su API y sus librerías para comprender mejor su funcionamiento y modificar el código de forma que facilite el proceso de firma al usuario.

- Lectura de documentos protegidos con certificado digital: Dado que existe la posibilidad de cifrar documentos PDF de modo que solo puedan ser abiertos a través de un certificado, la posibilidad de que la aplicación pudiera desbloquearlos utilizando un certificado instalado en el dispositivo haría a la aplicación mucho mas funcional.

10. BIBLIOGRAFIA

Cómo administrar todos los archivos de un dispositivo de almacenamiento. (s. f.). Android Developers. Enlace.

Descripción general del almacenamiento de archivos y datos. (s. f.). Android Developers. Enlace.

Enviroment. (s.f). Android Developers. Enlace.

Show a biometric authentication dialog. (s. f.). Android Developers. Enlace.

Wikipedia contributors. (2024, 26 marzo). Bouncy Castle (cryptography). Wikipedia. Enlace.

Spongy Castle by rtyley. (s. f.). Enlace.

Arquitectura de apps: Capa de datos - DataStore. (s. f.). Android Developers. Enlace.

Cómo usar las corrutinas de Kotlin con componentes optimizados para ciclos de vida. (s. f.). Android Developers. Enlace.

Executor. (s. f.). Android Developers. Enlace.

Handler. (s. f.). Android Developers. Enlace.

IText 5.5.10 API. (s. f.). Enlace.

Checking if pdf is password protected using itextsharp. (s. f.). Stack Overflow. Enlace.

Installing iText G for Android. (s. f.). Enlace.

Getting the orientation of a PDF that has been read in through iText. (s. f.). Stack Overflow. Enlace.

KeyChain. (s. f.). Android Developers. Enlace.

Check programmatically if device has NFC reader. (s. f.). Stack Overflow. Enlace.

Android.NFC. (s. f.). Android Developers. Enlace.

Programmatically change input type of the EditText from PASSWORD to NORMAL & vice versa. (s. f.). Stack Overflow. Enlace.

Conditions and loops | Kotlin. (s. f.). Kotlin Help. Enlace.

File. (s. f.). Android Developers. Enlace.

Manifest.permission. (s. f.). Android Developers. Enlace.

¿Cómo agregar y utilizar librerías .aar en Android Studio? (s. f.). Stack Overflow . Enlace.

Cómo crear una biblioteca de Android. (s. f.). Android Developers. Enlace.

ActivityResultContracts. (s. f.). Android Developers. Enlace.

Agentes de resolución de contenido. (s. f.). Android Developers. Enlace.

AlertDialog. (s. f.). Android Developers. Enlace.

Belinski, E. (s. f.). *Android API levels.* Enlace.

Casos prácticos y prácticas recomendadas de almacenamiento en Android. (s. f.). Android Developers. Enlace.

Ciclo de vida de la actividad. (s. f.). Android Developers. Enlace.

Cómo brindar navegación de retroceso personalizada. (s. f.). Android Developers. Enlace.

Criptografía. (s. f.). Android Developers. Enlace.

Data classes | Kotlin. (s. f.). Kotlin Help. Enlace.

Design Android EditText to show error message as described by google. (s. f.). Stack Overflow. Enlace.

FileProvider. (s. f.). Android Developers. Enlace.

Settings. (s. f.). Android Developers. Enlace.

Intent. (s. f.). Android Developers. Enlace.

KeyguardManager. (s. f.). Android Developers. Enlace.

Que son los certificados electrónicos. (s. f.). DNIElectronico.es. Enlace.

1553 - Diferencias entre .pfx .p12 .cer y .crt. - Sede. (s. f.). Enlace.

Código fuente de las aplicaciones. (s. f.). Enlace.