Реализовать Flutter-интерфейс в существующем приложении Example. В интерфейсе реализовать проверку работы методов для нативного iOS Certificate Service (для Android части работы предполагается начать по его готовности).

Работа Certificate Service позволяет хранить, удалять, получать, создавать, импортировать, проверять сертификаты, списки отзывов и инкапсулирует в себе работу с разными хранилищами сертификатов: CPROCSP, Keychain, File System.

Текущем приложении-примере Example уже есть вкладка-заглушка без внутренних элементов-виджетов:

Необходимо добавить на эту вкладку элементы (кнопки/списки/текст), которые будут тестировать конкретные 5 функций CertificateService.

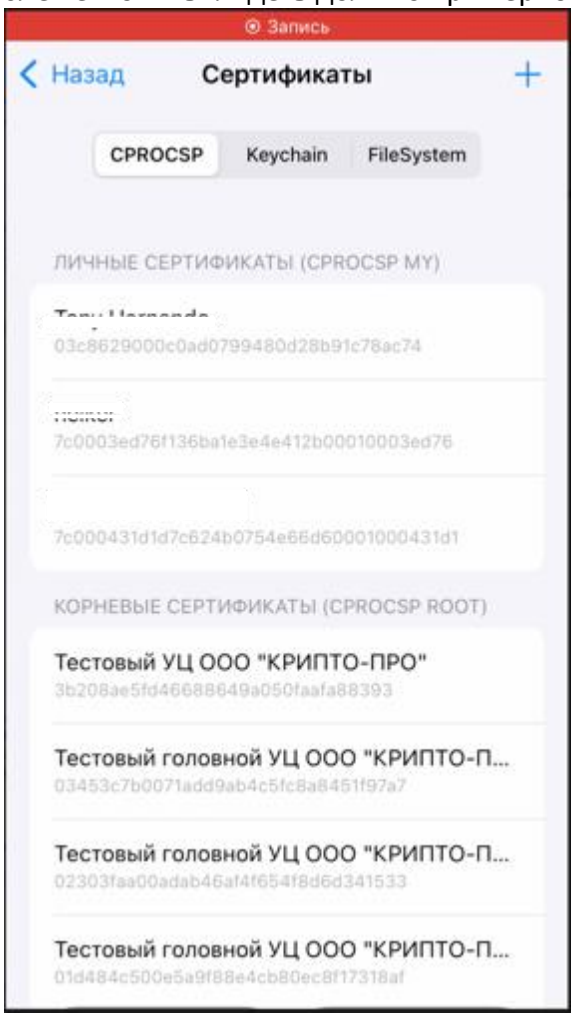
Реализация CertificateService на dart находится в файле lib/certificate_service.dart

Реализуемые функции:

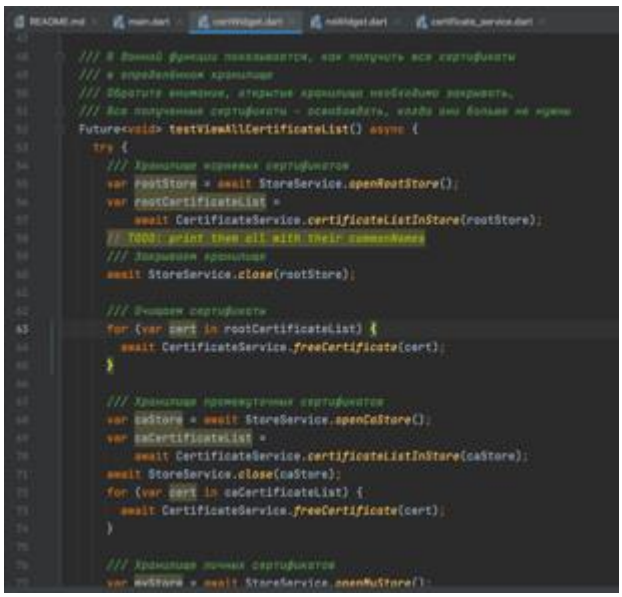
1. Просмотр списка всех сертификатов на устройстве с делением на корневые и личные
2. Добавление сертификата
3. Просмотр данных по сертификату
4. Просмотр цепочки сертификатов
5. Проверка (валидация) сертификата
6. Удаление сертификата

Просмотр списка всех сертификатов с делением на корневые и личные

Пустое пространство таба Certificates нужно заполнить список сертификатов полученных с устройства с разделением на группы: личные сертификаты, корневые сертификаты, промежуточные сертификаты. У каждой группы сделать заголовок и кликабельный список элементов. Выглядеть должно примерно вот так:



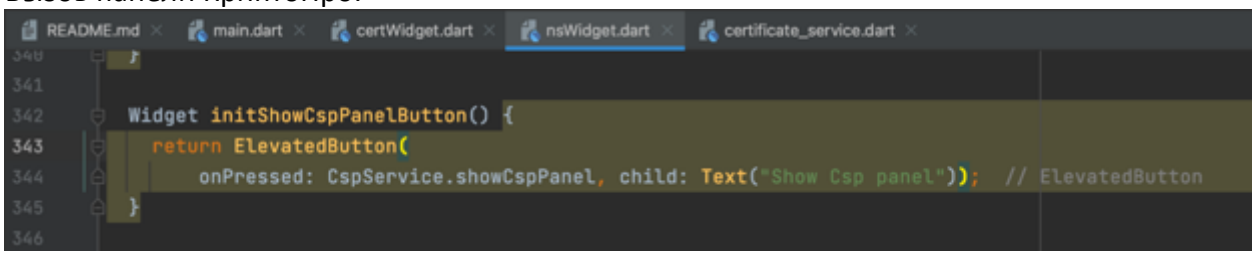
Для получения списка сертификатов следует воспользоваться уже написанным методом в certWidget.dart - _CertWidgetState.testViewAllCertificateList()



Добавление сертификата

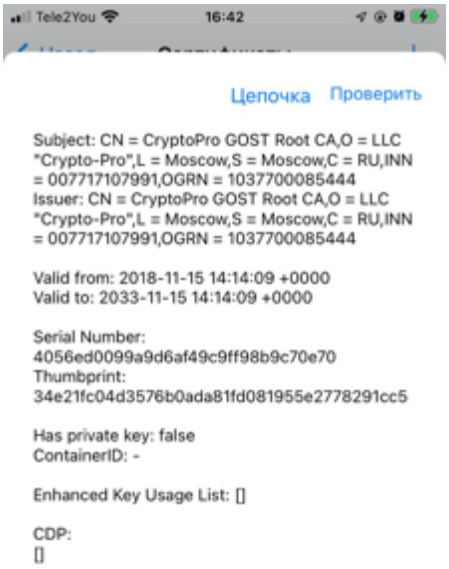
Для добавления сертификата необходимо разместить кнопку «Добавить» где-то на виджете, так чтобы ее было видно на первом экране. При нажатии кнопки вызываем панель панель КриптоПро. В рамках интерфейса Flutter приложения необходимо только ее вызвать, а далее все действия производятся внутри интерфейса панели КриптоПро согласно видео «Создание серта чере панель криптопро + просмотр списка и серта»

Вызов панели КриптоПро:



Просмотр данных по сертификату

В отображенном списке сертификатов все элементы представляющие сертификаты должны быть кликабельными. При клике на сертификат необходимо отобразить данные по сертификату в модальном окне или шторке всплывающей снизу экрана.
В этом же окне необходимо разместить кнопки «Цепочка» и «Проверить». Выглядеть это должно примерно вот так:



Все данные по сертификату, которые нужно отобразить, хранятся в самом объекте сертификата: class CertsApiCertificate

Просмотр цепочки сертификатов

Просмотр цепочки сертификатов вызывается кнопкой «Цепочка». Для построения цепочки сертификатов необходимо воспользоваться реализованным методом CertificateService.buildChain().
Цепочку сертификатов необходимо отобразить в виде древовидного списка, где верхним элементом будет сам выбранный сертификат и ниже – цепочка удостоверяющая его.
Выглядеть это может примерно вот так:

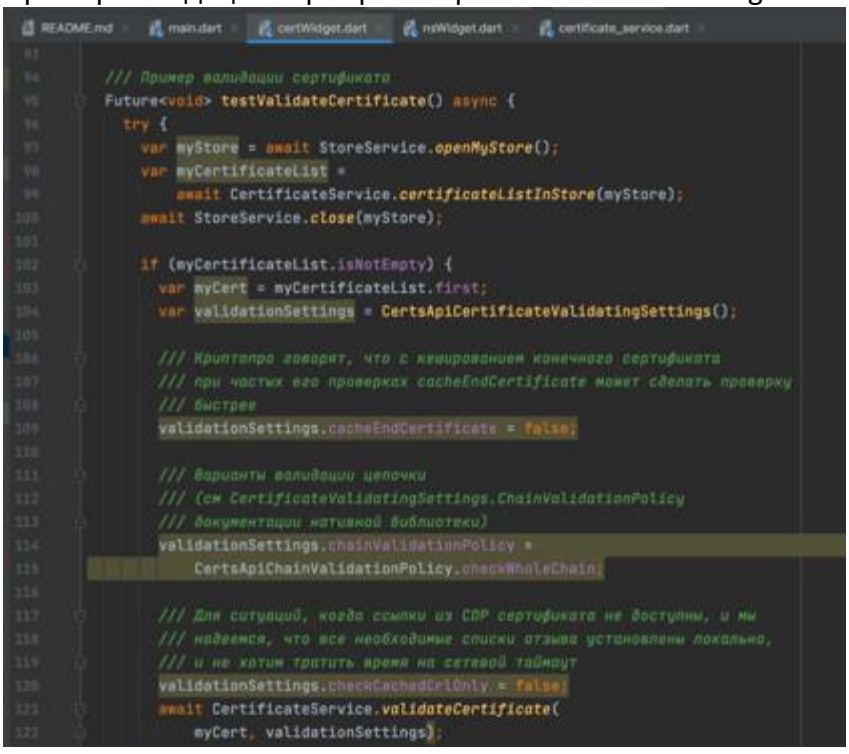
Или еще лучше в виде дерева примерно так (итоговый виджет дерева – на усмотрение разработчика):

Все элементы списка должны быть кликабельными. При клике на элемент списка мы отображаем описанное выше окно с подробностями сертификата и двумя кнопками «Цепочка» и «Проверить».

Проверка сертификата

Проверка сертификата должна вызывать кнопкой «Проверить» в окне отображающем информацию по сертификату.

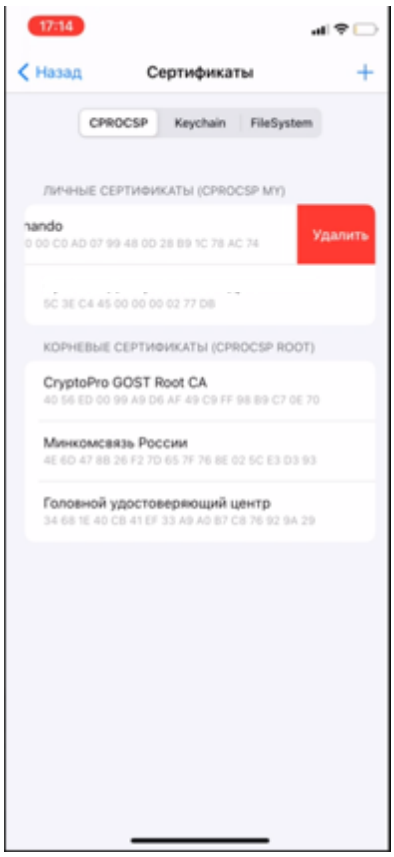
Пример валидации сертификата реализован в certWidget.dart:



Результат валидации сертификата показать во всплывающем окне.

Удаление сертификата

В списке (!!) сертификатов сделать элемент для вызова действия удаления сертификата (свайпом влево или кнопкой). Выглядеть это должно примерно вот так:

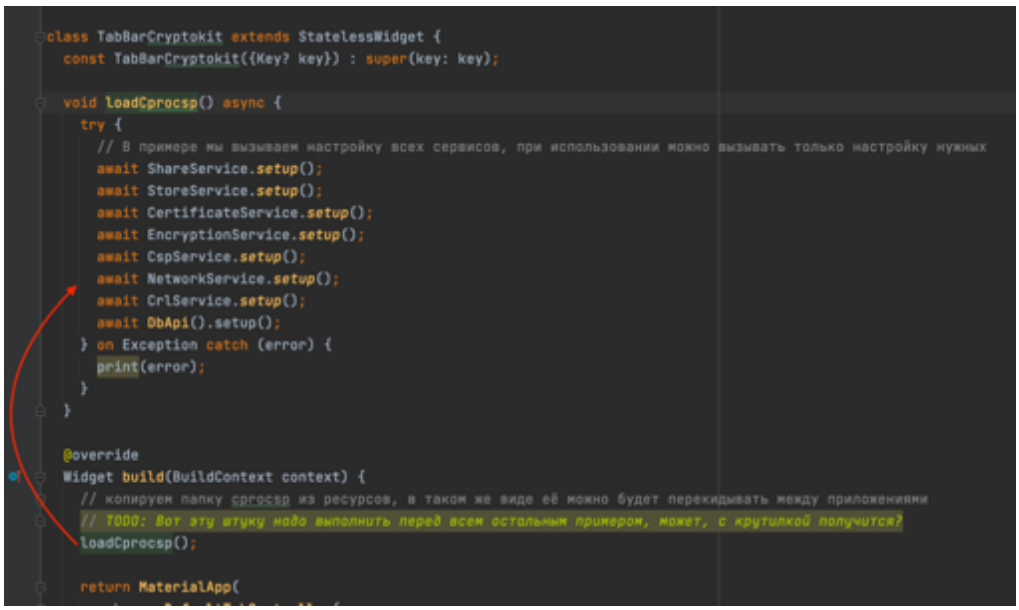


Для удаления сертификата из хранилища необходимо использовать уже реализованный метод CertificateService.removeCertificateFromStore.

Результат операции вывести в диалоговое окно и обновить список отображаемых на экране сертификатов.

Дополнительно:

При загрузке приложения вызываются методы setup у всех используемых в приложении сервисов, при это потенциально возможно длительное ожидание пользователем отклика от интерфейса:



```dart
class TabBarCryptokit extends StatelessWidget {
  const TabBarCryptokit({Key? key}) : super(key: key);

  void loadCprocsp() async {
    try {
      // В примере мы вызываем настройку всех сервисов, при использовании можно вызывать только настройку нужных
      await ShareService.setup();
      await StoreService.setup();
      await CertificateService.setup();
      await EncryptionService.setup();
      await CspService.setup();
      await NetworkService.setup();
      await CrlService.setup();
      await DbApi().setup();
    } on Exception catch (error) {
      print(error);
    }
  }

  @override
  Widget build(BuildContext context) {
    // копируем папку cprocsp из ресурсов, в таком же виде её можно будет перекидывать между приложениями
    // TODO: Вот эту штуку надо выполнить перед всем остальным примером, может, с крутилкой получится?
    loadCprocsp();

    return MaterialApp(
```

Для того, чтобы не было эффекта подвисания интерфейса, необходимо вставить loader, который скроется по завершении общей загрузки.