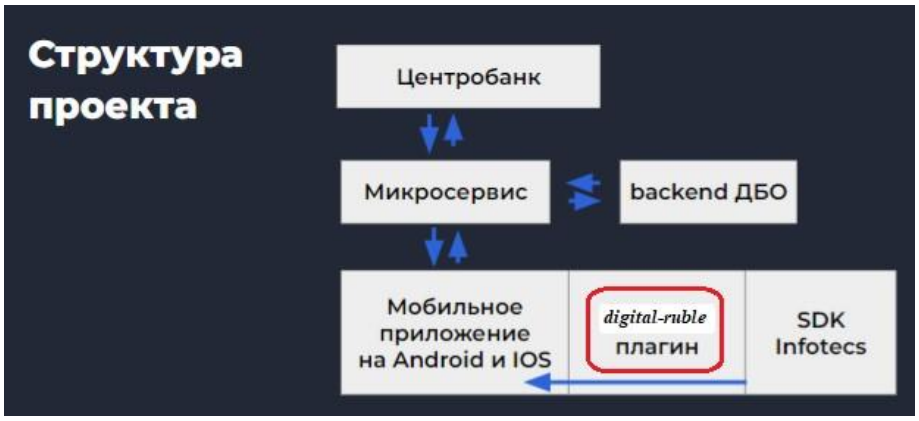
Функциональные характеристики и информация необходимая для установки, эксплуатации модуля «Abanking -плагин digital-ruble» (для мобильного приложения)

Функциональные характеристики

1. Основные требования

Реализован cordova[1] плагин для взаимодействия с SDK внутри приложения. Функционал разрабатывался для реализации интеграции с сервисами процесса «цифровой рубль» в мобильном приложении.

Схема взаимодействия:



Все операции с цифровым рублем происходят при помощи плагина. Содержит в себе следующий интерфейс для взаимодействия:

```
/** Инициализация SDK ЦР. */
coreInitialize(): Observable<void>

/** Получение уникального отпечатка устройства. */
getFingerPrint(): Observable<string>

/** Возвращает признак нужно ли собирать энтропию для работы SDK */
isEntropyRequired(): Observable<boolean>

/** Создание хранилища ключей
 * @param id id хранилища
 * @param pass string - пароль хранилища
 */
createKeyStore(id: number, pass: string): Observable<void>
/** Возвращает логическое значение, существует ли хранилище ключей с заданным id */
isKeyStoreExists(id: number): Observable<boolean>

/** Удалить хранилище ключей по заданному id */
deleteKeyStore(id: numbr): Observable<void>
```

[1] Cordova плагин - это платформенно-зависимый кусок кода, "обернутый" в JavaScript.

```
/** Логин в хранилище ключей с заданным id
 * @param id
 * @param pass - пароль хранилища
 */
keyStoreLogin(id: number, pass: string): Observable<boolean>

/** Логаут в хранилище ключей с заданным id */
keyStoreLogout(id: number): Observable<void>

/** Изменить пароль в хранилище ключей с заданным id */
changePassword(id: number, oldPassword: string, newPassword: string): Observable<void>

/** Импорт сертификата в формате base64 в хранилище ключей с заданным id */
keyStoreImportCertificate(certAsBase64: string, id: number): Observable<boolean>

/** Удалить хранилище вместе с ключами по заданному id */
deleteCertificateWithPrivateKey(id: number): Observable<void>

/** Создания запроса на сертификат (CSR) с помощью хранилища ключей по заданному id
*/
createCertRequest(id: number, requestParams: ICertRequestParams): Observable<string>

/** Возвращает признак, имеется ли в хранилище с заданным id сертификат с таким же
именем как id */
keyStoreContainsAlias(id: number): Observable<boolean>

/** Задать размеры окна для сбора энтропии */
setMotionBounds(width: number, height: number): Observable<void>

/** Добавить значение координаты в сбор энтропии */
addMotionEvent(x: number, y: number): Observable<number>

/** Получить список всех сертификатов в хранилище сертификатов */
getAliases(): Observable<string[]>

/** Проверяет наличие сертификата с указанным alias в хранилище сертификатов */
certStoreContainsAlias(alias: string): Observable<boolean>

/** Удалить сертификат с указанным alias */
deleteCertificate(alias: string): Observable<void>

/** получение сертфиката по alias. Возможно стоит убрать */
getCertificate(alias: string): Observable<void>

/** Получение типа сертфиката по alias */
getType(alias: string): Observable<string>

/** Импортировать сертификат в хранилище сертификатов */
```

```
certStoreImportCertificate(cert: IImportedCertificate): Observable<boolean>

/** Импортировать CRL в хранилище сертификатов */
importCRL(crlAsBase64: string): Observable<boolean>

/** Зашифровать сообщение указанным сертификатом */
encrypt(base64String: string, certAlias: string): Observable<string>

/**
 * Расшифровывает данные. Расшифрование возможно только если среди сертификатов в
 * хранилище ключей есть сертификат получателя. На хранилище ключей предварительно
должен быть
 * выполнен логин
 */
decrypt(id: number, base64String: string): Observable<string>

/**
 * Вычисляет "сырую" подпись данных data сертификатом certAlias из хранилища
 * keyStore. На keyStore предварительно должен быть выполнен KeyStore.login
 * Перед подписанием сертификат будет проверен.
 */
rawSign(id: number, base64String: string, certAlias: string): Observable<string>

/**
 * Вычисляет "сырую" подпись предоставленного хэша hash сертификатом certAlias
 * из хранилища keyStore. На keyStore предварительно должен быть выполнен
 * KeyStore.login Перед подписанием сертификат будет проверен.
 */
rawSignHash(id: number, base64String: string, certAlias: string): Observable<string>

/**
 * Выполняет CMS-подпись данных data сертификатом certAlias из хранилища
 * keyStore. На keyStore предварительно должен быть выполнен KeyStore.login
 * Перед подписанием сертификат будет проверен.
 */
sign(id: number, base64String: string, certAlias: string): Observable<string>

/**
 * Проверяет подписи в signData. При наличии в CMS нескольких структур
 * SignerInfo функция вернет результат проверки первой из них.
 */
verifySign(base64SignData: string, base64Data: string): Observable<void>
```

При взаимодействии с SDK были выделены следующие нюансы:

1. энтропия истекает спустя 1 месяц использования и ее нужно запрашивать заново;
2. нет проверки на то, вошли ли мы в хранилище ключей keyStore или нет.

Первый пункт решен тем, что если энтропия истекает, то все методы SDK начинают выбрасывать ошибку, однако текст ошибки неизвестен. Поэтому на все методы навешен обработчик ошибок, который в случае возникновения ошибки вызывает метод SDK isEntropyRequired, проверяющий необходимость в генерации новой энтропии.

Второй пункт решен добавлением кастомного метода, который в свою очередь вызывает метод SDK rawSing. Если вход в хранилище не выполнен, то метод плагина возвращает ошибку "Login is not performed".

2. Описание взаимодействия с запросами

Все взаимодействие с сервисом Центрального Банка происходит с помощью XML с нужным видом документа и каждый запрос xml содержит у себя:

- тег Document, в котором описывается схема самой xml. Все схемы с нашей стороны описаны в виде интерфейсов. Содержит атрибут xmlns с названием схемы, который описывает внутреннюю структуру документа.
- тег MsgHdr, включить MsgId - уникальный идентификатор сообщения, CreDt - время операции в форме изо-строки, Fr - идентификатор пользователя блоков рубля, To - идентификатор банка, куда отправляем сообщение, OprId - уникальный айди операции.
- тег FngrPrt - уникальные отпечатки устройств в формате base64

3. Описание работы Модуля "Плагин digital-ruble" внутри приложения

Регистрацию в разделе цифрового кошелька можно разделить на 3 этапа:

1. Вход в ЕСИА
2. Создание пароля
3. Генерация сертификата для пользования приложением.

Вход в ЕСИА осуществляется путем получения диплинка на esia через запрос. В который мы передаем successCallback и errorCallback, а в ответ получаем сам диплинк.

После создания пароля в приложение мы вызываем метод SDK с созданием хранилища ключей (keyStore). Но если до этого энтропия на устройстве не создавалась, то метод возвращает ошибку и мы генерируем энтропию.

Для того чтобы полноценно пользоваться цифровым рублем нужно создать хранилище ключей (keyStore) и добавить в него сертификат, который мы получаем при подтверждении регистрации.

Без сертификата пользоваться SDK нельзя.

Если сертификат потерялся или у него истек срок действия, то мы используем запрос на удаление сертификата и удаляем хранилище ключей, чтобы добавить новое.

После создания keyStore, входа в него и добавления в него сертификата мы можем полноценно пользоваться SDK.

Соответственно из-за того, что пользоваться SDK мы можем после того, как проинициализировались и произвели вход в хранилище ключей, то с главного экрана и из других разделов, без предварительного входа в раздел ЦР, запрсы содержащие xml мы отправлять не можем.

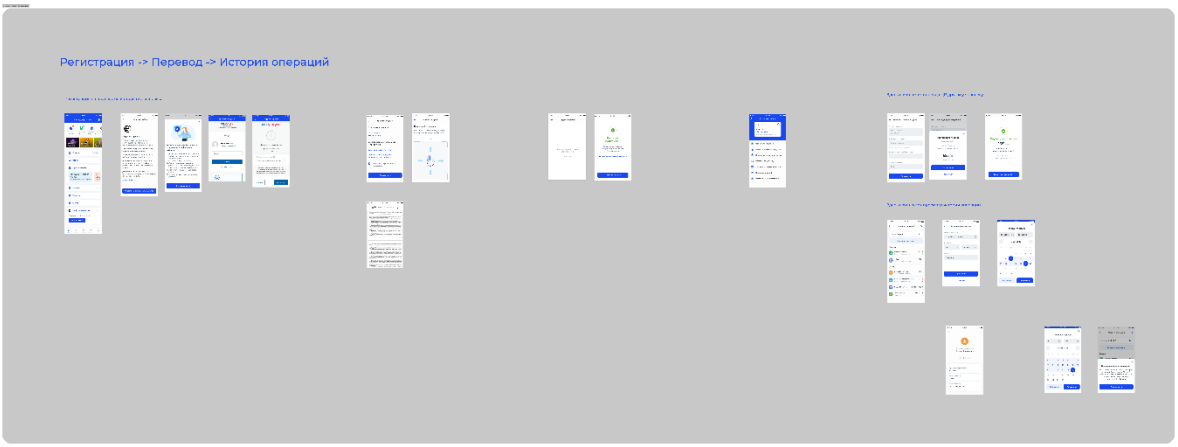
Информацию о кошельке, которую мы получали в авторизированной зоне, мы сохраняем в secureStorage устройства, для того, чтобы при повторном входе в приложение, пользователю отображалась последняя актуальная информация о кошельке.

Получение банковских сертификатов

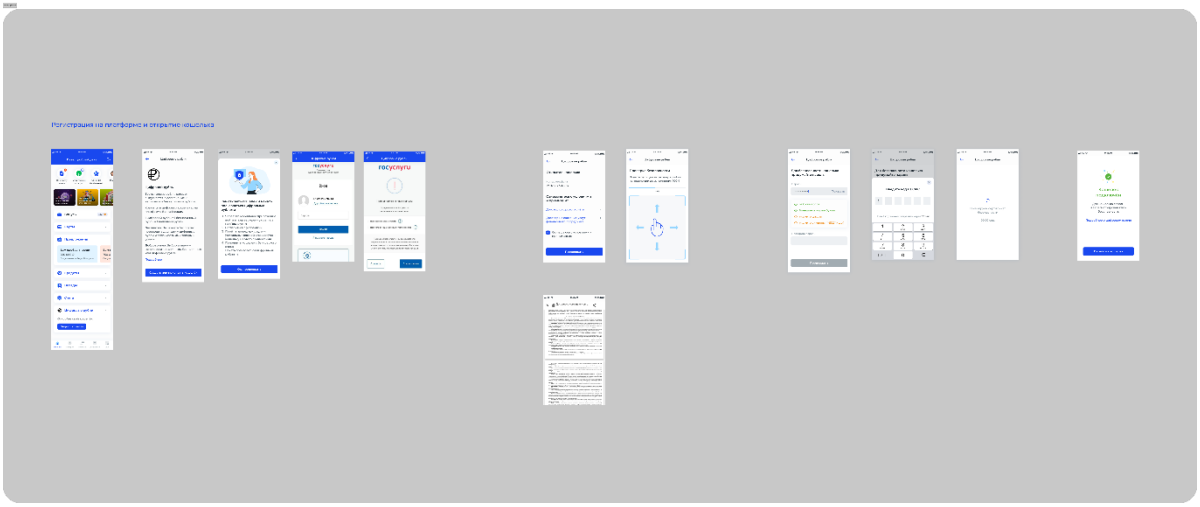
Получение всех сертификатов происходит путем запроса. После все сертификаты мы добавляем в хранилище сертификатов (certStore). Предварительно происходит проверка на то, есть ли такие сертификаты с таким же alias уже в хранилище. Если сертификат с таким alias уже имеется, то старый сертификат удаляется и добавляется новый.

4. Дизайн-макеты интерфейсов, которые возможно реализовать на основе внедрения плагина в мобильное приложение.

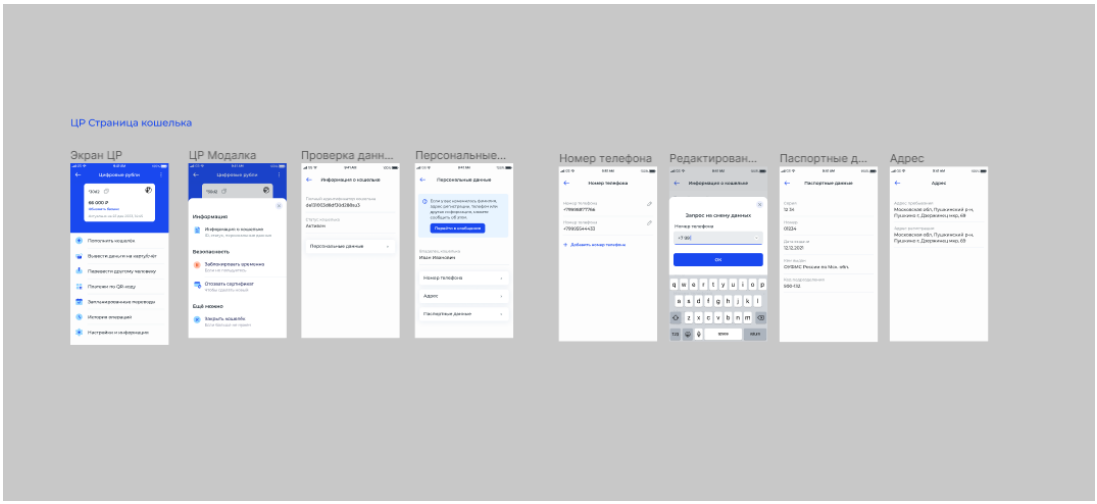
1 Сценарий : Регистрация -_ Перевод -_ История операций



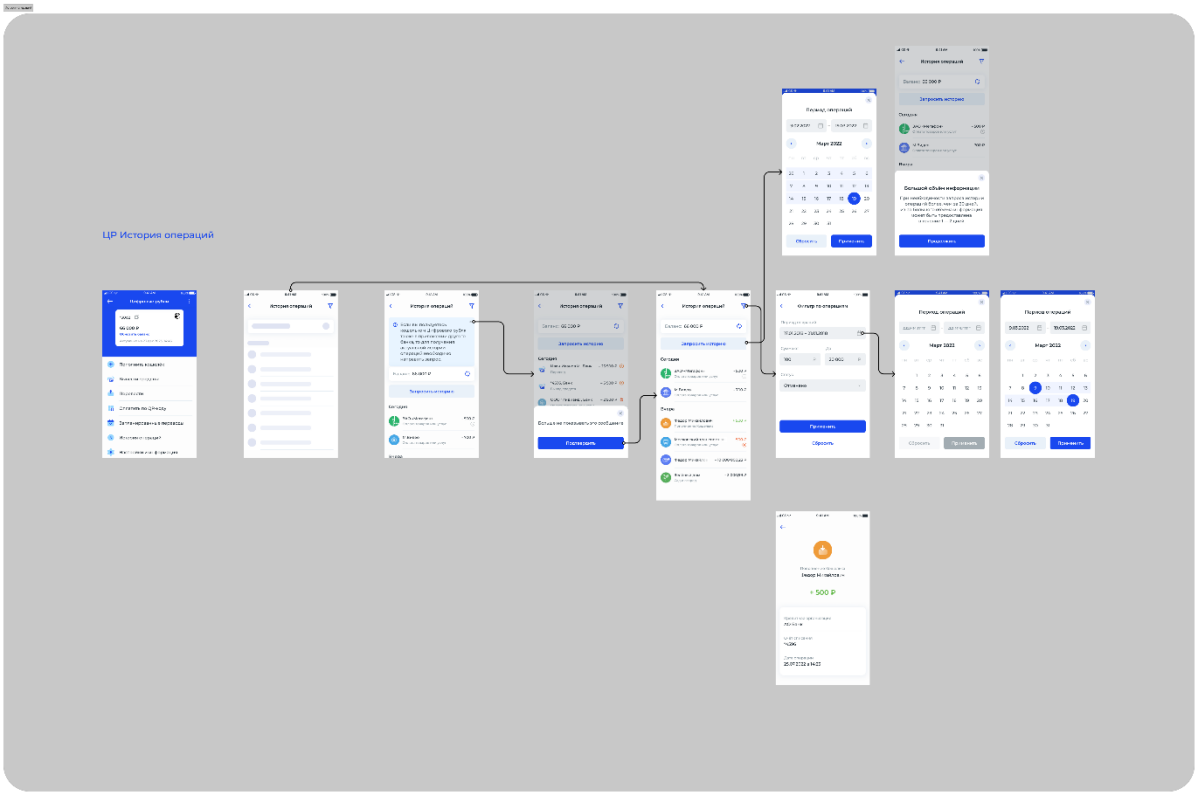
2 Сценарий: Регистрация.



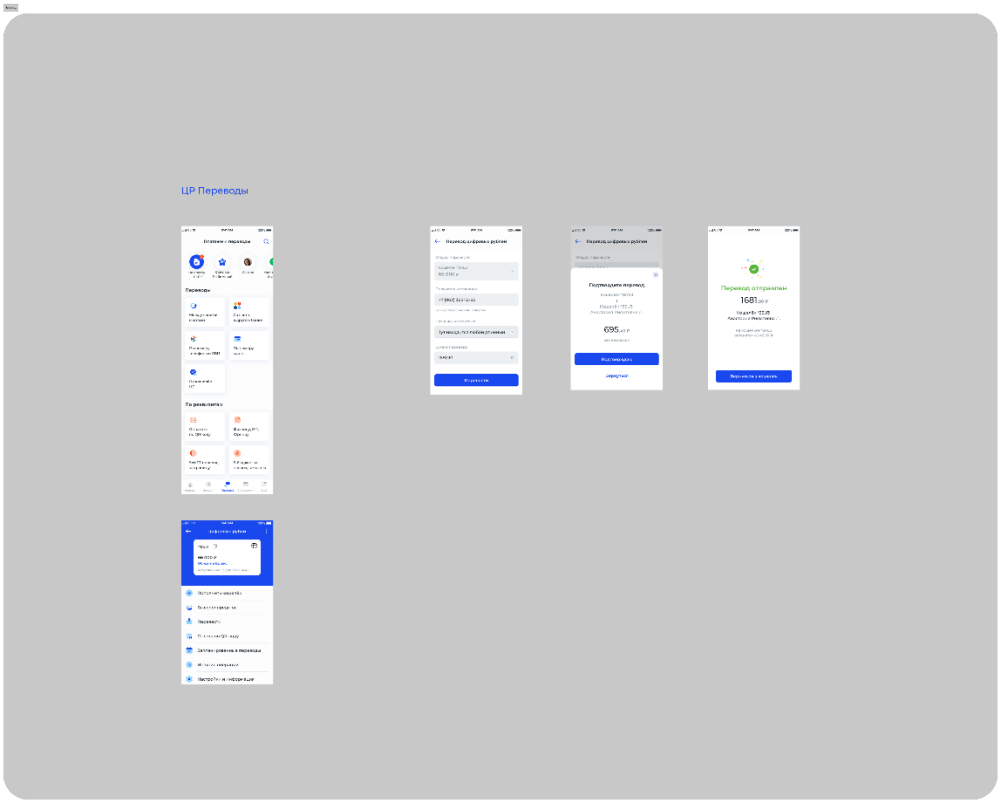
3 Сценарий: Страница кошелька.



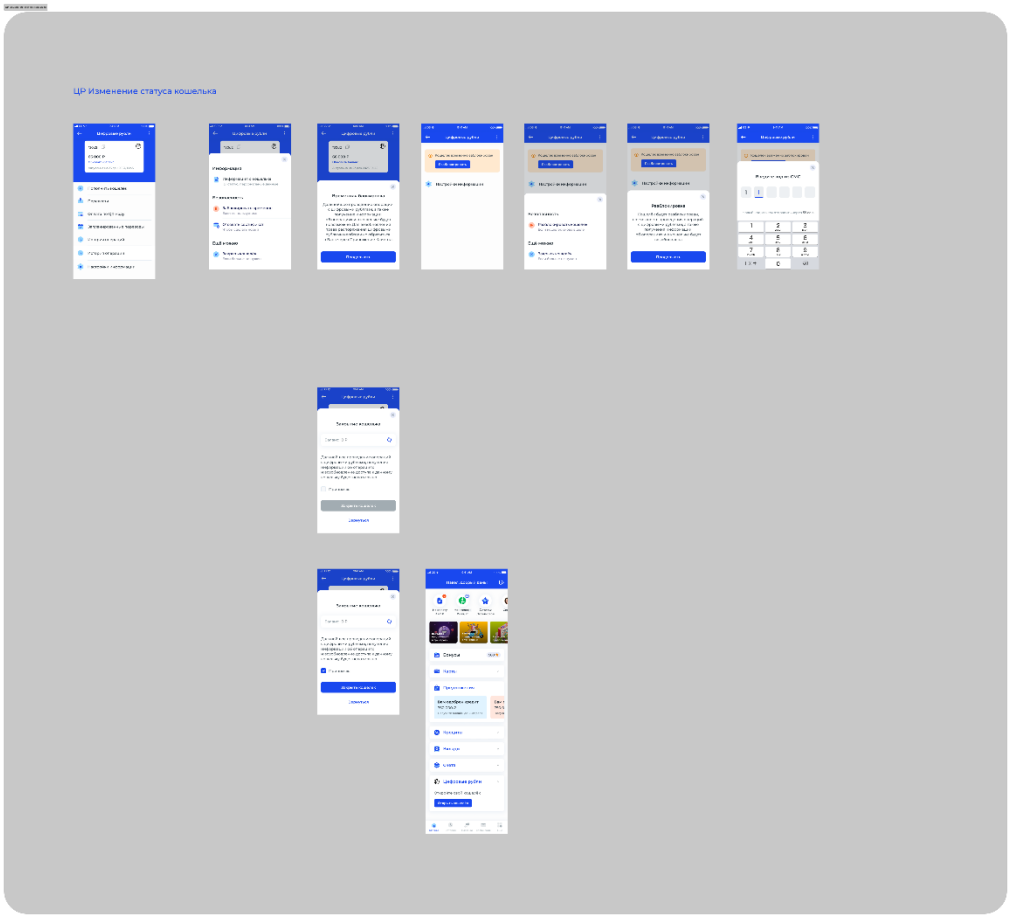
4 Сценарий: История операций



5 Сценарий: Перевод



6 Сценарий: Изменение статуса кошелька



7 Сценарий: Самоисполняемые сделки