Функциональные характеристики и информация необходимая для установки, эксплуатации модуля «Безбумажный офис - SDK»

«Безбумажный офис - SDK» — комплект программных средств и средств разработки (библиотек), позволяющий интегрировать функционал Программы в мобильные приложения для операционных систем Android и iOS.

Ограничение использования SDK

SDK встраивается в нативные приложения под управлением операционных систем: Android, iOS.

1. Регистрация и авторизация

ИСЛ (здесь и далее под этой аббревиатурой понимается Информационная система лицензиата) отображает иконку приложения, через которое производится подписание документов: «Безбумажный офис». При нажатии на кнопку ИСЛ передаёт в Модуль уникальный идентификатор клиента. Модуль генерирует одноразовый код для авторизации. На основе одноразового кода ИСЛ генерирует ссылку для входа. Клиент переходит по ссылке и авторизуется в Модуле.

Если клиент не зарегистрирован, ИСЛ отправляет запрос на регистрацию клиента в Модуль. Модуль регистрирует клиента и отправляет в ИСЛ уникальный идентификатор, по которому можно получить код авторизации.

2. Выпуск электронной подписи

Модуль проверяет наличие ключей электронной подписи на устройстве клиента при каждом входе. Если ключи не найдены, Модуль генерирует новые. Во время генерации ключей Модуль отображает информационное сообщение.

После успешной генерации ключей Модуль апредлагает клиенту подписать акт признания ключа*. Клиент подписывает акт кнопкой. Модуль фиксирует событие в доказательной базе.

После подписания акта Безбумажный офис открывает клиенту доступ к подписанию документов.

*Есть возможность генерировать акт признания ключа на стороне оператора систмеы. Процесс описан в описании API.

3. Подпись документов

Клиент выбирает интересный ему продукт в ИСЛ и нажимает на кнопку «Оформить». При нажатии на кнопку ИСЛ отправляет в Модуль комплект документов для подписания. Модуль отправляет документы на подпись клиенту и открывает страницу SDK для просмотра файлов. Клиент просматривает файлы и нажимает кнопку «Подписать». Модуль накладывает подпись на документы. После подписания приложение отображает клиенту подписанные документы. Если клиент закрыл приложение, он может в любой момент вернуться к подписанию: на главной странице ИСЛ отображается иконка «Безбумажный офис» и количество документов на подпись. При нажатии на иконку клиент авторизуется и

попадает на страницу со списком документов. На странице можно выбрать любой документ для подписания или просмотра.

4. Функциональное описание

№	Функциональная возможность	Описание
1. Авторизация		
1.1.	Авторизоваться в SDK через	Логика работы: 1. Пользователь нажимает на кнопку «Безбумажный офис» внутри ИСЛ; 2. ИСЛ отправляет в Модуль номер телефона и ФИО пользователя; 3. Модуль авторизует пользователя в системе. Подробная схема авторизации представлена в описании API.
2. Подписание акта признания ключа		
2.1.	Проверять, что на устройстве пользователя установлены ключи	Модуль проверяет наличие ключей на устройстве при каждом входе.
2.2.	Создавать новые ключи подписи, если на устройстве нет ключей	Если на устройстве пользователя не установлены ключи, Модуль создаёт новые.
2.3.	Генерировать ключи на устройстве	Модуль генерирует ключи на устройстве после создания сертификата.
2.4.	Сохранить событие подписания акта признания ключа в доказательную базу	Артефакты для сохранения: 1. Пользователь; 2. Название события; 3. Время события; 4. Отпечаток устройства пользователя; 5. Геопозиция;

		6. Акт признания ключа (напр., в base64).
3. Список заявок		
3.1.	Отобразить список заявок	Приложение отображает список заявок в табличном представлении. Данные в таблице: 1. Название заявки; 2. Статус; 3. Дата создания.
4. Работа с заявкой на подпись		
4.1.	Отобразить документы на подпись	Клиент видит форму заявки, на которой отображается превью документов, требующих подписи. Если документов несколько, то они доступны для пролистывания. Формат документов: PDF. Под документами отображаются две кнопки "Подписать" и "Отклонить".
4.2.	Отобразить страницу успеха после подписи заявки	После успешного подписания приложение отображает подписанные документы. Рядом с каждым документом отображается статус подписи: "Подписан".
4.3.	Подписать документы	Приложение накладывает электронную подпись клиента на документы при нажатии на кнопку "Подписать".
4.4.	Отклонить подпись	Приложение даёт возможность отклонить заявку. После того как Клиент отклонил заявку приложение перенаправляет его на главную страницу.
4.5.	Отобразить информацию о подписи	Приложение отображает рядом с каждым документом ссылку "Информация о подписи". При нажатии на ссылку можно посмотреть информацию о владельце сертификата и времени подписи.

4.6.	Отобразить штамп о подписи на документах	Приложение накладывает штамп о подписи на каждый документ.
4.7.	Отправить документы на почту клиента	Приложение отправляет подписанные документы на почту клиента.
4.8.	Отобразить Клиенту информацию: документ подписан Оператором	На странице заявки приложение отображает Клиенту информацию о том, что документ был подписан Оператором.

5. Примеры работы:

Сценарий 1. Встройка SDK в приложение системы. Система (Рис.1)

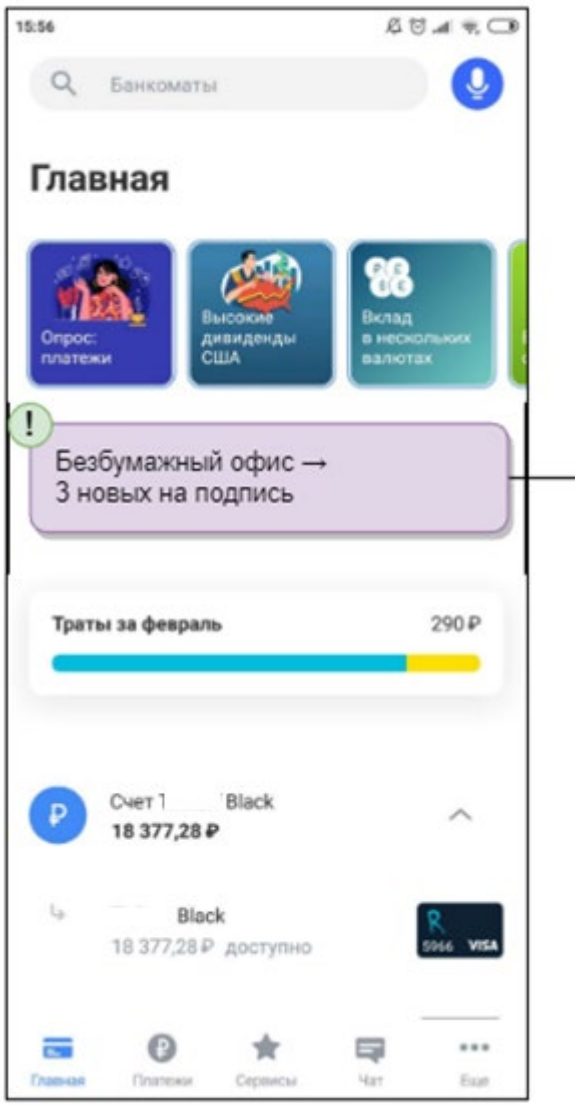


Рисунок 1

SDK открывается поверх Системы. Можно вернуться «Назад» и sdk закроется. (Рис.2)

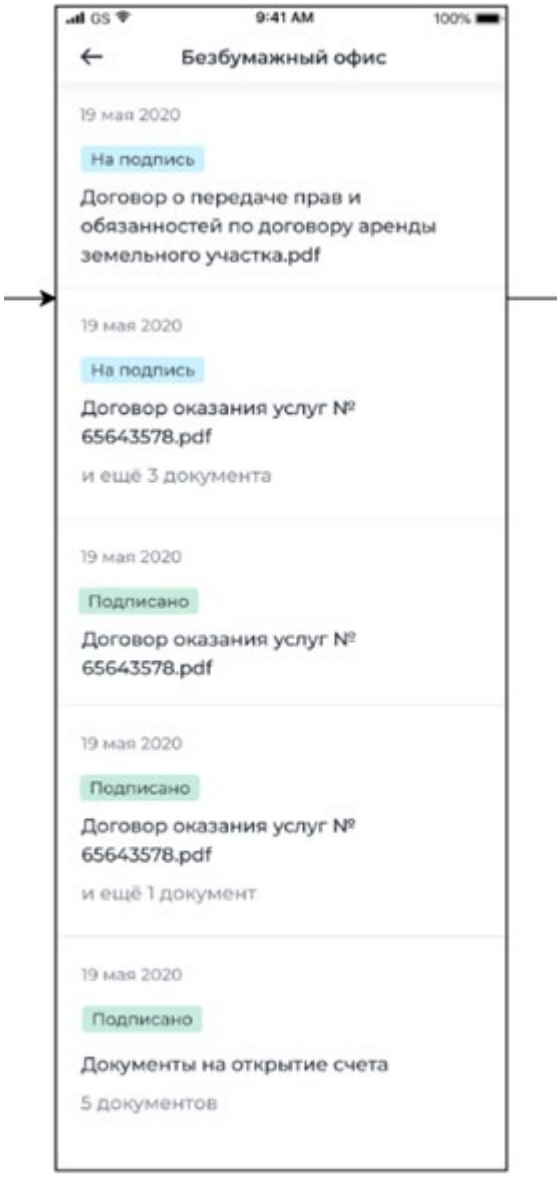


Рисунок 2

Далее, стандартная работа приложения ЛКК ФЛ: навигация, функционал внутри приложения. (Рис.3)


GS
9:41 AM
100%
Заявление на отпус...
и еще 3 документа
На подпись
Договор оказания услуг № 1.pdf
ДОГОВОР №
1. ПРЕДМЕТ ДОГОВОРА
2. ПРАВА И ОБЯЗАННОСТИ СТОРОН
3. ИЗМЕНЕНИЕ И ПРЕКРАЩЕНИЕ НАСТОЯЩЕГО ДОГОВОРА
Подписать все
Отклонить

Рисунок 3

Стандартная работа приложения. (Рис.4)

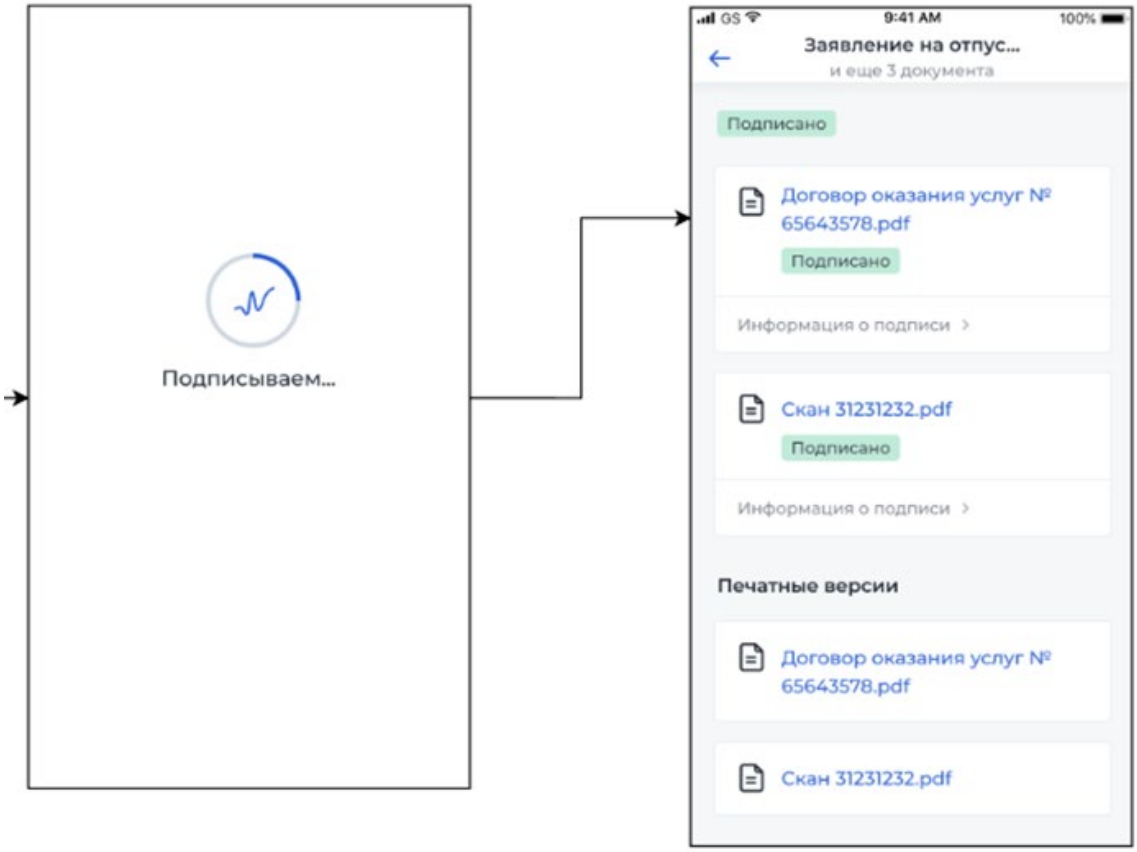


Рисунок 4

Сценарий 2. Встройка SDK в систему.

Система. (Рис.5)

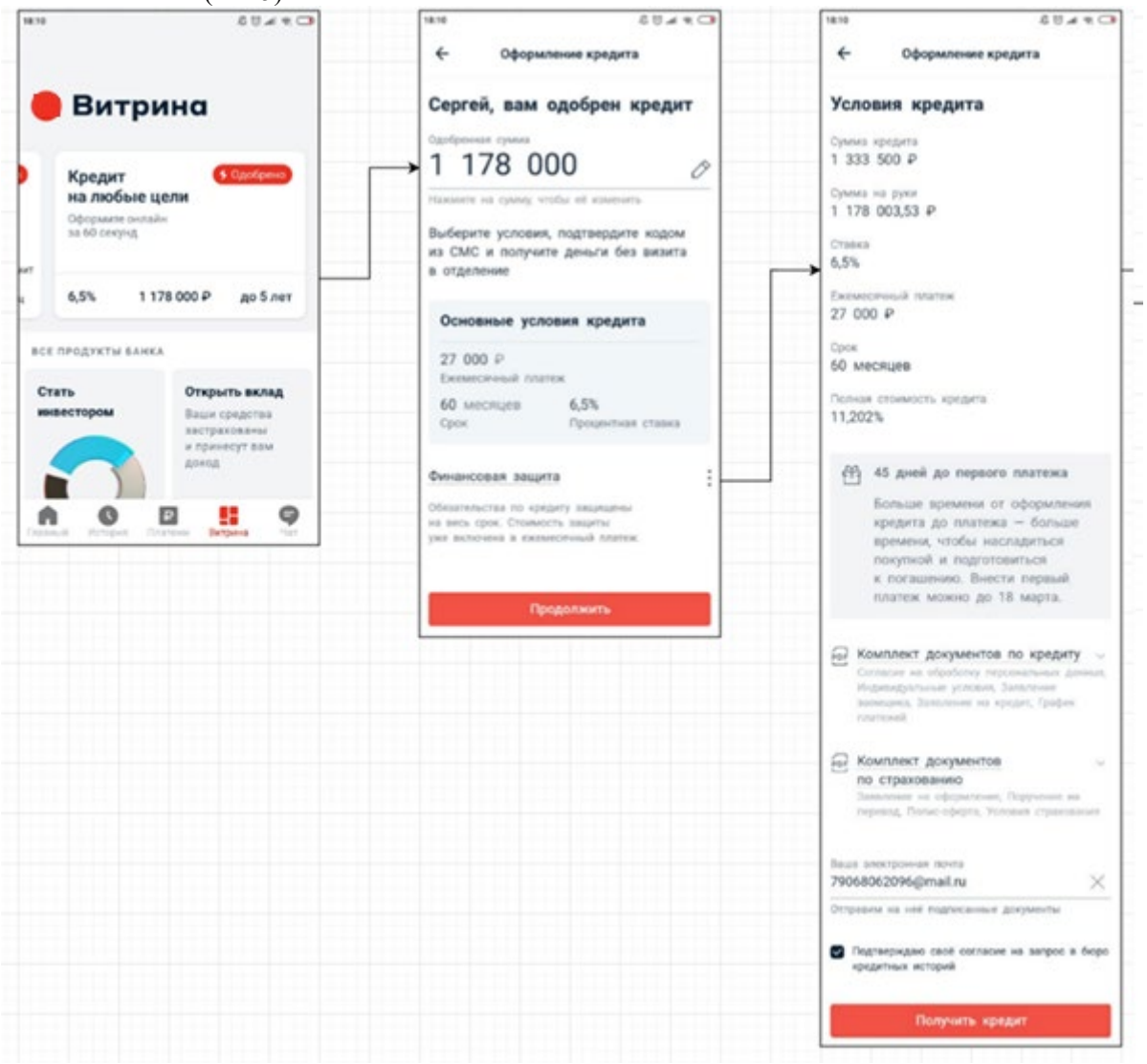


Рисунок 5

Система создаёт заявку на подпись в NPP, прикрепляет файлы к заявке, отправляет на подпись клиенту; открывает страницу SDK для просмотра файлов и подписи. И SDK открывается поверх системы. Можно вернуться «Назад» — NPP закроется. (Рис.6)

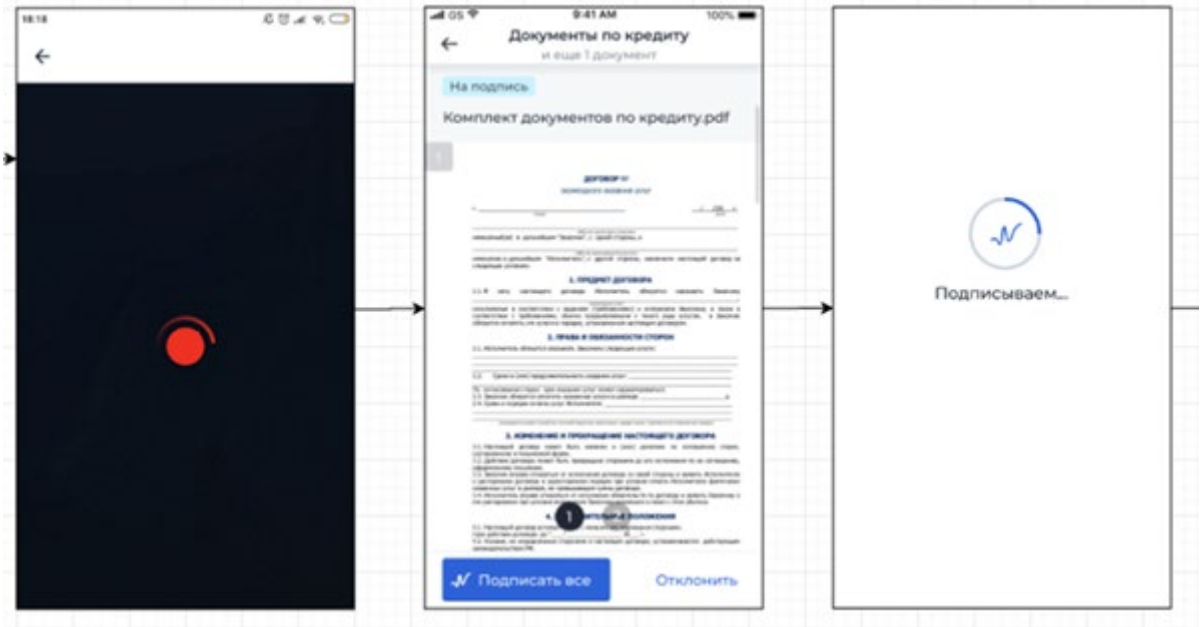


Рисунок 6

СИСТЕМА забирает подписанные файлы из NPP, отображает клиенту в приложении. Далее, система отображает клиенту историю документооборота. (Рис.7)

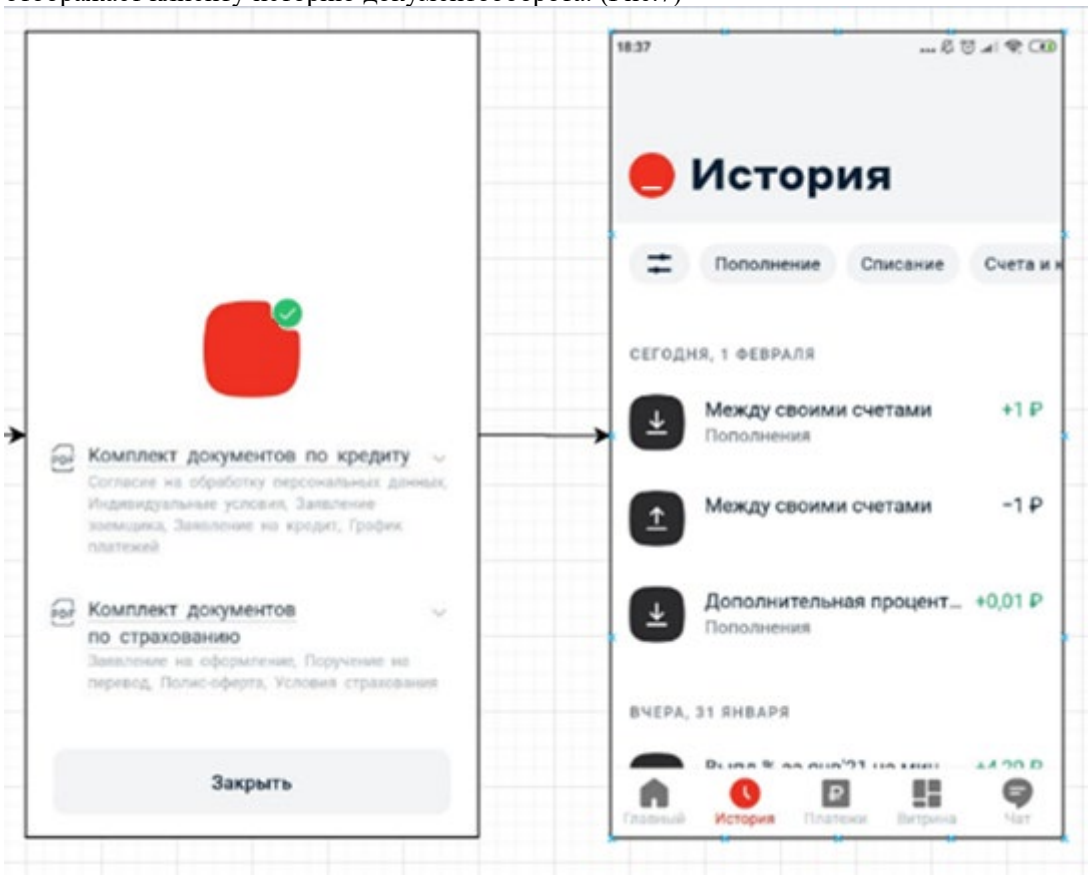
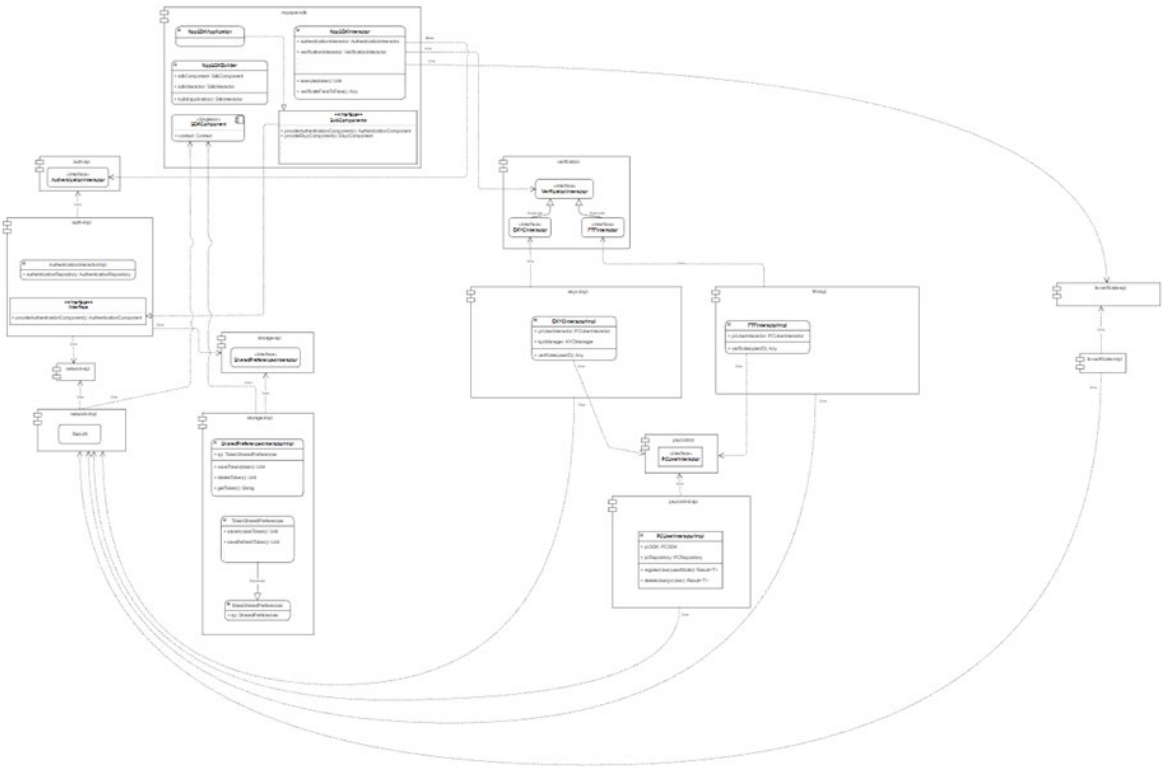


Рисунок 7

6. Архитектура SDK

Общая архитектура на диаграмме (список методов только для примера, реальная реализация будет отличаться) (Рис.8):



2.1. Состав:

1. nopaper-sdk – главный модуль SDK
2. манифест – xml-файл с объявленными android-компонентами
3. dagger – библиотека DI

4. dagger-компонент и dagger-модуль – классы, предоставляющие зависимости в компонент SDK (Не путать понятие dagger-модуля и компонент SDK. Компонент SDK - отдельная часть всего приложения. Dagger-модуль содержит методы типа

```
provideNppSDKInteractor(nppSDKInteractorImpl: NppSDKInteractorImpl): NppSDKInteractor
```

для получения нужных зависимостей)

Общий сценарий SDK с ui такой (Activity в данном примере принадлежит родительскому приложению):
Application.onCreate() → NppSDKInitializer.create() → NppSDKBuilder.build() → Activity.onStart() → Activity.onResume() → NppSDKActivity.onCreate() → nav_graph_main → ...
Без ui такой:
Application.onCreate() → NppSDKInitializer.create() → NppSDKBuilder.build() → Activity.onStart() → Activity.onResume() → NppSDKInteractor.auth() → какое-то отображение → NppSDKInteractor.verificate() → какое-то отображение → NppSDKInteractor.getDocuments() → ...
Подробное описание сценария:
В манифесте nopaper-sdk объявлен класс NppSDKInitializer(). У него есть метод:

```
override fun create(context: Context): Boolean {
        NppSDKBuilder.build(context)
        return true

}
```

Он вызывается при каждом старте родительского приложения и в нем вызывается метод build у класса NppSDKBuilder.

```
abstract class NppSDKBuilder {

    companion object {

        @JvmStatic
        lateinit var sdkInteractor: NppSDKInteractor
            private set

        internal lateinit var component: SDKComponent

        @JvmStatic
        internal fun build(context: Context): NppSDKInteractor {
            if (!this::sdkInteractor.isInitialized) {

                component = DaggerSDKComponent
                    .factory()
                    .create(context)

                sdkInteractor = component.inject()
            }

            return sdkInteractor
        }
    }
}
```

Метод build сначала проверяет, не проинициализировано ли свойство sdkInteractor, и если нет, то получает имплементацию SdkInteractor через dagger-компонент и возвращает ее. Теперь родительское приложение может получить это свойство, так как оно статическое.
После инициализации SDK управление вернется родительскому Application, которое запускает отображение какой-нибудь Activity. Внутри этой активити, когда это нужно, можно создать интент на запуск стартовой активити SDK вот так:
Intent sdkIntent = new Intent(this, NppSDKActivity.class);

sdkIntent.putExtra("oneTimeLink", "https://onetimelink");

startActivity(sdkIntent);

На этом этапе проиведется запуск активити SDK, корневым элементом которой является граф навигации nav_graph_main. И там уже будет идти цепочка вызовов фрагментов из вложенных графов. Каждый модуль SDK, содержащий ui, будет иметь свой граф. Он и будет вложенным в nav graph main. Подробнее про библиотеку навигации можно почитать здесь:
https://developer.android.com/guide/navigation/navigation-getting-started

7. Список основных модулей:

Каждый модуль содержит следующие пакеты: di и domain, а также data и ui, если предполагается работа с сетью, БД и пользовательским интерфейсом, соответственно.

- :nopaper-sdk

Главный модуль, зависит от всех остальных модулей. Содержит abanking-settings.json, MainActivity, корневой граф, ссылающийся на остальные, SdkComponent и SdkModule. Интерактор верхнего уровня SdkInteractor. Методы этого интерактора доступны родительскому приложению. Чтобы получить SdkInteractor необходимо вызвать метод build(application: Application) у класса SdkBuilder. Он создаст компонент, от которого можно получать зависимости, и вернет SdkInteractor.
Предполагаемые методы SdkInteractor:

```kotlin
package ab.nopaper.nopapersdk.domain

import ab.nopaper.cert_api.domain.entities.Certificate
import ab.nopaper.docs_api.domain.entities.Document
import ab.nopaper.utils.SDKState
import ab.nopaper.utils.Task

interface NppSDKInteractor {

    val currentState: SDKState
        get() = SDKState

    fun auth(token: String, username: String): Task<Unit> // готово

    fun logout() // готово

    /** Проверка и установка ключа на устройство */
    fun verificate(): Task<Int> // готово для очной, в процессе для ekyc

    /**
     * Получение ссылки на Акт признания ключа
     *
     *  @param certificateId Идентификатор ключа.
     */
    fun getKeyAcceptanceAct(certificateId: Int): Task<String> // готово
    /**
     * Активация сертификата
     *
     *  @param certificateId Идентификатор ключа.
     */
    fun activateCertificate(certificateId: Int): Task<Unit> // готово

    fun getAllDocument(): Task<List<Document>> // готово

    fun getLastDocument()

    fun getFirstDocument()

    fun getAllCertificate(): Task<List<Certificate>> // готово

    fun getLastCertificate()

    fun signDocumentOneSide()

    fun signDocumentTwoSide()

    fun signDocumentManySide()

    fun signDocumentReverseSide()

    fun getCertificateInfo()

    fun getLastCertificateInfo()

    fun reissueCertificate()

    fun getAgreement()

    fun getProfileId(): Result<Int> // готово
}
```

Главный модуль предоставляет АПИ для обращения к внутренним модулям, управления и получения состояния встраимого приложения.
В данный момент есть разделение на две реализации этого интерфейса (для очной верификации и для ekyc) через buildFlavors. В зависимости от выбранной сборки будет использоваться нужная реализация.

- :auth:impl

Модуль, отвечающий за авторизацию. Зависит от модуля network-api

- :storage

Модуль, отвечающий за работу с локальным хранилищем. На диаграмме показан SharedPreferencesInteractor, с помощью которого сохраняются, удаляются и получаются access и refresh токены.

- :verification

Модуль с интерфейсами. Пока что это VerificationInteractor, EkycInteractor и FTFInteractor

Суть в том, что на внешнем слое мы будем просить у даггера объект VerificationInteractor. И в зависимости от buildVariant, с которым мы будем делать сборку, даггер предоставит нам либо EkycInteractorImpl, либо FTFInteractorImpl. А с помощью дженериков можем менять типы аргументов и возвращаемого значения. Для примера везде Any (аналог Object из Java)

- :ekyc-impl

Модуль, отвечающий за прохождение eKYC-верификации. Зависит от network-api и от PCSDK.

- :ftf-impl

Модуль, отвещающий за прохождение очной верификации. Зависит от network-api и от PCSDK.

- :paycontrol:impl

Модуль, отвещающий за создание и удаление пользователя на сервере PC, сохранение ключей в хранилище.

- :nopaper-lk:certificate:impl

Модуль для работы с сертификатами через lk-api

- :nopaper-lk:documents:impl

Модуль для работы с документами через lk-api

- :nopaper-lk:profile:impl

Модуль для работы с профилем через lk-api

- :network

Модуль, предоставляющий объект Retrofit, необходимый для общения с сервером.

- :utils

Утилитарный модуль
Все зависимости, необходимые только вложенным модулям, предоставляются там же, то есть в главном модуле мы получаем только интерфейсы-интеракторы и взаимодействуем через них.

8. Стек технологий

- Kotlin
- Gradle DSL (для настройки сборки)
- Android Jetpack Navigation (для навигации по экранам)
- Android Architecture Components (различные зависимости для удобной разработки android)
- Dagger (для внедрения зависимостей)
- Retrofit (для запросов на сервер)
- GSON (парсинга Json)
- Coroutines (для многопоточности)
- Timber (для логов)

9. Описание методов SDK

- Аутентификация identity-server-api

Текущий метод в сервисе	Описание нового запроса	Запрос	Необходимые работы
generateOneTimeLink	-	-	-
Auth	-	-	-

- Сертификаты signing-api

Текущий метод в сервисе	Описание нового запроса	Запрос	Необходимые работы
createCertificateFl	Создание сертификата	POST /api/v1/debug/certificate/create	Модель/рест
getAllCertificate	Получение списка возможных сертификатов	GET /api/v1/public/certificate/list	Модель/рест
updateStatus	Смена статуса сертификата	PATCH /api/v1/debug/certificate/{certificateId}/change-status	Рест
getKeyAcceptanceAct	-	-	Перенесен в pc-api
getAllCertificateForSigning	Получение списка возможных сертификатов	GET /api/v1/public/container/{containerId}/certificate/list	Модель/рест

- Документы document-api

Текущий метод в сервисе	Описание нового запроса	Запрос	Необходимые работы
getAllDocuments	Получение списка документов	GET /api/v1/public/document/list	Модель/рест

getDocumentDetails	Получение детальной информации о документе	GET /api/v1/public/document/{documentId}	Модель/рест
getSigningParty	Получение информации о стороне подписи	GET /api/v1/public/signing-party/{signingPartyId}	Модель/рест
getDocumentFile	Получение информации о файле	GET /api/v1/public/document/{documentId}/file/{fileId}	Рест
changeStepByName	-	-	-
createContainerForDocument	Создать контейнер на подпись	POST /api/v1/public/signing-party/{signingPartyId}/container	Модель/рест
getStepNextList	-	-	-

- Signing signing-api

Текущий метод в сервисе	**Описание нового запроса**	**Запрос**	**Необходимые работы**
getContainerStatus	Получение информации о контейнере	GET /api/v1/public/container/{containerId}	Модель/рест
changeContainerStatus	Смена статуса контейнера	PATCH /api/v1/public/container/{containerId}/status	Модель/рест

- Profile profile-fl-api

Текущий метод в сервисе	**Описание нового запроса**	**Запрос**	**Необходимые работы**

getMyProfile	Получение информации о своем профиле	GET /api/v1/public/profile	Модель/рест

- · Certificate pc-api

Текущий метод в сервисе	**Описание нового запроса**	**Запрос**	**Необходимые работы**
createUser	1.1.2 Получение key json для установки ключей на МП	GET /api/v1/pay-control/certificate /{certificateId}/pc-user/create	Модель/рест
getAllTransactions	-	-	-
createCertificateEkyc	1.2.1 Получить информацию о сертификате по ID сессии eKYC	GET /api/v1/public/ekyc/{sessionId}/certificate	Модель/рест
-	Скачать файл акта признания ключа	GET /api/v1/public/certificate/{certificateId}/key-acceptance-act	Реализовать в pcInteractor
-	Принять акт признания ключа	POST /api/v1/public/certificate/{certificateId}/key-acceptance-act/accept	Реализовать в pcInteractor
-	Получить информацию о настройках сервера электронной подписи	GET /api/v1/settings/pay-control	Реализовать для работы Ekyc