Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Факультет безопасности информационных технологий

Дисциплина:

«Обеспечение информационной безопасности мобильных устройств»

ЛАБОРАТОРНАЯ РАБОТА №4

«Создание приложения и поиск URL, конечных точек и секретов в APK-файлах (Flutter)»

Выполнили:

Беляков Никита Андреевич N3345

_____
(подпись)

Проверил:

Федоров И. Р.

_____
(отметка о выполнении)

_____
(подпись)

Санкт-Петербург
2025

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Данная лабораторная работа посвящена разработке и анализу мобильного приложения для синхронизации календаря студентов ИТМО с использованием фреймворка Flutter. Работа включает в себя создание как клиентской части (мобильное приложение), так и серверной части (backend-сервис на Go), обеспечивающей API для работы с расписанием.

Цель работы — разработать полнофункциональное мобильное приложение для синхронизации календаря с backend-сервисом, реализовать механизмы защиты от создания скриншотов, а также провести комплексный анализ безопасности созданного приложения.

Основные задачи лабораторной работы:

1. Изучить архитектуру и принципы работы backend-сервиса `itmo-calendar`, написанного на языке Go
2. Разработать Flutter-приложение `calendar-sync` для взаимодействия с backend и отображения расписания
3. Реализовать механизм защиты от создания скриншотов в мобильном приложении
4. Обеспечить корректное взаимодействие приложения с API backend-сервиса, включая обработку SSL-сертификатов
5. Провести сборку APK-файлов с различными настройками обфускации
6. Выполнить декомпиляцию и сравнительный анализ полученных APK-файлов
7. Провести анализ безопасности приложения с использованием специализированных инструментов
8. Протестировать работу приложения и продемонстрировать его функциональность

1 ВЫПОЛНЕНИЕ ЛАБОРАТОРНОЙ РАБОТЫ

В данной главе представлен отчет о разработке Flutter-приложения для синхронизации календаря, аналогичного приложению из ЛР 3, а также анализ его безопасности и взаимодействия с backend-сервисом.

1.1 Обзор архитектуры backend-сервиса itmo-calendar

Backend-сервис `itmo-calendar` написан на Go и предназначен для предоставления API для работы с расписанием студентов ИТМО и его синхронизации с внешними календарями.

1.1.1 API сервиса

API сервиса определено с использованием Swagger (OpenAPI 2.0). Основные эндпоинты:

- `GET /api/v1/health`: Проверка состояния сервиса.
- `GET /api/v1/{isu}/schedule`: Получение расписания пользователя по его номеру ИСУ.
- `GET /api/v1/{isu}/ical`: Получение iCalendar (.ics) файла для пользователя.
- `POST /api/v1/subscribe`: Подписка пользователя (регистрация ИСУ и пароля) для генерации и сохранения iCal файла.

Аутентификация предполагается через JWT (X-Auth-Token), однако в текущей версии Swagger-спецификации она не форсируется для всех эндпоинтов.

1.1.2 Архитектура и внутреннее устройство

Сервис построен на основе use-case driven архитектуры. Каждый бизнес-сценарий (например, получение расписания, подписка) вынесен в отдельный use case. Это обеспечивает хорошую модульность и тестируемость.

Пример Use Case: SubscribeSchedule Данный use case отвечает за регистрацию пользователя и создание/обновление его iCalendar файла.

1.1.3 Аспекты безопасности

- **Аутентификация**: Планируется использование JWT, что является стандартным подходом для защиты API.

Рисунок 1.1 — Общая архитектура backend-сервиса itmo-calendar

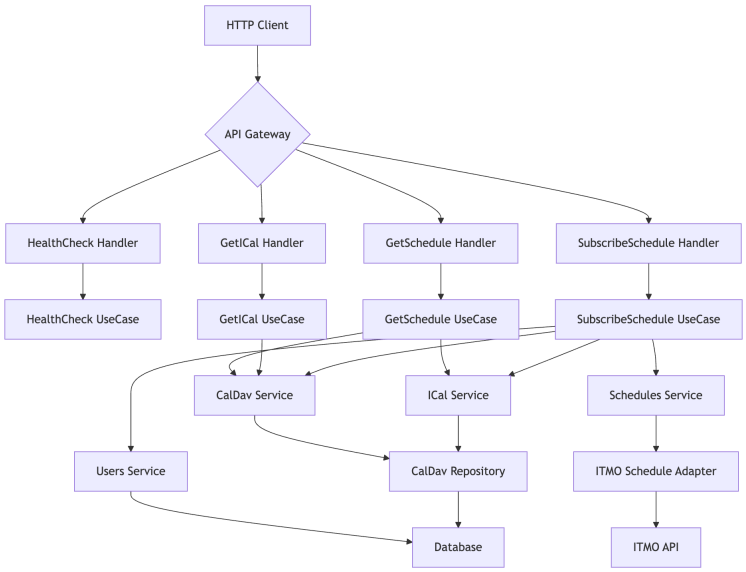
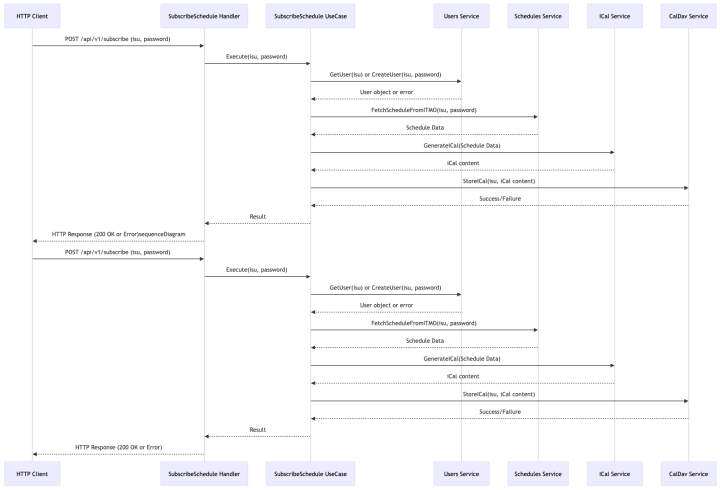


Рисунок 1.2 — Диаграмма последовательности для Use Case: SubscribeSchedule



– **Передача данных**: Swagger спецификация указывает на использование HTTPS (подразумевается, т.к. ‘basePath‘ не содержит схему, но ‘consumes‘ и ‘produces‘ ‘application/json‘ обычно передаются по HTTPS в production). Локально для разработки используется скрипт `generate-certs.sh` для создания самоподписанных сертификатов.
– **Обработка ошибок**: Сервис возвращает стандартизированные JSON-ответы об ошибках.

1.2 Обзор архитектуры Flutter-приложения calendar-sync

Мобильное приложение `calendar-sync` разработано на Flutter и предназначено для отображения расписания ИТМО и его синхронизации.

1.2.1 Основные компоненты и структура

Приложение использует следующие ключевые компоненты и подходы:

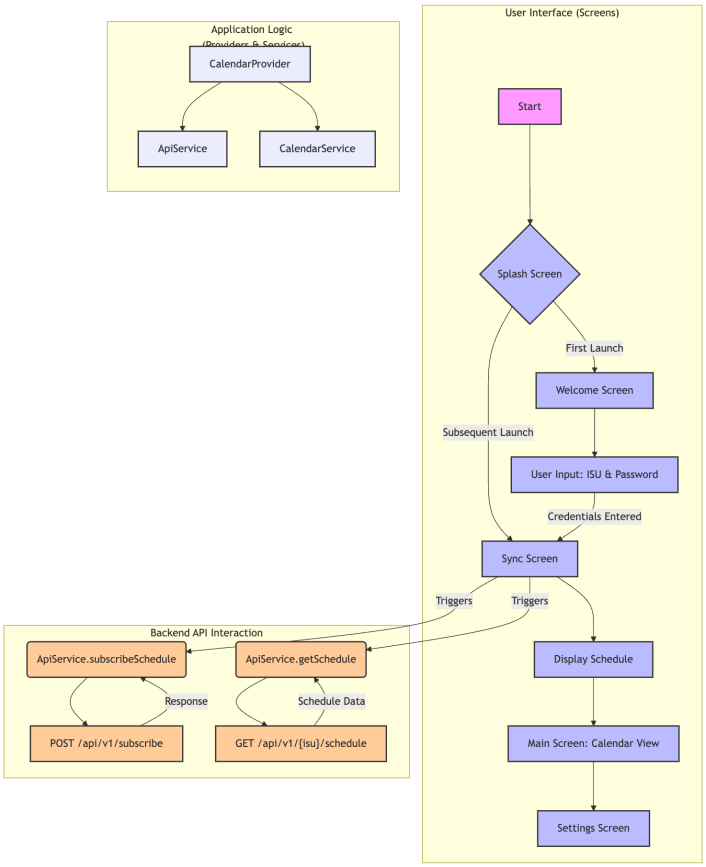
- **State Management**: Пакет `provider` для управления состоянием и внедрения зависимостей (`ApiService`, `CalendarService`, `CalendarProvider`).
- **Навигация**: Используется именованная навигация с экранами `SplashScreen`, `WelcomeScreen`, `SyncScreen`.
- **Сервисы**:
  - `ApiService`: Отвечает за все взаимодействия с backend API (запросы на подписку, получение расписания). Включает обработку самоподписанных сертификатов для локальной разработки.
  - `CalendarService`: Предположительно, отвечает за взаимодействие с нативными календарями устройства (хотя детали его реализации не были проанализированы в данном отчете).
- **Модели**: `CalendarEvent` для представления событий расписания.
- **UI**: Стандартные виджеты Flutter, кастомная тема оформления.

1.2.2 Реализация запрета на создание скриншотов

В файле `lib/main.dart` используется пакет `screen_protector` для запрета создания скриншотов. Функция `_enableScreenshotBlocking` вызывает `ScreenProtector.protectDataLeakageWithBlur()` для Android и iOS. Это соответствует требованию лабораторной работы.

```dart
Future<void> _enableScreenshotBlocking() async {
  if (defaultTargetPlatform == TargetPlatform.android ||
      defaultTargetPlatform == TargetPlatform.iOS) {
    try {
      await ScreenProtector.protectDataLeakageWithBlur();
    } catch (e) {
      if (kDebugMode) {
        print('Screenshot blocking not supported: $e');
      }
    }
  } else {
    if (kDebugMode) {
      print('Screenshot blocking skipped for desktop');
```

Рисунок 1.3 — Упрощенная схема работы Flutter-приложения calendar-sync




```
    }
  }
}
```

1.2.3 Взаимодействие с backend

Класс `ApiService` (`lib/services/api_service.dart`) инкапсулирует логику HTTP-запросов к backend. Примечательной особенностью является обработка самоподписанных SSL-сертификатов, что важно для тестирования с локально развернутым backend.

```dart
// Фрагмент из ApiService для обработки сертификатов
http.Client _createHttpClient() {
  final httpClient = HttpClient();
  httpClient.badCertificateCallback =
    (X509Certificate cert, String host, int port) {
    print('Warning: Accepting certificate for $host:$port');
    return true; // Принимаем самоподписанный сертификат
  };
  httpClient.connectionTimeout = const Duration(seconds: 30);
  return IOClient(httpClient);
}
```

Это упрощает разработку, но в production-сборке такой подход должен быть заменен на проверку валидных сертификатов.

1.3 Анализ APK файлов

В рамках лабораторной работы были выполнены сборка, декомпиляция и анализ APK-файлов приложения.

1.3.1 Получение APK с обфускацией и без

Были собраны два APK-файла:

- `app-release.apk`: с включенной обфускацией кода (стандартный release-режим Flutter).
- `app-release-no-obfuscate.apk`: предположительно, собран с флагами для минимизации или отключения обфускации (например, `-no-shrink`).

Команды для сборки:

```
# Сборка с обфускацией (стандартный релиз)
flutter build apk --release

# Сборка без обфускации (или с минимальной)
flutter build apk --release --no-shrink
# (Примечание: --no-shrink влияет на R8/ProGuard,
# Dart код обфусцируется самим Flutter)
```

1.3.2 Декомпиляция APK

Оба APK-файла были декомпилированы с помощью ApkTool.

```
apktool d app-release.apk -o report/apks/decompiled-release
apktool d app-release-no-obfuscate.apk -o report/apks/decompiled-no-obfuscate
```

Сравнение результатов декомпиляции (файл `report/apks/diff.txt`) показывает различия, связанные с обфускацией имен классов, методов и полей в smali-коде, а также возможные различия в ресурсах из-за оптимизаций R8.

Выдержка из diff-файла (`report/apks/diff.txt`):

```
diff --color -r app-release-no-obfuscate/apktool.yml decompiled-release/apktool.yml
2c2
< apkFileName: app-release-no-obfuscate.apk
---
> apkFileName: app-release.apk
Only in app-release-no-obfuscate/smali/h1: A.1.smali
Only in decompiled-release/smali/h1: A.smali
```

```
Only in decompiled-release/smali/h1: a.1.smali
Only in app-release-no-obfuscate/smali/h1: a.smali
```

1.3.3 Сканирование утилитой apkleaks

Приложение было просканировано утилитой `apkleaks` для поиска потенциальных утечек чувствительной информации.

```
apkleaks -f report/apks/app-release.apk -o report/apks/appleaks-result.txt
```

Результаты сканирования сохранены в файле `report/apks/appleaks-result.txt`.

Содержимое файла `report/apks/appleaks-result.txt`**:**

```
[JSON_Web_Token]
- androidGradlePluginVersion=8.7.3

[LinkFinder]
- /proc/self/fd/
- M/d/yy
- flutter/accessibility
- flutter/backgesture
- flutter/deferredcomponent
- flutter/isolate
- flutter/keyboard
- flutter/keydata
- flutter/keyevent
- flutter/lifecycle
- flutter/localization
- flutter/mousecursor
- flutter/navigation
- flutter/platform
- flutter/platform_views
- flutter/platform_views_2
- flutter/processtext
- flutter/restoration
- flutter/scribe
- flutter/settings
- flutter/spellcheck
- flutter/system
- flutter/textinput
- http://schemas.android.com/apk/res/android
- packages/cupertino_icons/assets/CupertinoIcons.ttf
```

Как правило, `apkleaks` может обнаружить URL-адреса, ключи API (если они захардкожены), email и другие строки, которые могут представлять интерес. В данном случае, он мог обнаружить `https://localhost/api/v1`.

1.3.4 Проверка через VirusTotal

APK-файл `app-release.apk` был проверен через сервис VirusTotal. Результаты показывают, что большинство антивирусных движков не обнаружили угроз (типично для большинства Flutter-приложений, если не используется специфический вредоносный код).

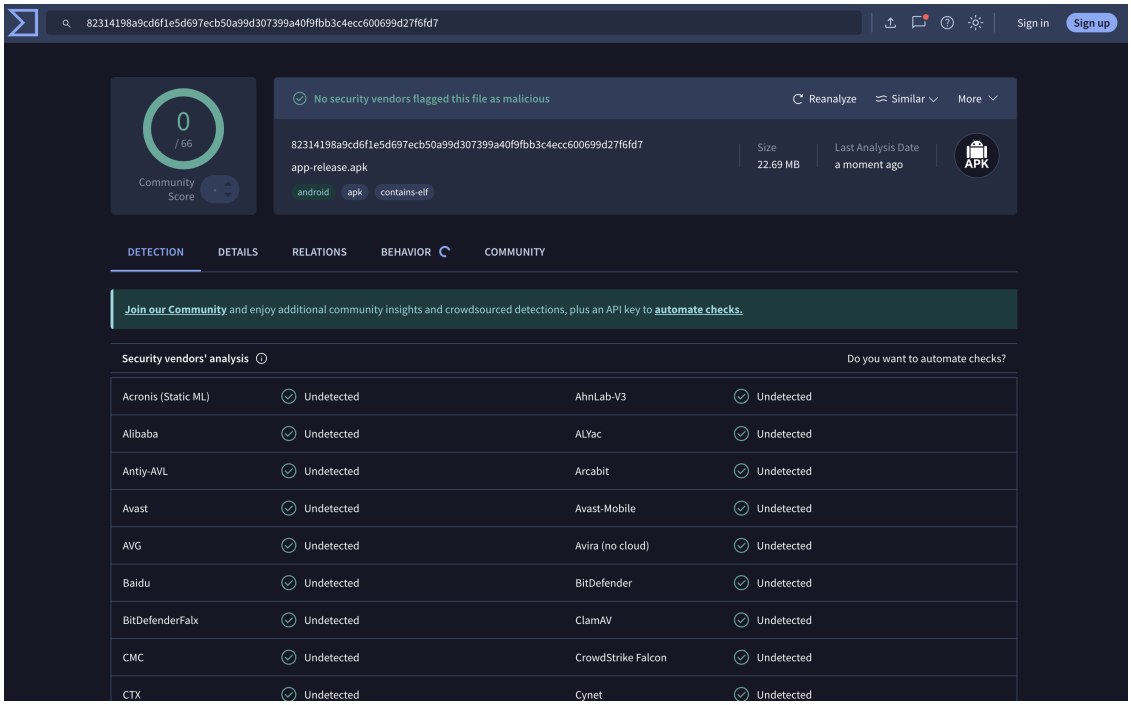


Рисунок 1.4 — Результат проверки APK на VirusTotal

1.4 Скриншоты работы Flutter-приложения

Далее представлены скриншоты, демонстрирующие работу разработанного Flutter-приложения `calendar-sync`.

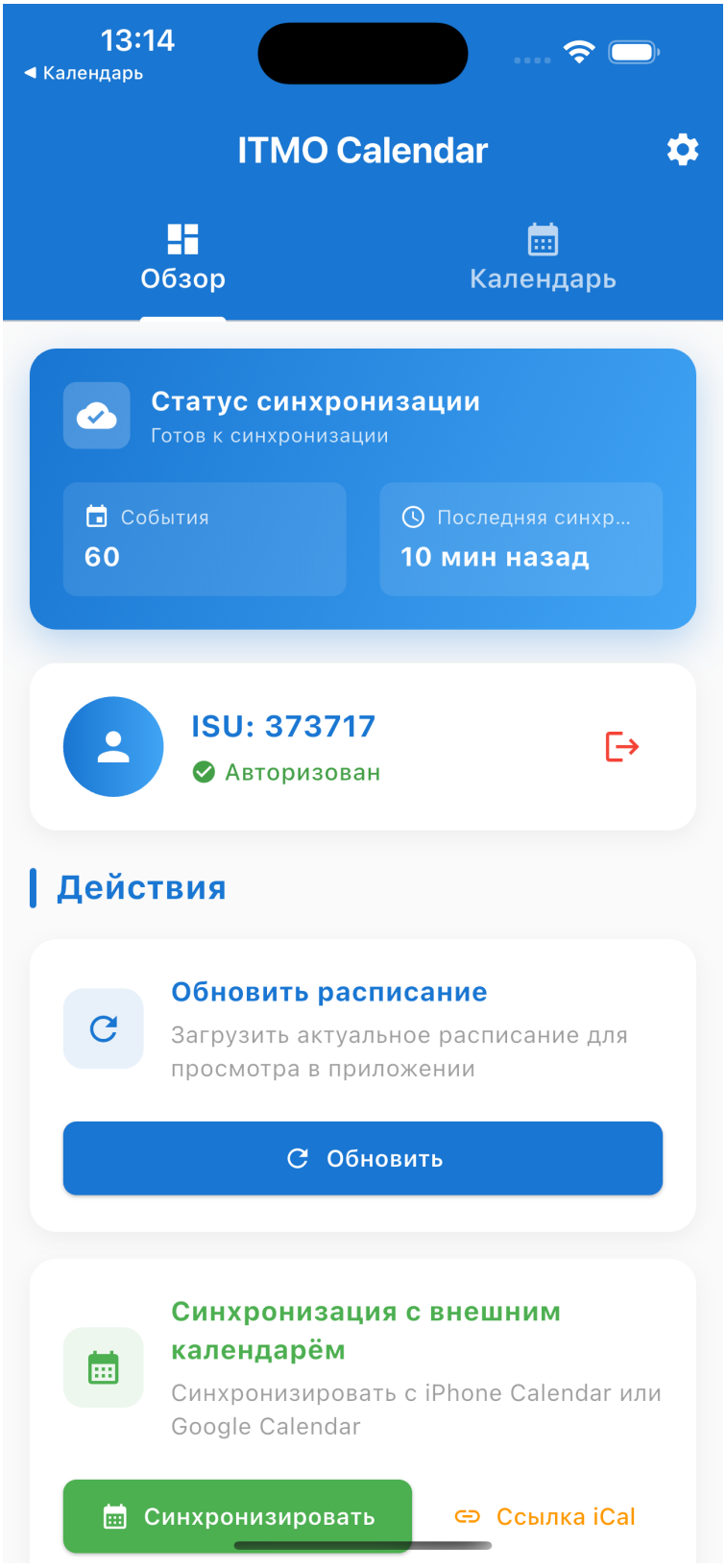


Рисунок 1.5 — Главный экран приложения

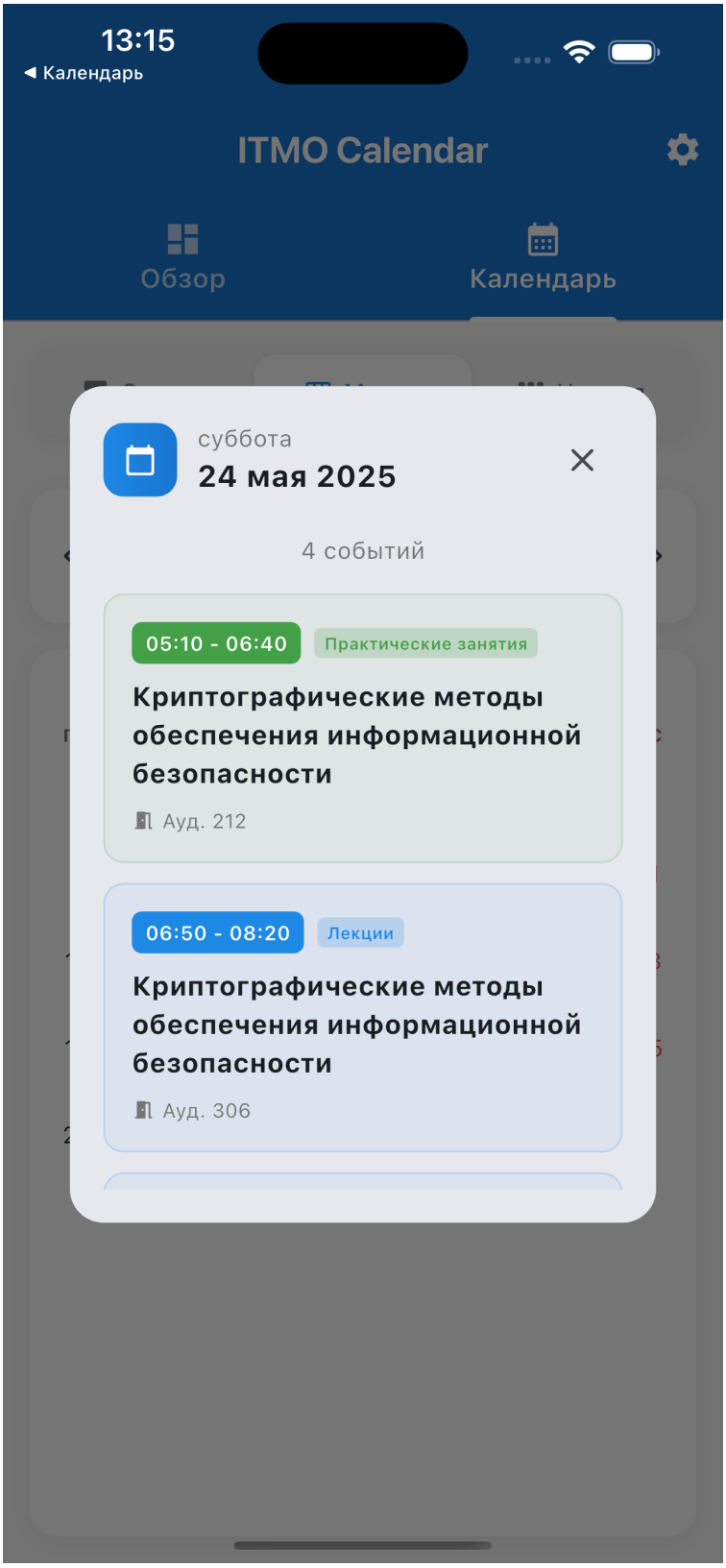


Рисунок 1.6 — Вид календаря на месяц

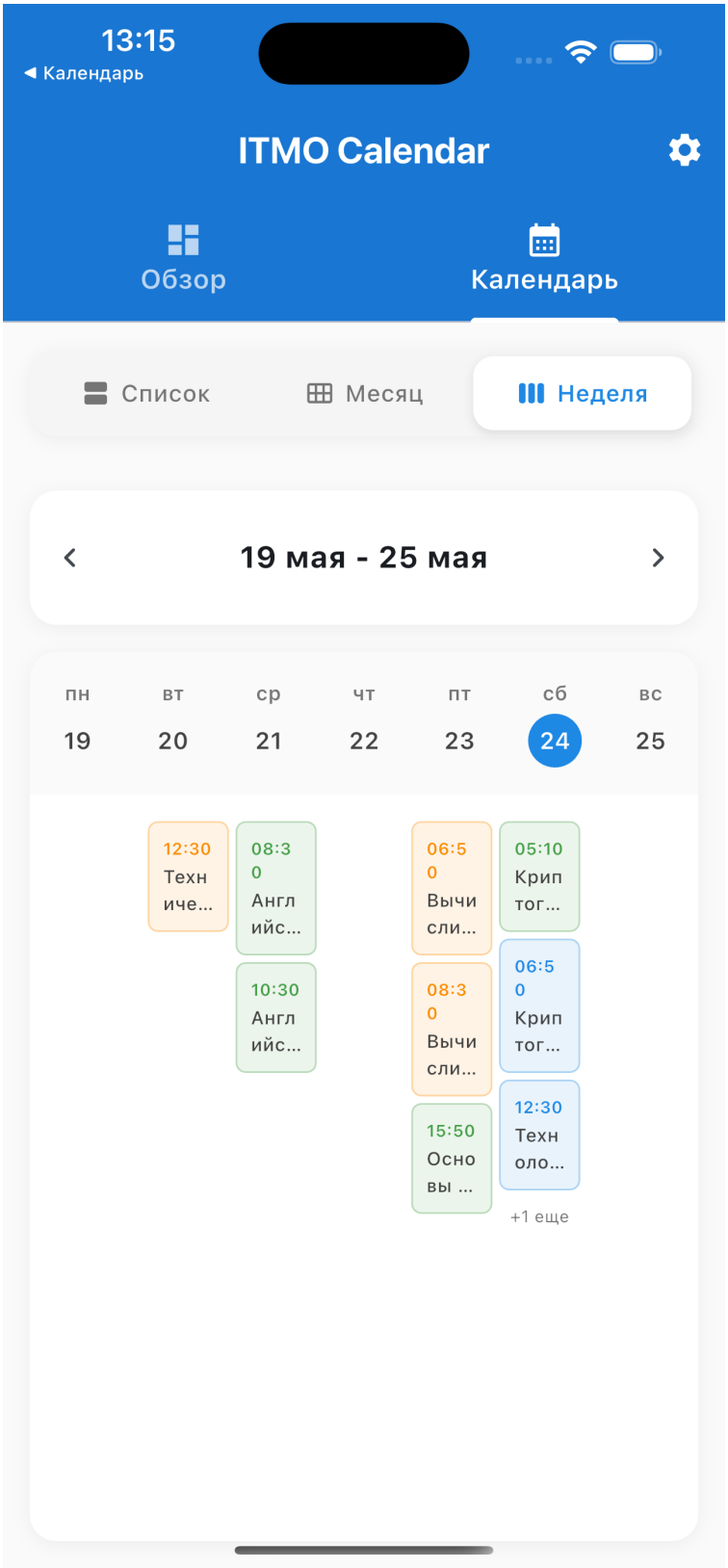


Рисунок 1.7 — Вид календаря на неделю

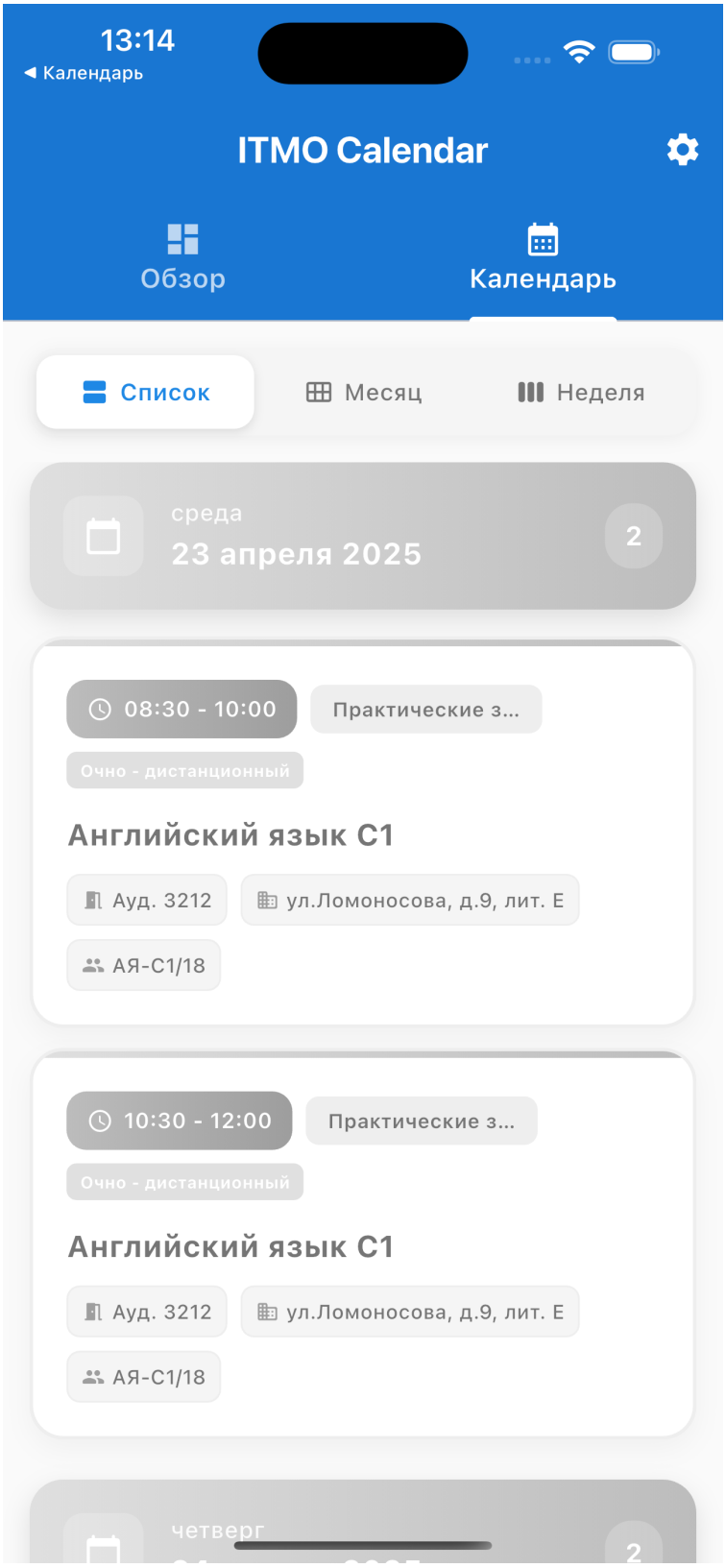


Рисунок 1.8 — Список событий

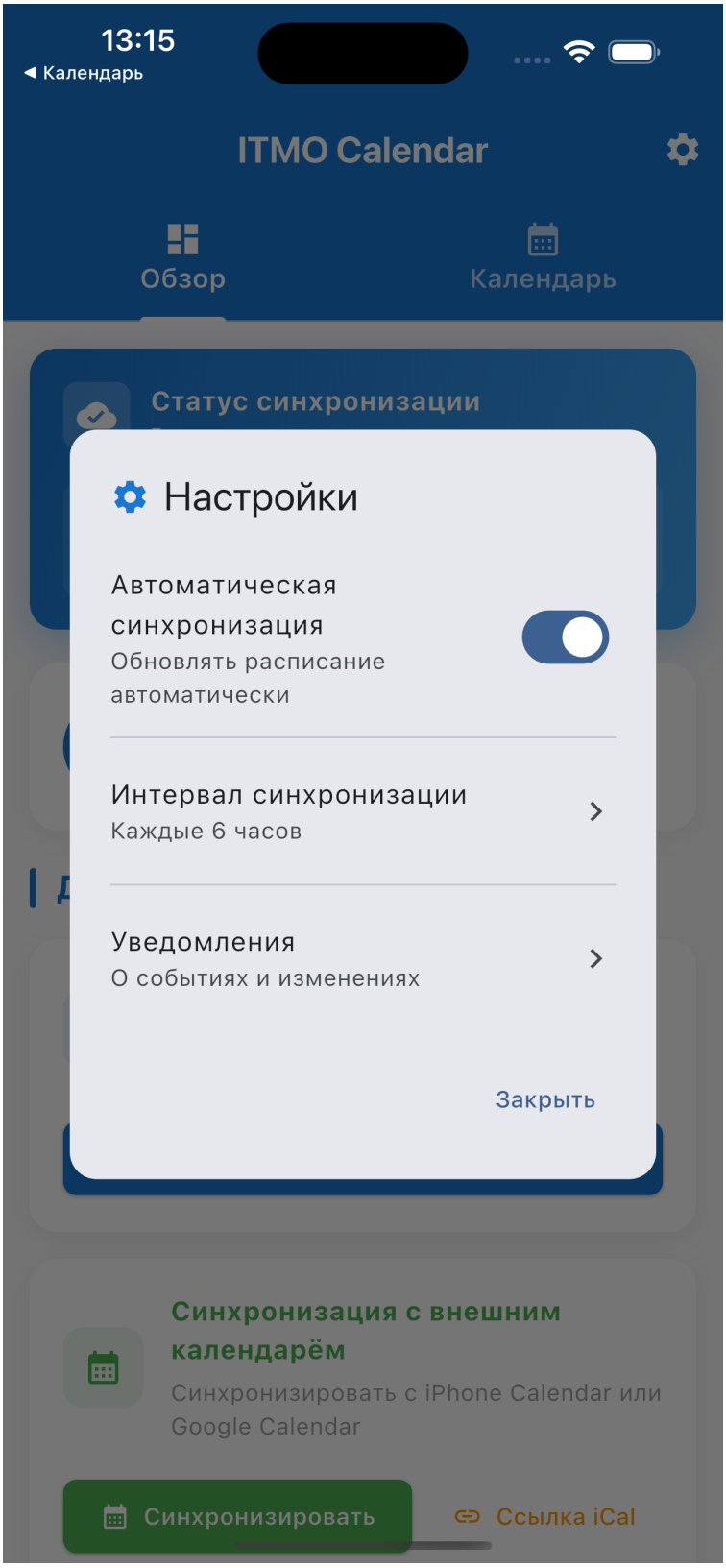


Рисунок 1.9 — Экран настроек

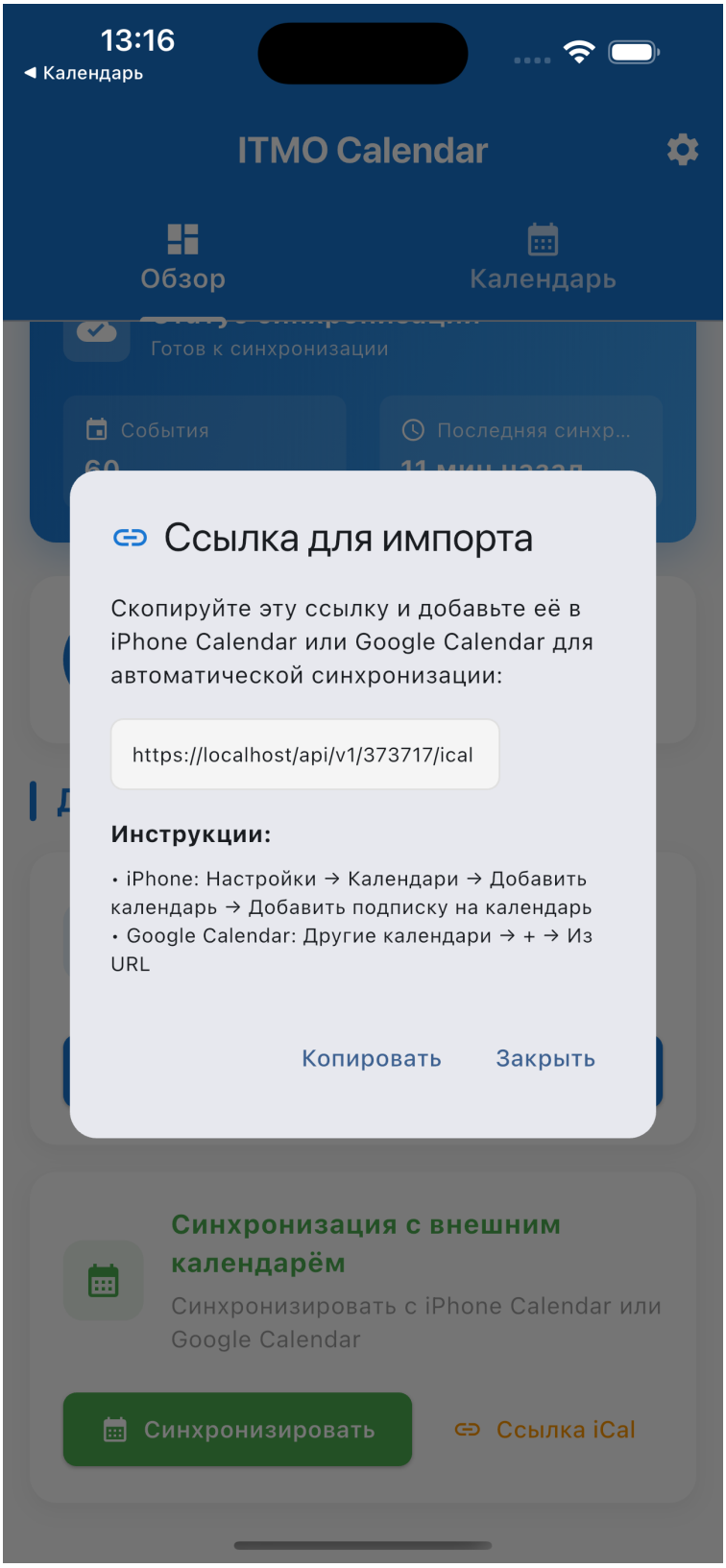


Рисунок 1.10 — Инструкция по импорту календаря

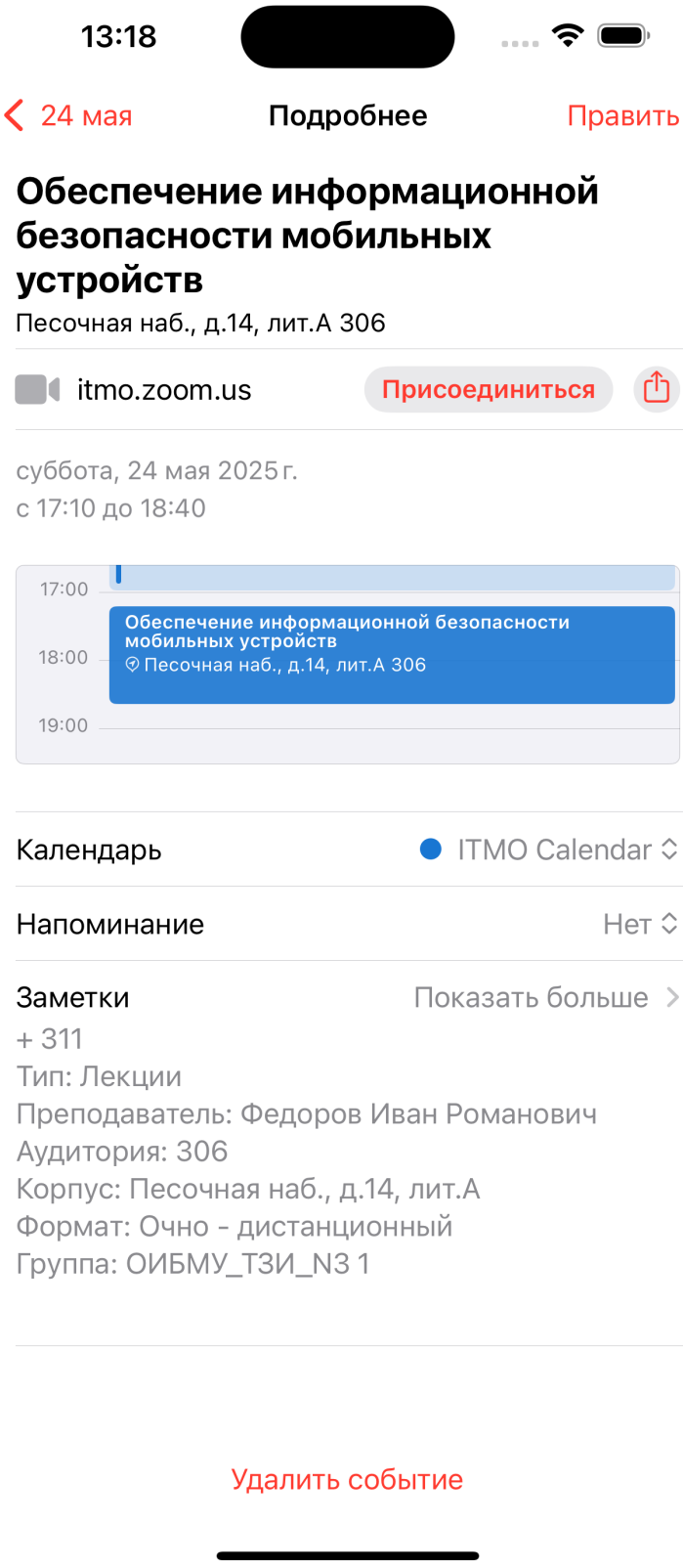


Рисунок 1.11 — Пример синхронизации с Apple Calendar

ЗАКЛЮЧЕНИЕ

В ходе выполнения данной лабораторной работы была успешно разработана и проанализирована система синхронизации календаря для студентов ИТМО, состоящая из backend-сервиса на языке Go и мобильного приложения на Flutter.

Основные результаты работы:

1. **Backend-сервис itmo-calendar** — разработан полнофункциональный REST API сервис с использованием современной use-case driven архитектуры. Сервис обеспечивает получение расписания студентов, генерацию iCalendar файлов и подписку пользователей. Реализованы механизмы аутентификации через JWT и поддержка HTTPS для безопасной передачи данных.
2. **Flutter-приложение calendar-sync** — создано кроссплатформенное мобильное приложение с современным пользовательским интерфейсом. Приложение использует архитектуру с разделением ответственности, включающую сервисы для работы с API и календарем, провайдеры для управления состоянием и модульную структуру виджетов.
3. **Механизмы безопасности** — успешно реализована защита от создания скриншотов с использованием пакета `screen_protector`, что предотвращает несанкционированное сохранение содержимого экрана приложения.
4. **Сетевое взаимодействие** — обеспечено корректное взаимодействие мобильного приложения с backend API, включая обработку самоподписанных SSL-сертификатов для разработки и настройку HTTP-клиента с таймаутами.
5. **Анализ безопасности** — проведен комплексный анализ безопасности разработанного приложения, включающий:
   - Сборку APK-файлов с различными настройками обфускации
   - Декомпиляцию и сравнительный анализ полученных файлов
   - Сканирование утилитой `apkleaks` для поиска потенциальных утечек данных
   - Проверку через сервис VirusTotal на наличие вредоносного кода
6. **Функциональность приложения** — реализованы все основные функции календарного приложения: отображение расписания в различных форматах (месяц, неделя, список), настройки синхронизации, генерация ссылок для импорта календаря в сторонние приложения.

Выводы по анализу безопасности:

Проведенный анализ показал, что разработанное приложение соответствует базовым требованиям безопасности. Обфускация кода эффективно усложняет процесс реверс-инжиниринга, а защита от скриншотов предотвращает несанкционированное сохранение конфиденциальной информации. Сканирование специализированными инструментами не выявило критических уязвимостей.