

## 6.5. Аутентификация пользователей

Сайт: [Samsung Innovation Campus](https://innovationcampus.ru)  
Курс: Мобильная разработка на Kotlin  
Книга: 6.5. Аутентификация пользователей

Напечатано.: Murad Rezvan  
Дата: понедельник, 3 июня 2024, 19:16

## Оглавление

1. 6.5.1 Аутентификация и авторизация.
2. 6.5.2 Протокол OAuth2\*
3. 6.5.3 Авторизация VK
4. 6.5.4 Авторизация Google
5. 6.5.5 Авторизация Firebase

## 1. 6.5.1 Аутентификация и авторизация.

Как только было реализовано клиент-серверное приложение - сразу же возникает проблема ограничения или регулирования доступа к информации на сервере. С точки зрения теории информационной безопасности это регулирование распадается на два процесса:

- Аутентификация пользователя - это процесс подтверждения того кем является пользователь. Подтверждение основывается на знании пользователя определенной информации - например логина и пароля. Также аутентификация может проводиться на основании считывания каких либо параметров - например биометрии или подписанного сертификата. Зачастую, в результате успешного прохождения процедуры аутентификации пользователь (клиентское ПО) получает известный серверу аутентификатор (токен, ключ), который пользователь при дальнейшей работе с сервером предъявляет при запросах. Этот аутентификатор может быть неизменным от сеанса к сеансу - как например в случае с http basic authentication, а может и меняться - как например в случае с Kerberos. В REST архитектуре, поскольку в ней нет сохранения состояний между вызовами, аутентификатор должен предъявляться с каждым запросом к серверу. В любом случае, часть системы отвечающая за аутентификацию должна иметь сохраненный набор аутентификационной информации (например логинов и паролей) для всех пользователей, которым разрешен доступ в систему и на выходе выдавать клиенту и серверу аутентификатор, по предъявлении которого сервер предоставляет доступ.
- Авторизация - это процесс разграничения доступа к информации в рамках системы. Здесь определяется кто из пользователей что может делать с информацией и с какой именно. Здесь возможно несколько вариантов:
  - избирательная модель управления доступом (DAC)
  - модель управление доступом на основе ролей (RBAC)
  - мандатное управление доступом (MAC)

OAuth2 - современный протокол аутентификации и авторизации работающий поверх HTTP, позволяющий выдать сервису или приложению права на доступ к ресурсам пользователя на целевом сервисе. Протокол избавляет от необходимости предоставлять приложению логин и пароль, а также позволяет выдавать ограниченный набор прав.

### Аутентификация пользователя и приложения.

Рассмотрим ситуацию когда пользователю нужно подключиться к онлайн серверу "напрямую" - при помощи веб браузера. При этом пользователь предъявляет сервису свою аутентифицирующую информацию - например логин и пароль. В случае успешной аутентификации сервис возвращает пользователю аутентификатор, который используется при дальнейшем HTTP обмене с веб-сервисом. После завершения работы с сервисом или разлогина, сервис аннулирует аутентификатор, и дальнейший обмен от имени текущего пользователя становится невозможным. Таким образом, производится только аутентификация пользователя.

Однако, в ситуации когда на мобильном устройстве работает приложение, которое от имени пользователя аутентифицируется на веб-сервисе возникает необходимость авторизации самого приложения на веб-сервисе. Это необходимо по многим причинам, например производитель приложения часто третье лицо, а вводимый логин и пароль пользователя приложение может и запомнить, поэтому необходима возможность ограничить операции на сервисе, которые может осуществлять приложение, либо вовсе отозвать разрешение на доступ к сервису.

Для реализации этого алгоритма все современные публичные соцсети и сервисы, такие как VK, Google, Yandex, Facebook, Twitter, Github и т.д. используют OAuth2.

Использование OAuth2 подходит для:

- Аутентификация на онлайн сервисе от имени пользователя.
- Обработка ошибок аутентификации.
- Получения разрешения от пользователя для доступа к онлайн службе с помощью своей учетной записи (через -агента-).

Таким образом в связи с вышесказанным становится ясно, что для реализации аутентификации в разрабатываемом приложении, основанной на использовании какого либо публичного сервиса, необходимо зарегистрировать приложение на этом сервисе. Происходит это по разному на разных сервисах, и в следующих частях лекции буду соответствующие примеры, но общий принцип регистрации похож. Нужно предоставить следующие сведения о регистрируемом приложении:

- тип приложения - Android (иногда Standalone, и т.п.)
- название - обычно просто название вашего проекта
- имя пакета - это значение после ключевого слова package, например

```
package ru.samsung.itschool.firebaseauth
```

- иногда имя класса активности
- отпечаток(криптохеш) SHA1 вашего ключа, важный момент есть ключи debug и release

Рассмотрим последний пункт более подробно. SHA1 отпечаток можно получить разными способами. Этот процесс описан на множестве источников - например у Google [Authenticating Your Client](#). В рассматриваемом примере это делалось следующим образом (см рис):

- в окне Android Studio открыть окно terminal
- в открывшемся окне терминала нужно набрать в ОС Linux

```
./gradlew signingReport
```

а для ОС windows

```
gradlew signingReport
```



```
Terminal: Local x +
d.yacenko@notebook:~/AndroidStudioProjects/FirebaseAuth> ./gradlew signingReport
Starting a Gradle Daemon (subsequent builds will be faster)
WARNING:: Please remove usages of `jcenter()` Maven repository from your build scripts and migrate your
This repository is deprecated and it will be shut down in the future.
See http://developer.android.com/r/tools/jcenter-end-of-service for more information.
Currently detected usages in: root project 'FirebaseAuth', project ':app'

> Task :app:signingReport
Variant: debug
Config: debug
Store: /home/d.yacenko/.android/debug.keystore
Alias: AndroidDebugKey
MD5: 7A:69:1C:39:88:2A:03:30:6A:0C:C9:EF:F4:8E:37:0D
SHA1: 48:04:5E:.....F7:AB:94:32
SHA-256: 8D:D4:A4:43:B4:3F:AC:E4:BB:4A:70:4B:73:DC:69:C2:93:FB:A9:CC:B4:CC:E2:6D:E8:8D:83:4A:8C:F8:73:2
Valid until: понедельник, 26 декабря 2050 г.
```

Важно понимать, что регистрация на ресурсах приложения происходит по отпечатку SHA1 приватного ключа из хранилища ключей debug или release, которые находятся на компьютере пользователя в его домашней папке - например `.android/debug.keystore`. Очень важно не потерять содержимое этой папки, т.к. в этом случае будут сгенерированы новые отпечатки, и новая сборка приложения не сможет авторизоваться на сервере!

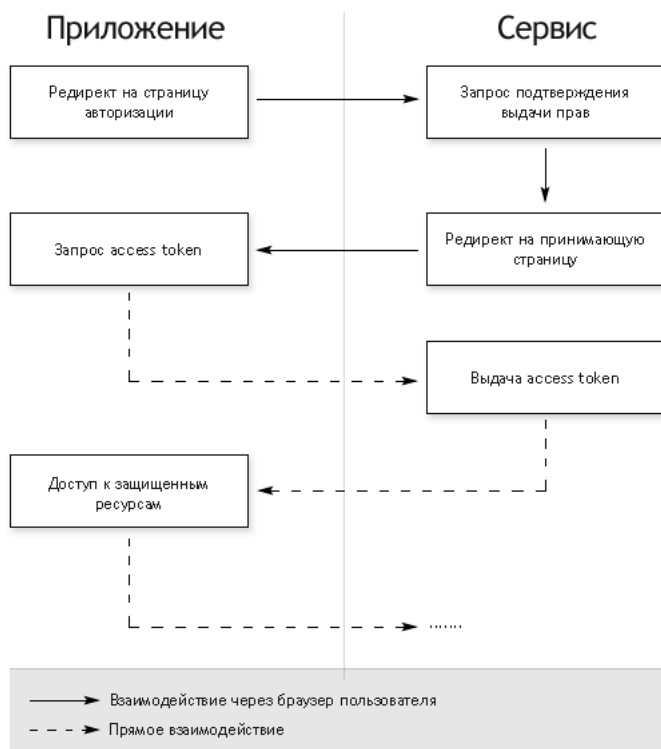
## 2. 6.5.2 Протокол OAuth2\*

OAuth2 - это протокол взаимодействия между участниками процесса аутентификации/авторизации. Протокол построен поверх протокола прикладного уровня - HTTP. Полное описание протокола можно прочитать в документе IETF - RFC6750 . Протокол имеет несколько сильно отличающихся режимов работы:

- авторизация для приложений, имеющих серверную часть [документация](#),
- авторизация полностью клиентских приложений [документация](#),
- авторизация по логину и паролю [документация](#),
- восстановление предыдущей авторизации [документация](#)

### Первый вариант авторизации - клиент-сервер.

1. При начале работы клиента, открывается браузер, в котором происходит переход на страницу авторизации,
2. На открывшейся странице возможно потребуется аутентификация, после чего у пользователя запрашивается подтверждение выдачи прав,
3. В случае согласия пользователя, браузер редиректится на URL, указанный как параметр **redirect\_uri** при открытии страницы авторизации (1), с добавлением в GET-параметры специального ключа — authorization code,
4. Сервер приложения выполняет POST-запрос к серверу авторизации с полученным authorization code в качестве параметра. В результате этого запроса, серверу приложения возвращается текущий access token.



### Второй вариант - отдельное приложение.

1. Открытие встроенного браузера со страницей авторизации
2. У пользователя запрашивается подтверждение выдачи прав
3. После подтверждения пользователем, браузер по коду 302 редиректится на страницу-заглушку во фрагменте (после символа #) URL которой добавляется access token
4. Приложение перехватывает редирект и получает access token из адреса страницы

Этот вариант требует поднятия в приложении окна браузера, но не требует серверной части и дополнительного вызова сервер-сервер для обмена authorization code на access token.



В качестве демонстрации работы протокола рассмотрим ручной обмен запросами в рамках протокола по второму варианту работы.

## Демонстрация обмена по протоколу OAuth2.

В начале нужно отправить через браузер GET запрос к серверу авторизации. В рассматриваемом примере это соцсеть VK. В рассматриваемом примере это

```
https://oauth.vk.com/authorize?client\_id=7855102&redirect\_uri=https://oauth.vk.com/blank.html&response\_type=token
```

Где **client\_id** - номер полученный при регистрации приложения на сервере (см следующую главу), а **redirect\_uri** - специальный URL-заглушка. В случае потребности соцсеть аутентифицирует пользователя. Однако если вы уже были аутентифицированы в окне браузера, то повторная аутентификация не потребует. В соответствии с протоколом, после отправки запроса будет получен ответ с кодом 302 и **location** который редиректит браузер на промежуточный URL. А оттуда вторым редиректом на **location** :

```
https://oauth.vk.com/blank.html#access\_token=edae146a20b23cc77bf016a9b2d9ead35a830ad3dcbed54785adf384aaf231fdb20babf222609f5c4831a&expires\_in=86400&user\_id=197324579
```

OAuth Blank

oauth.vk.com/blank.html#access\_token=edae146a20b23cc77bf016a9b2d9ead35a830ad3dcbed54785adf384aaf231fdb20babf222609f5c4

One Shot Learning... armashoteless MDev Kotlin: Пеко... IT Школа Samsun... дела Simplest version [...] Train\_Net.Ipynb - C... SQL и пар

Пожалуйста, **не копируйте** данные из адресной строки для сторонних сайтов. Таким образом Вы можете **потерять доступ** к Вашему аккаунту.

Elements Console Sources Network Performance Memory Application

Filter Hide data URLs All XHR JS CSS Img Media Font Doc WS

20 ms 40 ms 60 ms 80 ms 100 ms 120 ms 140 ms 160 ms 180 ms

Name

authorize?client\_id=7855102&redirect\_uri=https://o...  
?act=grant\_access&client\_id=7855102&settings=0&...  
auth\_redirect?app\_id=7855102&authorize\_url=https...  
blank.html

Headers Preview Response Init

General

Request URL: https://oauth.vk.com/a  
Request Method: GET  
Status Code: 302  
Remote Address: 87.240.129.135:443  
Referrer Policy: strict-origin-when-c

Response Headers

cache-control: no-store  
content-encoding: gzip  
content-length: 20  
content-type: text/html; charset=win  
date: Mon, 14 Jun 2021 15:04:50 GMT  
location: https://login.vk.com/?act=a092d098d7178&hash=1623683090\_4ef0  
server: kittenx  
set-cookie: remixir=DELETED; expires  
set-cookie: remixlhk=DELETED; expire  
strict-transport-security: max-age=15768  
x-powered-by: KPHP/7.4.107504

Request Headers

:authority: oauth.vk.com  
:method: GET  
:path: /authorize?client\_id=7855102&

4 requests 1.8 kB transferred 1.1 kB resources F

Вот из этого **location** нужно извлечь **access\_token**. Теперь в последующих запросах к Web API сервера нужно передавать полученные токен. Например GET запрос информации о пользователе:

```
curl 'https://api.vk.com/method/users.get?user_ids=197324579&fields=bdate&access_token=edae146a20b23cc77bf016a9b2d9ead35a830ad3dcbed54785adf384aaf231fdb20babf222609f5c4831a&v=5.131'
{"response":{"first_name":"Дмитрий","id":197324579,"last_name":"Яценко","can_access_closed":true,"is_closed":false,"bdate":"17.9.1970"}}
```

Дополнительную информацию по OAuth2 и WebAPI VK можно по ссылкам

- [https://vk.com/dev/oauth\\_dialog](https://vk.com/dev/oauth_dialog)
- <https://vk.com/dev/users.get>
- <https://vk.com/dev/account.getProfileInfo>
- [https://vk.com/dev/api\\_requests](https://vk.com/dev/api_requests)

Далее приведен общий протокол обмена с VK:

```
# запрос в браузере
# ID приложения: 7855102
GET -> https://oauth.vk.com/authorize?client\_id=7855102&redirect\_uri=https://oauth.vk.com/blank.html&response\_type=token
Status Code: 302
location: https://login.vk.com/?act=grant\_access&client\_id=7855102&settings=0&response\_type=token&group\_ids=&token\_type=0&v=&display=page&ip\_h=5dee5fa1b795bbb30d&hash=1623513398\_d77f36343e524fea36&https=1&state=&redirect\_uri=https%3A%2F%2Foauth.vk.com%2Fblank.html
GET -> https://login.vk.com/?act=grant\_access&client\_id=7855102&settings=0&response\_type=token&group\_ids=&token\_type=0&v=&display=page&ip\_h=5dee5fa1b795bbb30d&hash=1623513398\_d77f36343e524fea36&https=1&state=&redirect\_uri=https%3A%2F%2Foauth.vk.com%2Fblank.html
Status Code: 302
location: https://oauth.vk.com/auth\_redirect?app\_id=7855102&authorize\_url=https%253A%252F%252Foauth.vk.com%252Fblank.html%2523access\_token%253Dd3e8bc3fc5355f2c164815985bd3863ed69988f32586ef1537283715c5c7374cc6f84cc541fb9920377c9%2526expires\_in%253D86400%2526user\_id%253D197324579&redirect\_hash=78376e08b80554b07f

# Полученный access_token: d3e8bc3fc5355f2c164815985bd3863ed69988f32586ef1537283715c5c7374cc6f84cc541fb9920377c9

# Запрос из CLI
d.yacenko@notebook:~> curl 'https://api.vk.com/method/users.get?user_ids=197324579&fields=bdate&access_token=d3e8bc3fc5355f2c164815985bd3863ed69988f32586ef1537283715c5c7374cc6f84cc541fb9920377c9&v=5.131'
{"response":{"first_name":"Дмитрий","id":197324579,"last_name":"Яценко","can_access_closed":true,"is_closed":false,"bdate":"17.9.1970"}}
```

Однако работа непосредственно по протоколу довольно громоздка, поэтому большинство социальных сетей, облачных платформ и других серверов авторизации OAuth2 предоставляют свои библиотеки для Android, упрощающий процесс аутентификации/авторизации. В следующих главах будут рассмотрены именно такие примеры.



### 3. 6.5.3 Авторизация VK

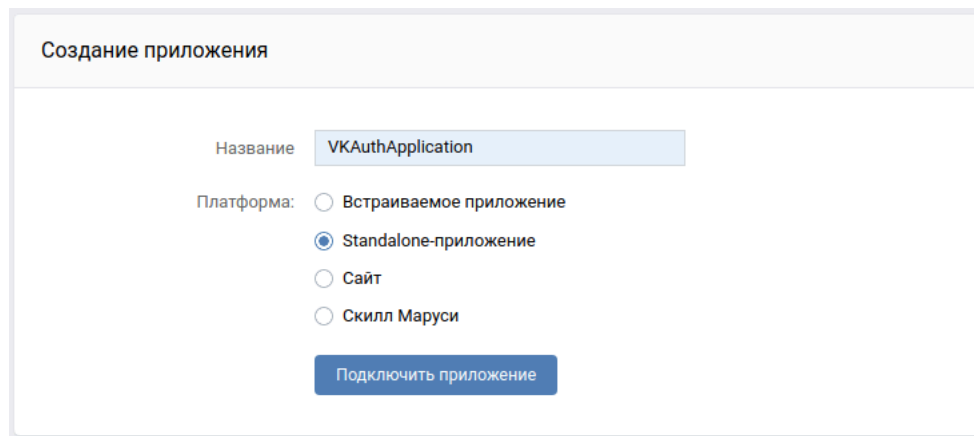
В этом разделе рассмотрим авторизацию с использованием VK SDK. В качестве примера реализуем приложение, с двумя кнопками Login для аутентификации на сервисе VK, а Logout для отключения от сервиса (аннулирования аутентификатора).

Рассматриваемые далее примеры входят в практическую работу, исходный код которой можно скачать по [ссылке](#)

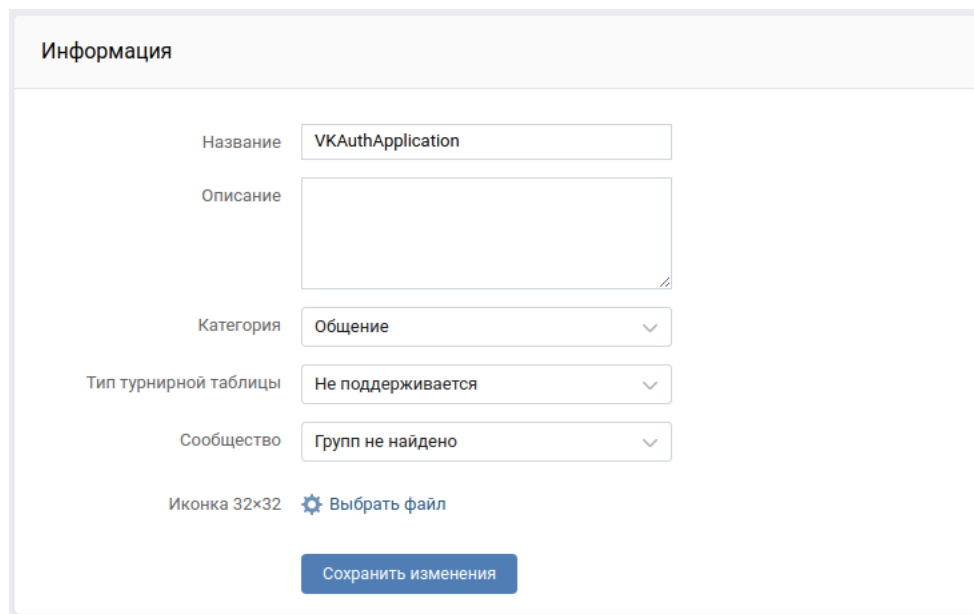
Однако, важно отметить, что приведенный код не заработает, т.к. при регистрации приложения указывался отпечаток сертификата преподавателя, а значит он не совпадет с отпечатком вашего сертификата, и как следствие приложение **не сможет работать с сервером авторизации** ! Поэтому приведенный код можно рассматривать лишь в виде теоретического пособия. Для того чтобы приложение заработало, нужно зарегистрировать его от имени студента на сервере авторизации с указанием отпечатка сертификата студента, и соответствующим образом перенастроить проект.

Посмотреть документацию по настройке подключения приложения расположена на платформе соцсети в статье [Android SDK](#) . Но в общем смысле, также как и в случае с другими платформами, сначала необходимо зарегистрировать приложение заполнив в консоли управления - "Мои приложения" требуемые формы. На первой форме нужно указать имя проекта и указать тип приложения Standalone.

Несмотря что VK, как и большинство сервисов VK поддерживает протокол авторизации OAuth, в рассматриваемом примере будет использоваться библиотека от VK, которая реализует все операции протокола, и использование которой минимизирует количество кода для осуществления аутентификации в VK.



В форме заполнения информации можно заполнить только поле название



В разделе настройки нужно поле состояние переключить - "приложение включено и видно всем", а в разделе настройки SDK нужно заполнить три поля - пакет, имя класса и отпечаток ключа.

### Настройки

ID приложения

7846657

Защищённый ключ

\*\*\*\*\*

Сервисный ключ доступа

\*\*\*\*\*

Состояние

Приложение включено и видно всем

Первый запрос к API

Установка приложения

Не требуется

Open API

Отключён

Push-уведомления

Не подключены

Настройки SDK

App Bundle ID для iOS

орaque — не перекрывает уведомления и

App ID для iOS

орaque — не перекрывает уведомления и

Название пакета для Android

ru.samsung.itschool.vkauthapplication

Main activity для Android

MainActivity

Отпечаток сертификата для Android

48:04:5C...AB:94:32

Добавить ещё

Windows App ID

Сохранить изменения

После настройки приложения на сайте VK можно переходить к созданию непосредственно приложения. Предварительно подготовим приложение. Добавим в приложение библиотеку VK - в файл **build.gradle(app)** в раздел **dependencies**:

```
implementation 'com.vk:androidsdk:2.1.1'
```

В том же файле, в разделе **android** включаем особенность **viewBinding** (убираем **findViewById**)

```
buildFeatures {
    viewBinding = true
}
```

Рассматривать отдельно макет интерфейса не будем, отметим только что там описаны две кнопки

**bt\_login** и **bt\_logout** и текстовый виджет **tv\_res**. Для аутентификации достаточно вызвать метод **VK.login()**. При этом будет в отдельном интенте (предоставляемом библиотекой) вводиться данные аутентификации пользователя и производиться обмен с сервером.

```
binding.btLogin.setOnClickListener({
    VK.login(this, arrayListOf(VKScope.WALL, VKScope.PHOTOS))
})
```

Для получения результатов аутентификации необходимо переопределить стандартный метод **onActivityResult()**. В этом методе необходимо вызвать метод **VK.onActivityResult()** передав в него полученный интент и объект колбэка (**VKAuthCallback**). Методы этого колбэка - **onLogin()** и **onLoginFailed()** и будут вызваны в случае успешной или не успешной аутентификации.

```

override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
    val callback = object : VKAuthCallback {
        override fun onLogin(token: VKAccessToken) {
            // User passed authorization
            binding.tvRes.text = "Login success. ID: " + token.userId
        }

        override fun onLoginFailed(errorCode: Int) {
            // User didn't pass authorization
            binding.tvRes.text = "Login fail"
        }
    }
    if (data == null || !VK.onActivityResult(requestCode, resultCode, data, callback)) {
        super.onActivityResult(requestCode, resultCode, data)
    }
}

```

Для отмены авторизации достаточно вызвать функцию **VK.logout()** :

```

binding.btLogoff.setOnClickListener({
    VK.logout()
    if (!VK.isLoggedIn()) binding.tvRes.text = "Not login to VK"
})

```

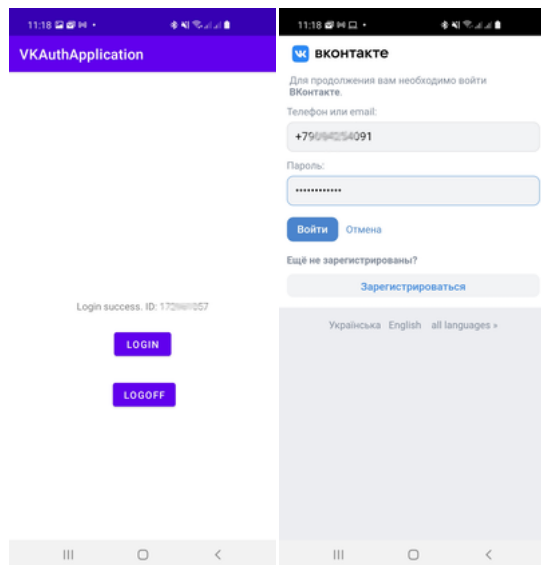
Для определения текущего статуса аутентификации можно использовать функцию **VK.isLoggedIn()** :

```

if (!VK.isLoggedIn()) binding.tvRes.text = "Not login to VK"
else binding.tvRes.text = "Now logged in to VK"

```

В итоге, используя библиотеку VK, можно разработать довольно компактное приложение осуществляющее аутентификацию с использованием сервиса соцсети. Скриншоты демонстрации процесса аутентификации приведены ниже.



# 4. 6.5.4 Авторизация Google

В этом разделе рассмотрим авторизацию с использованием Google SDK. В качестве примера реализуем приложение, с двумя кнопкой Login для аутентификации на платформе Google. После подключения необходимо вывести информацию о пользователе.

Рассматриваемые далее примеры входят в практическую работу, исходный код которой можно скачать по [ссылке](#)

Однако, важно отметить, что приведенный код не заработает, т.к. при регистрации приложения указывался отпечаток сертификата преподавателя, а значит он не совпадет с отпечатком вашего сертификата, и как следствие приложение **не сможет работать с сервером авторизации** ! Поэтому приведенный код можно рассматривать лишь в виде теоретического пособия. Для того чтобы приложение заработало, нужно зарегистрировать его от имени студента на сервере авторизации с указанием отпечатка сертификата студента, и соответствующим образом перенастроить проект.

Информацию по настройке приложения в консоли управления Google можно найти в официальной документации:

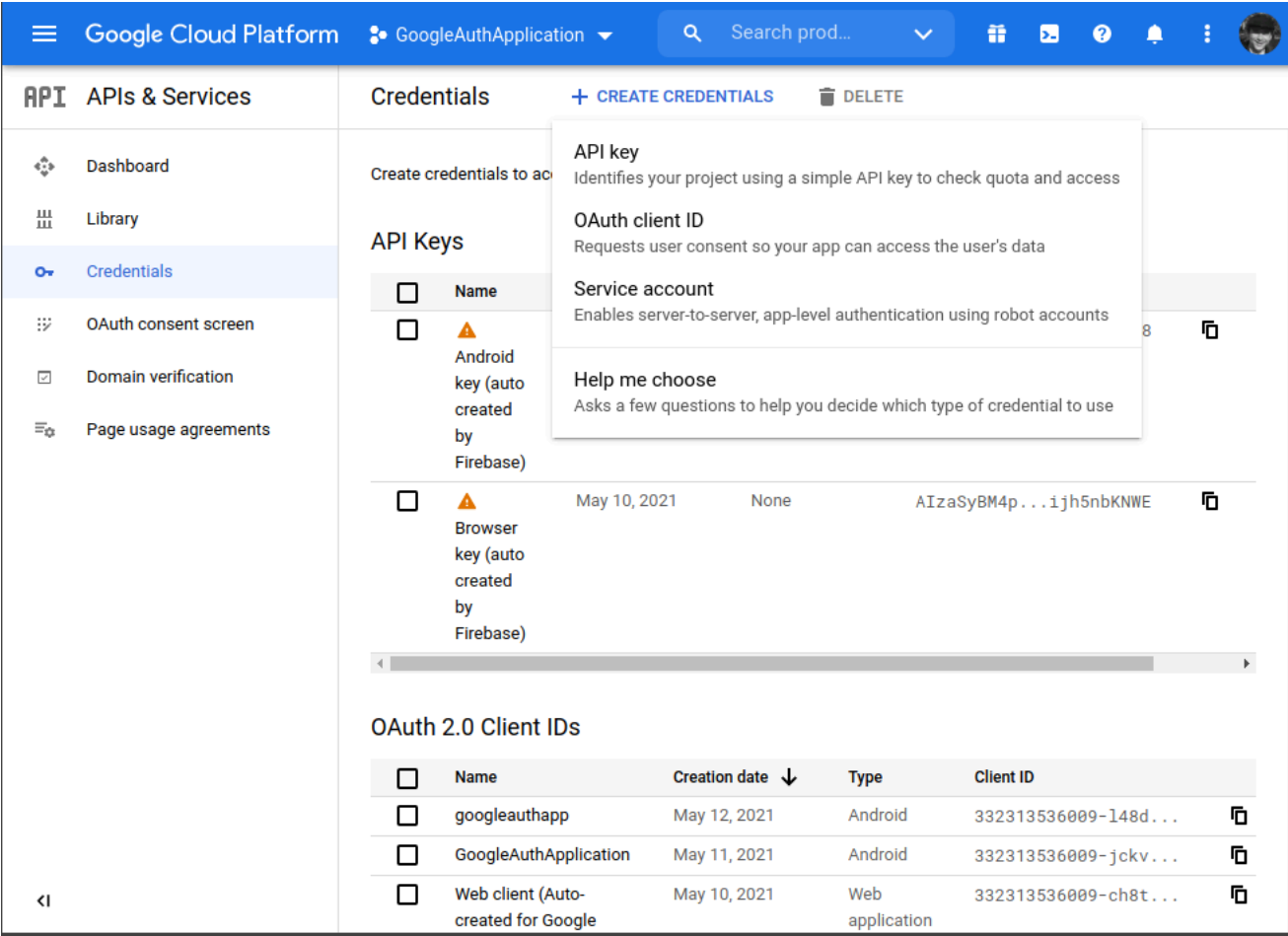
- [Authenticating Your Client](#)
- [Start Integrating Google Sign-In into Your Android App](#)
- [Integrating Google Sign-In into Your Android App](#)
- [Getting Profile Information](#)

Вначале необходимо создать и настроить проект и приложение по кнопке в статье [Start Integrating Google Sign-In into Your Android App](#) - компактная форма или в консоли управления [GoogleCloudPlatform](#).

В консоли управления сначала необходимо создать проект, после чего в разделе [API&Services](#) в созданный настроить "OAuth consent screen" для проекта. Тут нужно будет добавить довольно много информации для экрана представления приложения пользователю:

- User Type - external,
- имя приложения,
- e-mail поддержки,
- e-mail разработчика.

Далее необходимо добавить в проект Credentials - Create credentials - OAuth client ID:



Здесь необходимо добавить следующие данные:

- Application type - Android,

- Name название приложения,
- Package name - имя пакета приложения, значение после оператора **package** ,
- SHA-1 certificate fingerprint - отпечаток ключа.

**APIs & Services** | **Create OAuth client ID**

A client ID is used to identify a single app to Google's OAuth servers. If your app runs on multiple platforms, each will need its own client ID. See [Setting up OAuth 2.0](#) for more information.

**Application type \***  
Android

[Learn more](#) about OAuth client types

**Name \***  
GoogleAuthApplication

The name of your OAuth 2.0 client. This name is only used to identify the client in the console and will not be shown to end users.

**Package name \***  
ru.samsung.itschool.googleauthapplication

From your AndroidManifest.xml file.

**SHA-1 certificate fingerprint \***  
48:04:5E:A...AB:94:32

SHA-1 signing certificate fingerprint restricts usage to your Android apps. [Learn more](#)

Use this command to get the fingerprint.

```
$ keytool -keystore path-to-debug-or-production-keystore -list -v
```

**CREATE** **CANCEL**

После настройки проекта в консоли GoogleCloudPlatform, можно настраивать проект. В манифест добавить разрешение работы с интернет.

```
<uses-permission android:name="android.permission.INTERNET" />
```

В файл **build.gradle(app)** в раздел **dependencies** библиотеку **play-services-auth** :

```
implementation 'com.google.android.gms:play-services-auth:15.0.1'
```

Для запуска процесса аутентификации необходимо вызвать интент - **signInIntent**, в котором будет осуществлен процесс аутентификации на платформе Google.

```
val gso = GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
    .requestEmail()
    .build()
val mGoogleSignInClient = GoogleSignIn.getClient(this, gso);
binding.activityButtonSignIn.setOnClickListener({
    val signInIntent = mGoogleSignInClient.signInIntent
    startActivityForResult(signInIntent, Companion.RC_AUTH_CODE)
})
```

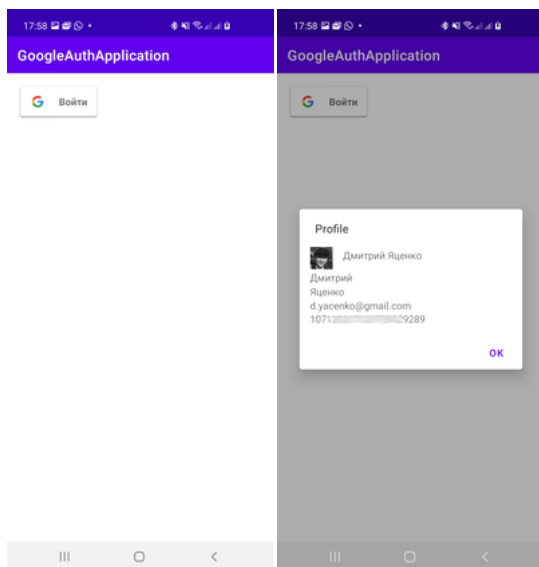
Для получения результатов процесса аутентификации необходимо использовать стандартный способ получения информации от вызванного интента - переопределение метода **onActivityResult()** :

```
override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
    super.onActivityResult(requestCode, resultCode, data)
    if (requestCode == Companion.RC_AUTH_CODE) {
        val task: Task<GoogleSignInAccount> = GoogleSignIn.getSignedInAccountFromIntent(data);
        handleSignInResult(task);
    }
}
```

При этом в полученном объекте **task** находится вся информация о подключении к учетной записи. Получить ее можно например следующим образом:

```
GoogleSignInAccount acct = task.getResult(ApiException.class);  
val email = acct.email  
val name = acct.displayName  
val id = acct.id
```

Ниже представлен пример работающего приложения с аутентификацией посредством платформы Google.



## 5. 6.5.5 Авторизация Firebase

В этом разделе рассмотрим авторизацию с использованием Firebase. В качестве примера реализуем приложение, с четырьмя кнопками Register, Login, Logoff, Info для регистрации, аутентификации, а также получения информации о пользователе на платформе Firebase.

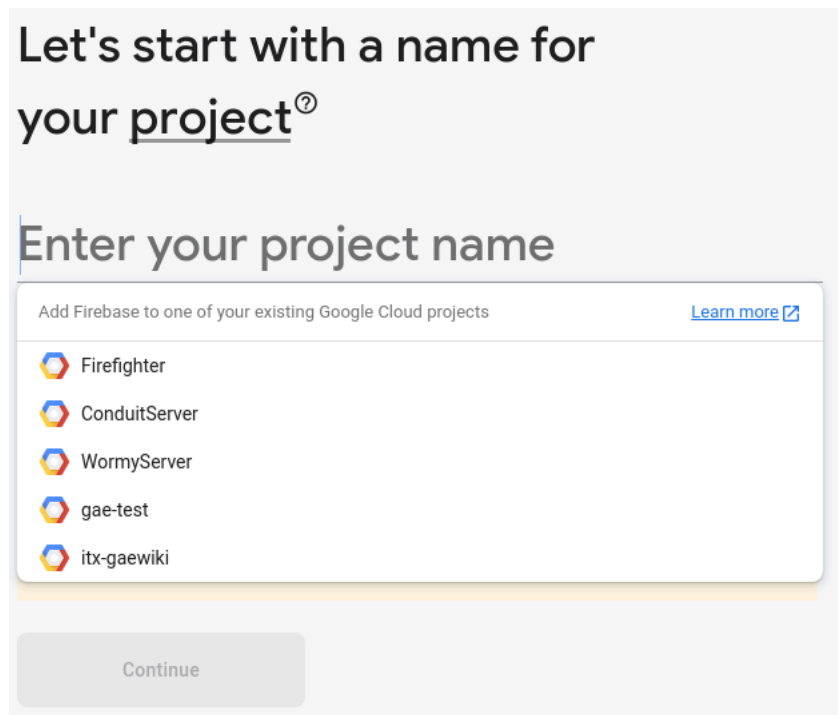
Рассматриваемые далее примеры входят в практическую работу, исходный код которой можно скачать по [ссылке](#)

Однако, важно отметить, что приведенный код не заработает, т.к. при регистрации приложения указывался отпечаток сертификата преподавателя, а значит он не совпадет с отпечатком вашего сертификата, и как следствие приложение **не сможет работать с сервером авторизации** ! Поэтому приведенный код можно рассматривать лишь в виде теоретического пособия. Для того чтобы приложение заработало, нужно зарегистрировать его от имени студента на сервере авторизации с указанием отпечатка сертификата студента, и соответствующим образом перенастроить проект.

Информацию о настройке приложения в консоли можно получить в документации:

- [Add Firebase to your Android project](#)
- [Get Started with Firebase Authentication on Android](#)

Как и в рассмотренных ранее случаях необходимо зарегистрировать проект и приложение на платформе Firebase. Для этого необходимо войти на сайт [консоли](#) и нажать кнопку add project, после чего в мастере как на рисунке ниже в ввести имя нового проекта в поле ввода "Enter you project name", или выбрать существующий в выпадающем списке. Нажать кнопку Continue.



Если не предусматривается сбор аналитики по использованию приложения, выключаем ее и продолжаем настройку:

# Google Analytics for your Firebase project

Google Analytics is a free and unlimited analytics solution that enables targeting, reporting, and more in Firebase Crashlytics, Cloud Messaging, In-App Messaging, Remote Config, A/B Testing, Predictions, and Cloud Functions.

Google Analytics enables:

- ✕ A/B testing ?
- ✕ User segmentation & targeting across Firebase products ?
- ✕ Predicting user behavior ?
- ✕ Crash-free users ?
- ✕ Event-based Cloud Functions triggers ?
- ✕ Free unlimited reporting ?

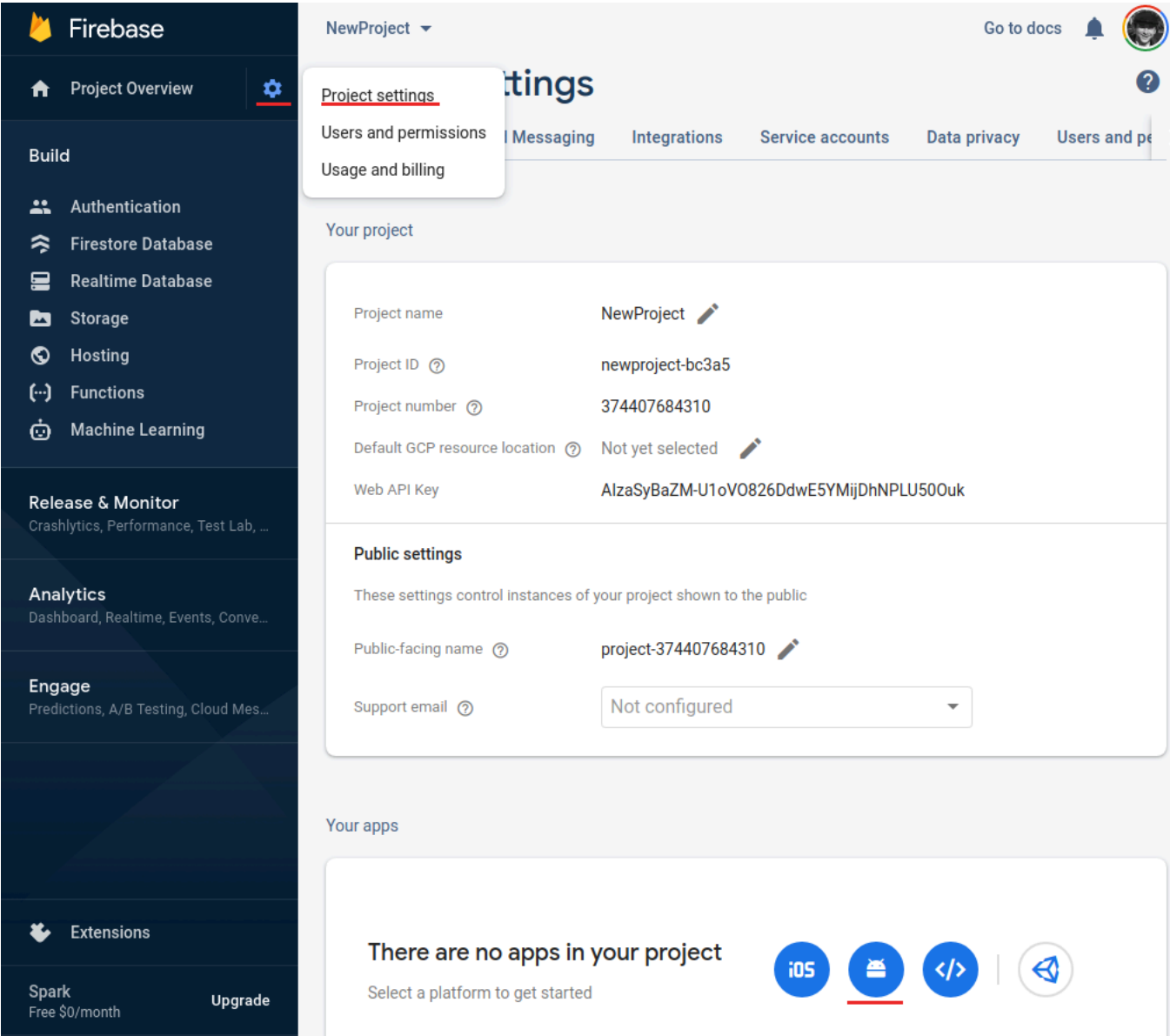
☐ Enable Google Analytics for this project  
Recommended

[Previous](#)

Create project

После завершения описания проекта, необходимо описать в нем создаваемое Android приложение. Для этого в "Project overview" необходимо перейти в раздел "settings" и в разделе "You apps" кликнуть по кнопке с логотипом Android:





В форме регистрации приложения необходимо ввести название проекта, название пакета, отпечаток ключа после чего нажать кнопку "Register app":

× **Add Firebase to your Android app**

1

Register app

Android package name ?

ru.samsung.itschool.firebaseauth

App nickname (optional) ?

NewProject

Debug signing certificate SHA-1 (optional) ?

48:04:.....A2:44:02:F7

Required for Dynamic Links, and Google Sign-In or phone number support in Auth. Edit SHA-1s in Settings.

Register app

2

Download config file

3

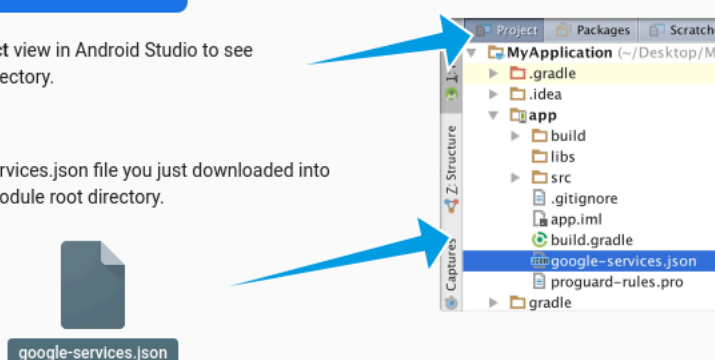
Add Firebase SDK

4

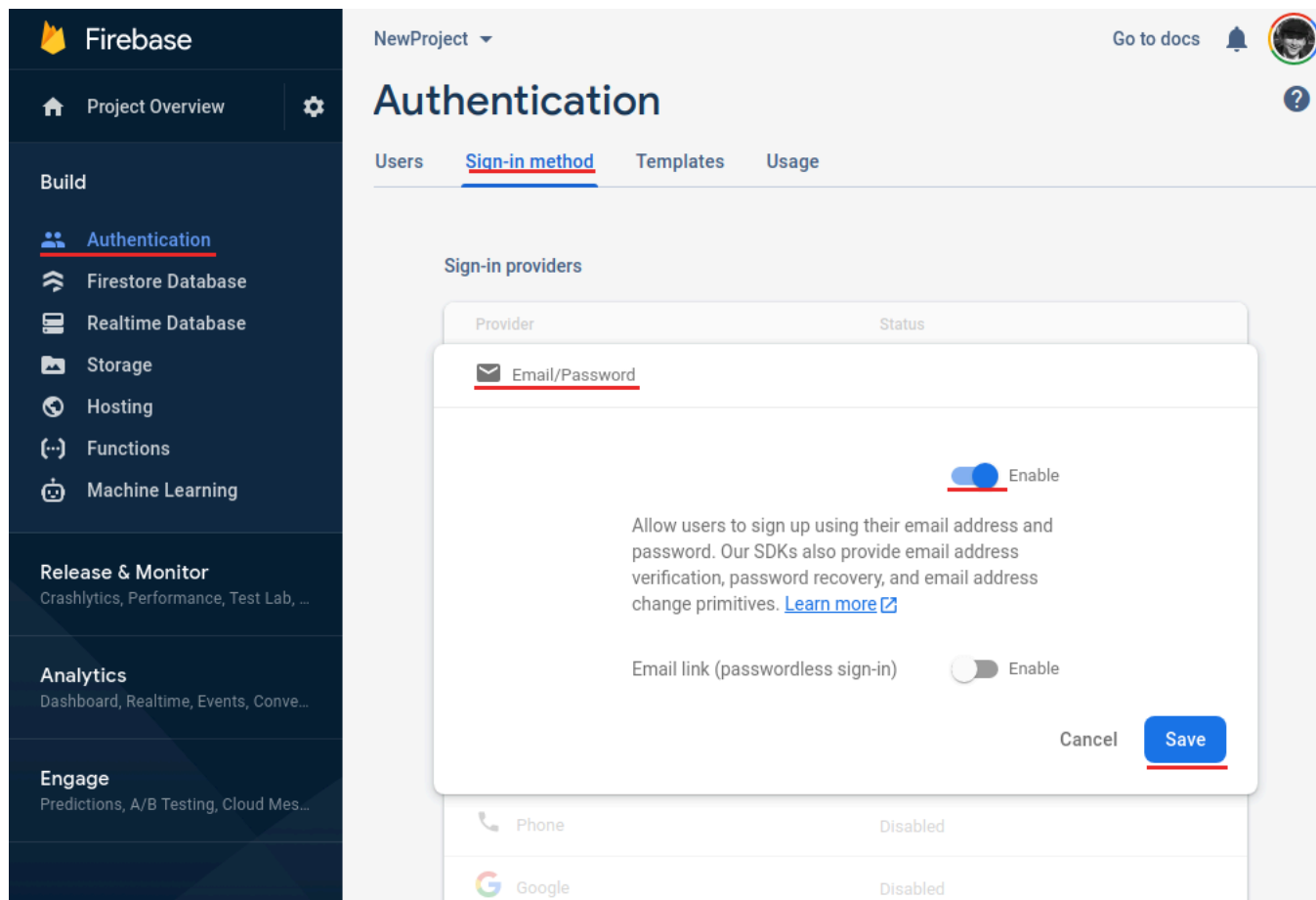
Next steps

На следующем шаге необходимо скачать файл **google-services.json** , который необходимо скопировать в папку проекта **/app**

## ✕ Add Firebase to your Android app

- ✓ Register app  
Android package name: ru.samsung.itschool.firebaseauth\_, App nickname: NewProject
- 2 Download config file  
Instructions for Android Studio below | [Unity](#) [C++](#)  
[Download google-services.json](#)  
Switch to the Project view in Android Studio to see your project root directory.  
Move the google-services.json file you just downloaded into your Android app module root directory.  
  
[Next](#)
- 3 Add Firebase SDK
- 4 Next steps

Далее необходимо завершить шаги регистрации приложения и настроить методы аутентификации. Для этого переходим в "Project overview" - "Authentication" - "Sign-in method" и в открывшемся списке вариантов аутентификации выбираем необходимый. В рассматриваемом примере это "Email/password"



Provider	Status
Email/Password	Enable
Phone	Disabled
Google	Disabled

На этом настройка приложения в консоли закончена и можно переходить к настройке приложения в Android studio. В файл **build.gradle(app)** в раздел **dependencies** добавим необходимые библиотеки:

```
implementation platform('com.google.firebase:firebase-bom:28.0.0')
implementation 'com.google.firebase:firebase-analytics-ktx'
implementation 'com.google.firebase:firebase-auth-ktx'
```

Кроме того в конец файла **build.gradle(app)** нужно добавить строки:

```
apply plugin: 'com.android.application'
apply plugin: 'com.google.gms.google-services'
```

В файле **build.gradle** в раздел **dependencies** необходимо добавить параметр **classpath** :

```
classpath 'com.google.gms:google-services:4.3.5'
```

На этом настройка Android проекта завершена. В коде, для работы с авторизацией необходимо проинициализировать объекты фреймворка:

```
FirebaseApp.initializeApp(this)
auth = Firebase.auth
```

После инициализации можно приступить непосредственно к операциям по авторизации. Рассмотрим, каким образом пользователь может зарегистрировать новый логин/пароль:

```
auth.createUserWithEmailAndPassword(email,password)
    .addOnCompleteListener(this) { task ->
        if (task.isSuccessful)
            toast("Register success")
        else
            toast("Register failed")
    }
```

Для реализации самого процесса аутентификации , в рассматриваемом пример воспользуемся функцией **signInWithEmailAndPassword()**

```
auth.signInWithEmailAndPassword(email,password)
    .addOnCompleteListener(this) { task ->
        if (task.isSuccessful)
            toast("Sign in success as " + auth.currentUser?.email)
        else
            toast("Sign in fail")
    }
```

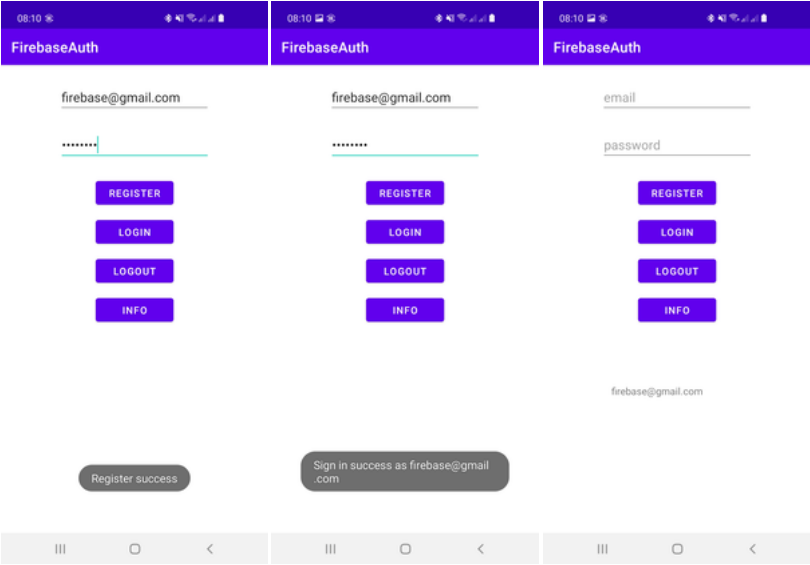
Для прекращения действия аутентификации необходимо использовать функцию **signOut()**

```
if (auth !=null) {
    auth?.signOut();
    toast("Sign out")
}
```

Для получения информации о пользователе можно извлечь эти данные из объекта **auth.currentUser** :

```
val user = auth.currentUser
user?.let {
    val name = user.displayName
    val mail = user.email
    val emailVerified = user.isEmailVerified
    val uid = user.uid
}
```

Ниже представлен пример работающего приложения с аутентификацией посредством платформы Firebase:



[Начать тур для пользователя на этой странице](#)