11 лекция.

Тема: Flutter: научитесь работать с API на примере мобильного приложения с запущенным ботом Telegram в Android Studio

План:

1. **Разработка прототипа**
2. **Генерация SSL сертификата**
3. **Автоматическое создание бота**

1. Разработка прототипа

Изучение темы создания Telegram ботов по официальной документации и по примерам. В основном все примеры были написаны на Python. Поэтому не долго думая, модно искать способы запуска Python сервера на Android. Но оценив время на изучение Python и не найдя ничего подходящего для запуска сервера, занялся поиском альтернатив и наткнулся на несколько библиотек на Java для написания Telegram ботов. В итоге остановился на проекте от Pengrad: java-telegram-bot-api.

Данная библиотека позволяла, на тот момент, инициализировать бота и получать-отправлять сообщения, что мне было и нужно. Добавив библиотеку в свой проект, я реализовал простой сервис, который запускал в фоновом потоке цикл по получению сообщений из Telegram и их обработке. Предварительно необходимо было зарегистрировать нового бота через родительский бот @Botfather и получить его токен. Подробнее о создании бота по ссылке.

Для того, чтобы сервис не убивался системой, когда устройство находится с выключенным экраном, при запуске сервиса, устанавливался WakeLock.

Приведу в пример функцию, позволяющую получать последние сообщения и отправлять их на обработку:

private void getUpdates(final TelegramBot bot)

Позже, в целях безопасности, я добавил возможность привязки бота к разрешенным Telegram-аккаунтам и возможность запрета выполнения

определенных команд для заданных пользователей.

Добавив несколько команд для бота, такие как: отправка, чтение СМС, просмотр пропущенных звонков, информация о батарее, определение местоположения и др., я опубликовал приложение в Google Play, создал темы на нескольких форумах, стал ждать комментарии и отзывы.

В основном отзывы были хорошие, но вскрылась проблема большого расхода батареи, что, как вы могли догадаться, было связано с WakeLock и постоянной активностью сервиса.Немного погуглив, решил периодически запускать сервис через AlarmManager, затем после получения сообщений и ответа на них сервис останавливать.

Это немного помогло, но появилась другая проблема, AlarmManager некорректно работал на некоторых китайских устройствах. И поэтому бот иногда не просыпался после нескольких часов, проведенных в состоянии сна. Изучая официальную документацию, я читал о том, что Long Polling это не единственная возможность получения сообщений, сообщения еще можно было получать используя Webhook.

Получение сообщений через Webhook

Я зарегистрировался на Digital Ocean, создал VPS на Ubuntu, затем реализовал простейший http сервер на Java, использующий Spark Framework. На сервер можно делать запросы 2 типов: push (отправка пуш-уведомления через webhook) и ping.

Пуш-нотификации отправлялись с помощью Google Firebase.

2. Генерация SSL сертификата

Протестировав отправку пуш-уведомлений, я стал разбираться с тем, как настроить и запустить сервер с HTTPS, так как это одно из требований при получении сообщений из Telegram через webhook.

Бесплатный сертификат можно сгенерировать с помощью сервиса letsencrypt.org, но одним из ограничений является то, что указываемый хост при генерации сертификата не может быть ip адресом. Регистрировать

доменное имя я пока не хотел, тем более официальная документация Telegram Bot API разрешает использование самоподписанных сертификатов, поэтому я стал разбираться, как создать свой сертификат.

После нескольких часов, проведенных в попытках и поисках, получился скрипт, позволяющий сгенерировать нужный сертификат.

create_cert.sh

После запуска скрипта, на выходе получаем два файла: keystore.jks — используется на сервере, public_cert.pem — используется при установке webhook в Android приложении.

Для того, чтобы запустить HTTPS на Spark Framework достаточно добавить 2 строки, одну указывающую порт (разрешенные порты для webhook: 443, 80, 88, 8443), другую, указывающую сгенерированный сертификат и пароль к нему:

```
port(8443);
secure("keystore.jks", "password", null, null);
```

Чтобы установить webhook для бота, необходимо добавить в андроид-приложение следующие строки:

```
SetWebhook setWebHook = new SetWebhook().url(WEBHOOK_URL + "/" + pushToken + "/" + secret).certificate(getCert(context));
BaseResponse res = bot.execute(setWebHook);
```

При регистрации webhook, в качестве URL указывается адрес webhook, затем передается пуш-токен, необходимый для отправки пуш-уведомлений и секретный ключ, генерируемый на устройстве, который я добавил для дополнительной проверки входящих уведомлений.

Функция чтения публичного сертификата из RAW ресурса:

```
private static byte[] getCert(Context context) throws IOException {
        return
IOUtils.toByteArray(context.getResources().openRawResource(R.raw.public_cert)
);
}
```

После модификации сервиса по обработке сообщений в Android приложении, бот стал расходовать батарею намного меньше, но и добавилась зависимость работы приложения от сервера пуш-нотификаций, что было необходимостью для стабильной работы приложения.

3. Автоматическое создание бота

После обновления механизма получения сообщений, осталась еще одна проблема, которая не позволяла пользоваться приложением некоторому проценту пользователей из-за сложности создания бота через BotFather. Поэтому я решил автоматизировать этот процесс.

В этом мне помогла библиотека tdlib от создателей Telegram. К сожалению, я нашел очень мало примеров использования этой библиотеки, но разобравшись в API, оказалось, что не так все сложно. В итоге удалось реализовать авторизацию в Telegram по номеру телефона, добавление @Botfather в список контактов и отправку и получение сообщений заданному контакту, а в конкретном случае, боту @Botfather.

Добавление новых возможностей

После решения первостепенных проблем с автономностью, я занялся добавлением новых команд.

В итоге были добавлены такие команды как: фото, запись видео, диктофон, скриншот экрана, управление плеером, запуск избранных приложений и т.д. Для удобного запуска команд, добавил Telegram-клавиатуру и разбил команды по категориям.

По просьбам пользователей, я также добавил возможность вызова команд Tasker и отправки сообщений из Tasker в Telegram.

После этого я задумался о том, что неплохо бы добавить внешний доступ из сторонних приложений для оправки сообщений в Telegram. Сообщения могут быть как текстовыми, так и включать в себя аудио, видео, местоположение по координатам. В итоге, я написал библиотеку, которую можно добавить в свой проект.