

本日の内容

- Firebase
- Firestore

今回は Firebase の機能の一部を使用してみます。

■ Firebase

そもそも Firebase とは？

「Web アプリやモバイルアプリ」の「バックエンド」と呼ばれるユーザには見えない裏側の処理の為の機能を提供する「クラウドサービス」です。



■ Firebase の特徴

Firebase と連携するアプリは複数の言語での開発が可能となっており、クロスプラットフォーム、マルチプラットフォームなアプリの開発をサポートします。

モバイルアプリの場合：モバイル SDK を呼び出すことで連携。Android,iOS をサポート

Web アプリの場合：SDK(JavaScript による API)を提供しており、ブラウザからアクセスが可能。

■ Firebase を使用するメリット

◆ 豊富な機能を活用出来る

アプリ開発に必要とされている様々な機能を備えています。

その為、firebase の機能を組み合わせることにより、開発の期間を短縮することが可能です

◆ サーバーの管理や保守の手間を省ける

バックエンド処理を firebase で行うことにより、サーバーの管理、保守が不要となります。

■FireBase の機能例(2022 年時点)

◆リアルタイムデータベース (Firebase Realtime Database)



クラウドホスト型 NoSQL データベースです。

このデータベースを利用することで、

アプリ間でデータをリアルタイムで保存・同期することができます。

活用例：リアルタイムチャットアプリ

◆NoSQL データベース (Cloud Firestore)



NoSQL のクラウドデータベースサービスです。

利用開始時には、無料で利用できる領域が用意されており、

それ以降は従量制での料金体系となります。

シンプルにアプリ間同期が取りたい場合は

「Firebase Realtime Database」

複雑なクエリの活用も視野に入る場合は

「Cloud Firestore」といった形で目的や規模に沿って選択します。

活用例：ToDo リスト

◆クラウドメッセージング (Firebase Cloud Messaging)



クロスプラットフォームで通知を無料送信できる機能です。

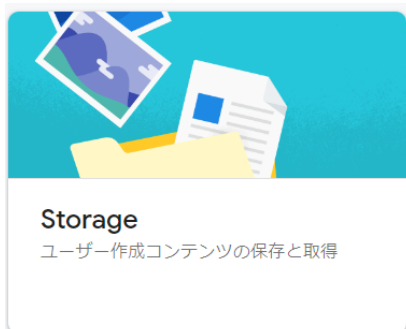
セグメントを使用して

メッセージを送るターゲットを設定することも可能です。

送信したメッセージはダッシュボードでモニタリングできるので、アプリマーケティングの分析に活躍します。

活用例：セールお知らせアプリ

◆データ保存 (Cloud Storage for Firebase)



「Cloud Storage for Firebase」はクラウドストレージサービスです。

Firebase を利用するモバイルアプリ、

Web アプリなどに向けてデータ格納領域(ストレージ)を提供します。

活用例：アプリ内で使用する画像の格納領域

◆認証 (Firebase Authentication)



ユーザーの初期登録や認証をアプリに簡単に実装できる機能です。
ログイン方法は、「FirebaseUI」に備わっている
「ドロップイン認証ソリューション」を利用する、
または「Firebase Authentication SDK」を使って
手動でアプリに統合するかを選択できます。
活用例：ログイン機能

等々…他にもありますのでご自身で調べてみましょう！

■FireBase のデメリット

◆大規模サービスでは扱いにくい

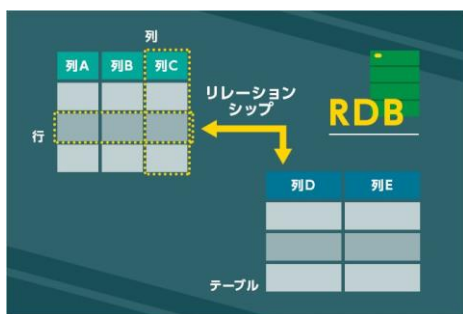
FireBase は NoSQL を活用している為、複雑なデータの管理に向いていません。
小規模な開発で効果を発揮します。

◆完全無料ではない

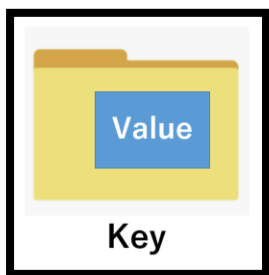
ある程度のアクセスまでは無料で使用出来ますが、
大量の通信や、膨大なデータを扱う場合は料金が追加で必要になります。
この点も大規模サービスに向いていない点です。

■RDB と NoSQL

RDB とは普段皆さんが MySQL で使用しているような表のように管理された DB です。
拡張性が低い分、SQL 文を活用し複雑なデータ管理を可能としています。



NoSQL とは、名前の通り SQL を使わない「キーバリュー型」の DB です。



今回は「Cloud Firestore」の機能を使用し、簡易的な ToDo リストの作成を行い使用方法を学びます。



■Cloud Firestore のファイル構造

コレクション

ドキュメントは、ドキュメントの単なるコンテナであるコレクションに存在します。たとえば、さまざまなユーザーを含む `users` コレクションを作成し、それぞれをドキュメントで表すことができます。

ユーザー

- アロブレイス**
`first : "Ada"`
`last : "Lovelace"`
`born : 1815`
- 中**
`first : "Alan"`
`last : "Turing"`
`born : 1912`

The diagram shows a yellow folder labeled "collection" containing several blue documents. One document is labeled "document" and contains the text "data".

collection がテーブル

document が主キー

data がカラム及び値

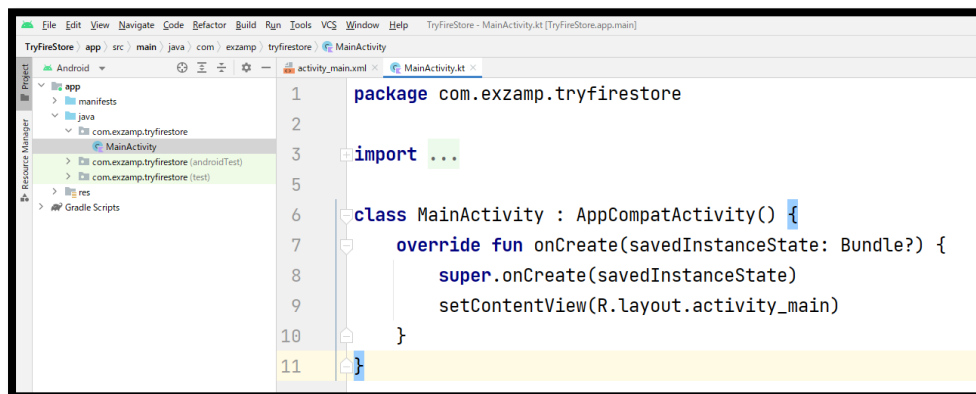
上記のようなイメージになります。

■事前に新規プロジェクトを作成してきましょう。

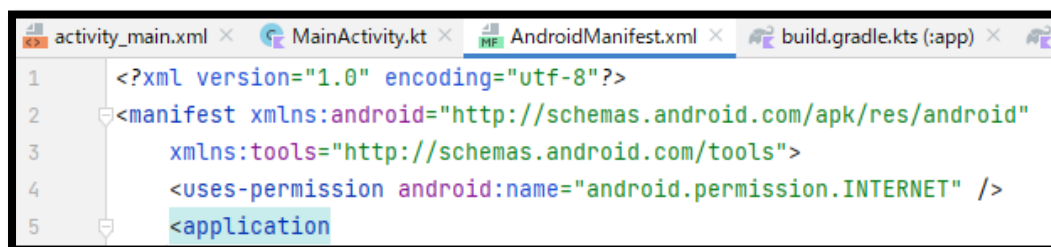
テンプレート: EmptyViewActivity

ファイル名: TryFirestore

言語: Kotlin



AndroidManifest.xml のネットワークのパーミッションも忘れずに



■Firebase へのアカウント登録

※今回の課題には、個人の Google アカウントが必要です。未作成の場合は、先に作成を行って下さい。

＜Google アカウントの作成＞

<https://support.google.com/accounts/answer/27441?hl=ja>

既存の受信可能なメールアドレスで新規登録可能。電話番号必要なし。

＜Firebase のサイト＞

<https://firebase.google.com/?hl=ja>

個人ごとに作成済みの Google アカウントでログインする。以下案内に沿って設定を進めていく。

「使ってみる」をクリック



■Firebase の設定

①「プロジェクトを追加」をクリックする



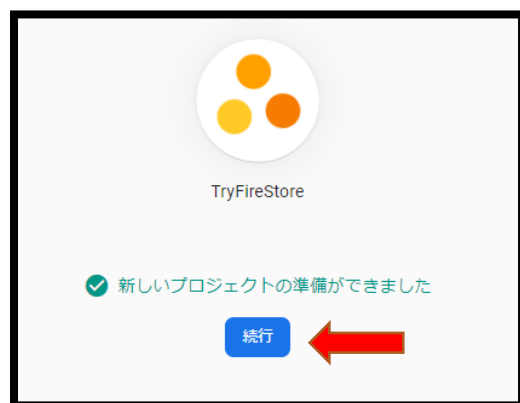
②プロジェクト名を入力して[続行]をクリックする。



③処理時間がかかるため、
「このプロジェクトで Google アナリティクスを有効にする」を
オフにして[プロジェクトを作成]をクリックする。



④続行をクリック



⑤ドROID君をクリック

Android で活用する為の設定を行ないます。



⑥作成済みプロジェクトのパッケージ名をセットして、[アプリを登録]をクリックする。

FireBase にアクセスする Android アプリのパッケージを指定します。

[illegible]

⑦google-services.json をダウンロード

×

Android アプリに Firebase を追加

✓

アプリの登録

Android パッケージ名: com.exzamp.tryfirestore

2

構成ファイルをダウンロードして追加する

Android Studio については下記参照 | [Unity](#) [C++](#)

📄 google-services.json をダウンロード

Android Studio の [Project] 表示に切り替え、プロジェクトのルートディレクトリを表示します。

ダウンロードした google-services.json ファイルをモジュール（アプリレベル）のルートディレクトリに移動します。

google-services.json

Project Packages Scratch

MyApplication (~\Desktop\My

gradle

idea

app

build

libs

src

.gitignore

app.iml

build.gradle

google-services.json

proguard-rules.pro

gradle

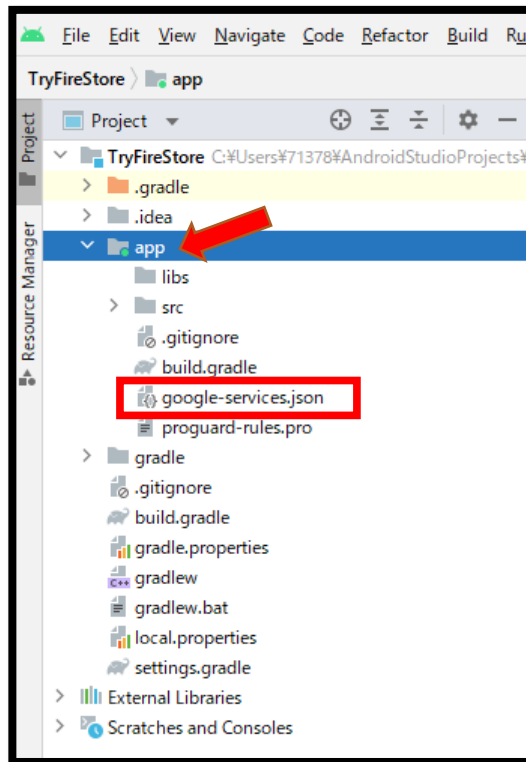
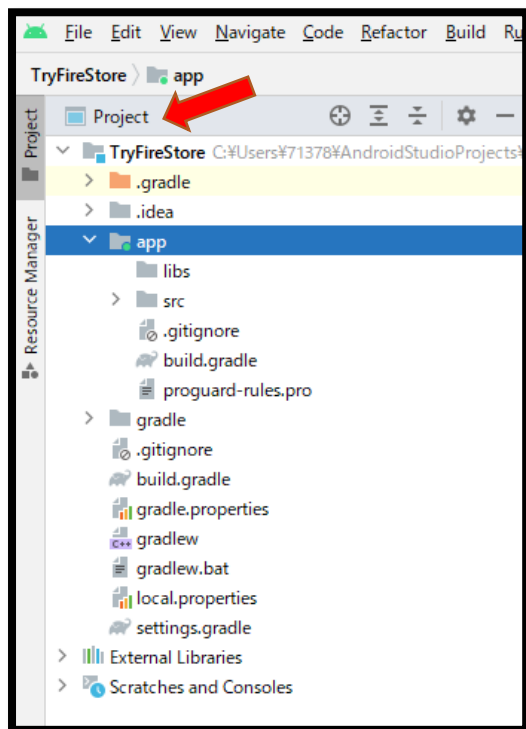
Dependencies

次へ

⑧ダウンロードした json ファイルを Anroid studio のプロジェクト内の app フォルダ内に配置

表示形式を Project に変更

app フォルダ内に先程の json ファイルを配置



次へ



※特にややこしいです

⑨(<project>/build.gradle)ファイルに SDK の記述を追記

1. google-services.json 構成ファイルの値に Firebase SDK からアクセスできるようにするには、Google サービスの Gradle プラグインが必要です。

☒ Kotlin DSL (build.gradle.kts) ☐ Groovy (build.gradle)

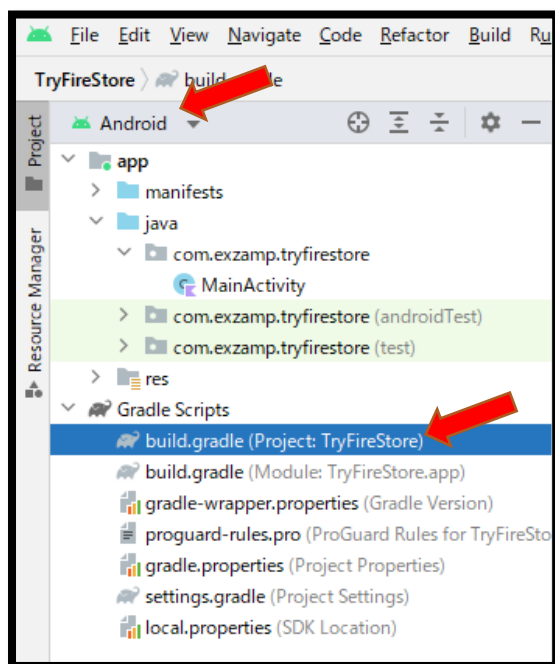
このプラグインを、プロジェクトレベルの build.gradle.kts ファイルに依存関係として追加します。

ルートレベル (プロジェクトレベル) の Gradle ファイル (<project>/build.gradle.kts) :

```
plugins {
    // ...

    // Add the dependency for the Google services Gradle plugin
    id("com.google.gms.google-services") version "4.4.0" apply false
}
```

<project>/build.gradle を開き

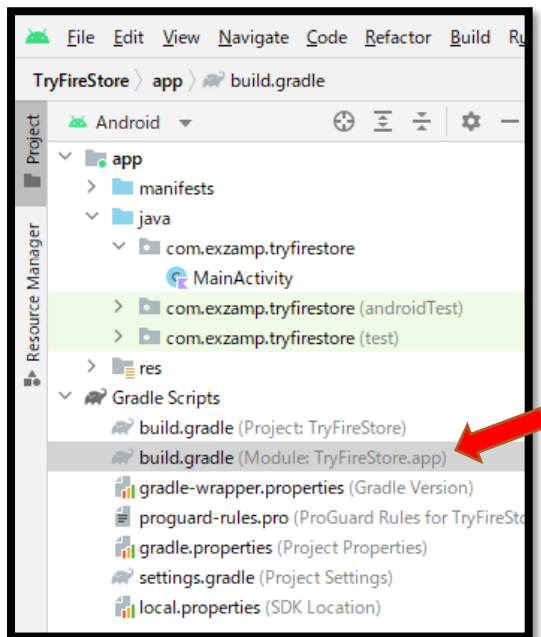


指定されたコードをコピペします(Sync now で反映を忘れずに)

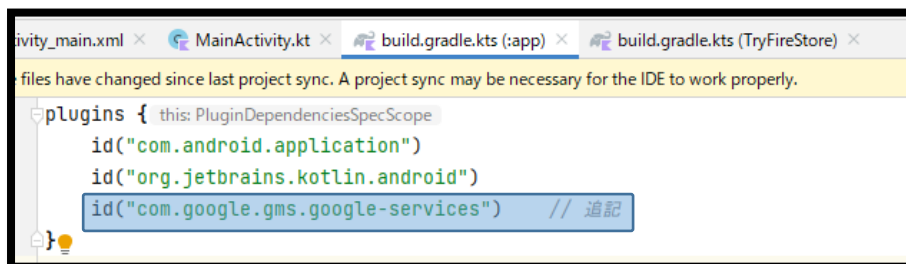
You can use the Project Structure dialog to view and edit your project configuration

```
1 // Top-level build file where you can add configuration options common to all sub-projects/modules.
2 plugins { this: PluginDependenciesSpecScope
3     id("com.android.application") version "8.1.0" apply false
4     id("org.jetbrains.kotlin.android") version "1.8.0" apply false
5     id("com.google.gms.google-services") version "4.4.0" apply false // 追記
6 }
```

⑩<project>/<app-module>/build.gradle) のファイルにもプラグインとライブラリを登録



指定されたコードをコピペします(Sync now で反映を忘れずに)



Sync Now で正常に同期が完了したら次へ(※エラーが出ているのに無視して先へ行かない事)

3. プラグインと必要な SDK を追加したら、Android プロジェクトと Gradle ファイルを同期します。

前へ

次へ

コンソールに進むをクリック

× Android アプリに Firebase を追加

- ✓ アプリの登録
Android パッケージ名: com.exzamp.tryfirestore
- ✎ 構成ファイルをダウンロードして追加する
- ✎ Firebase SDK の追加
- 4 次のステップ

これで設定は完了です。

[このドキュメント](#) をご覧になり、アプリで使用する各種の Firebase プロダクトの利用開始方法をご確認ください。

[サンプルの Firebase アプリ](#) もご覧いただけます。

または、コンソールに進んで Firebase をご確認ください。

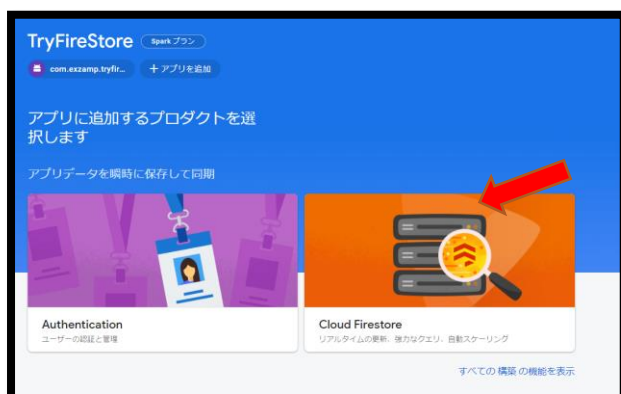
前へ

コンソールに進む

■ Cloud Firestore の設定

(公式ドキュメント：<https://firebase.google.com/docs/firestore/quickstart?authuser=0>)

「Cloud Firestore」コンテンツをクリック



データベース作成をクリック



ロケーションを Osaka に変更



テストモードを選択し「有効にする」を選択



これで firebase 側のデータベースが作成されました



続いて Android 側の設定

(公式ドキュメント：<https://firebase.google.com/docs/firestore/quickstart?authuser=0>)

Firestore のライブラリについて設定します。

必要なコードを(app/build.gradle)へコピーしましょう。

開発環境を設定する

必要な依存関係とクライアント ライブラリをアプリに追加します。

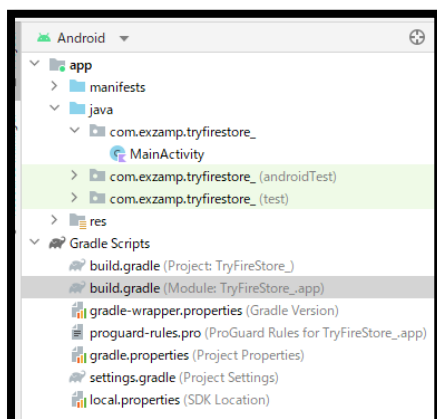
ウェブ向けの名前空間... ウェブ向けのモジュラー... iOS+ **Android** Dart Flutter Java もっと見る ▼

1. Android アプリに Firebase を追加するの手順に沿って操作します。
2. Firebase Android BoM (部品構成表) を使用して、**モジュール (アプリレベル) の Gradle ファイル** (通常は app/build.gradle.kts または app/build.gradle) で Android 向け Cloud Firestore ライブラリの依存関係を宣言します。

```
dependencies {
    // Import the BoM for the Firebase platform
    implementation(platform("com.google.firebase:firebase-bom:32.5.0"))

    // Declare the dependency for the Cloud Firestore library
    // When using the BoM, you don't specify versions in Firebase library dependencies
    implementation("com.google.firebase:firebase-firestore")
}
```

app/bulde.gradle を開き



公式ドキュメントからコピペ(**Sync Now** で反映を忘れずに)

```
dependencies { this: DependencyHandlerScope
    implementation("androidx.core:core-ktx:1.9.0")
    implementation("androidx.appcompat:appcompat:1.6.1")
    implementation("com.google.android.material:material:1.10.0")
    implementation("androidx.constraintlayout:constraintlayout:2.1.4")
    testImplementation("junit:junit:4.13.2")
    androidTestImplementation("androidx.test.ext:junit:1.1.5")
    androidTestImplementation("androidx.test.espresso:espresso-core:3.5.1")
    implementation(platform("com.google.firebase:firebase-bom:32.5.0")) // 追記
    implementation("com.google.firebase:firebase-firestore") // 追記
}
```

■データを保護する

アプリなどからのアクセス権限をテスト用に変更しておきます。

(公式ドキュメント：<https://firebase.google.com/docs/firestore/quickstart?authuser=0>)

データを保護する

Web、Android、または Apple プラットフォームの SDK を使用している場合は、[Firebase Authentication](#)と[Cloud Firestore セキュリティ ルール](#)を使用して Cloud Firestore のデータを保護します。

ここでは、開始するために使用できるいくつかの基本的なルール セットを示します。コンソールの[\[ルール\]](#) タブで、セキュリティ ルールを変更できます。

認証が必要です ロックモード **テストモード**

```
// Allow read/write access to all users under any conditions
// Warning: **NEVER** use this rule set in production; it allows
// anyone to overwrite your entire database.
service cloud.firestore {
  match /databases/{database}/documents {
    match /{document=**} {
      allow read, write: if true;
    }
  }
}
```

コピー

テストモードの設定をコピーし

FireBase 上の FireStore のルールページへ
移動しましょう

今回の TryFirestore プロジェクトを選択

続行するにはプロジェクトを選択してください

Firebase プロジェクト

+
プロジェクトを追加

TryFirestore
tryfirestore-cbdd4

ルールタブを選択し

先程の設定をペースト

最後に公開をクリック

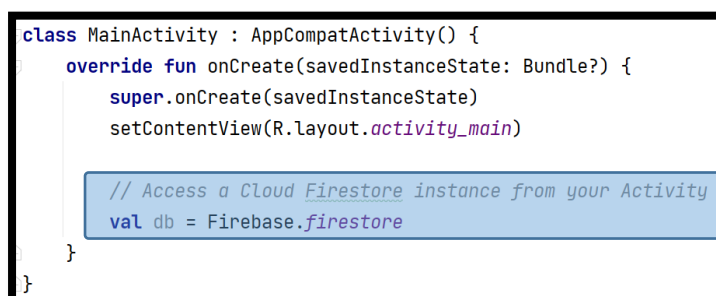


■Firestore への接続動作確認

公式ドキュメントを元に動作確認を行っていきましょう。

(公式ドキュメント：<https://firebase.google.com/docs/firestore/quickstart?authuser=0>)

◆Cloud Firestore の初期化



◆データを追加 : .add()

Firestore 上に追加するデータを記述します(onCreate メソッド内)

追加する場合は、add()メソッドを使用します。

データの追加

Cloud Firestore はデータをドキュメントに保存します。ドキュメントはコレクションに保存されます。データを初めてドキュメントに追加すると、Cloud Firestore によってコレクションとドキュメントが暗黙的に作成されます。コレクションやドキュメントを明示的に作成する必要はありません。

次のサンプルコードを使用して、新しいコレクションとドキュメントを作成します。



```
// Create a new user with a first and last name
val user = hashMapOf(
    "first" to "Ada",
    "last" to "Lovelace",
    "born" to 1815,
)

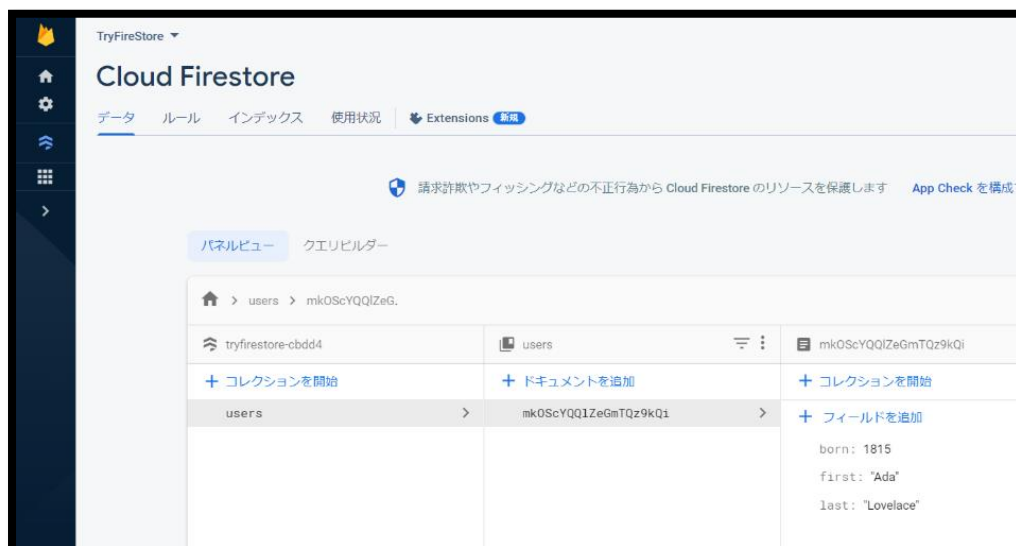
// Add a new document with a generated ID
db.collection("users")
    .add(user)
    .addOnSuccessListener { documentReference ->
        Log.d(TAG, "DocumentSnapshot added with ID: ${documentReference.id}")
    }
    .addOnFailureListener { e ->
        Log.w(TAG, "Error adding document", e)
    }
}
```

DocSnippets.kt

TAG の定数が無いので適当に作成しておきます。

```
class MainActivity : AppCompatActivity() {
    val TAG = "TryFirestore" // デバッグ用のタグ
}
```

実行し、データが増えているか確認してみましょう



Cloud Firestore

データ ルール インデックス 使用状況 Extensions 拡張

パネルビュー クエリビルダー

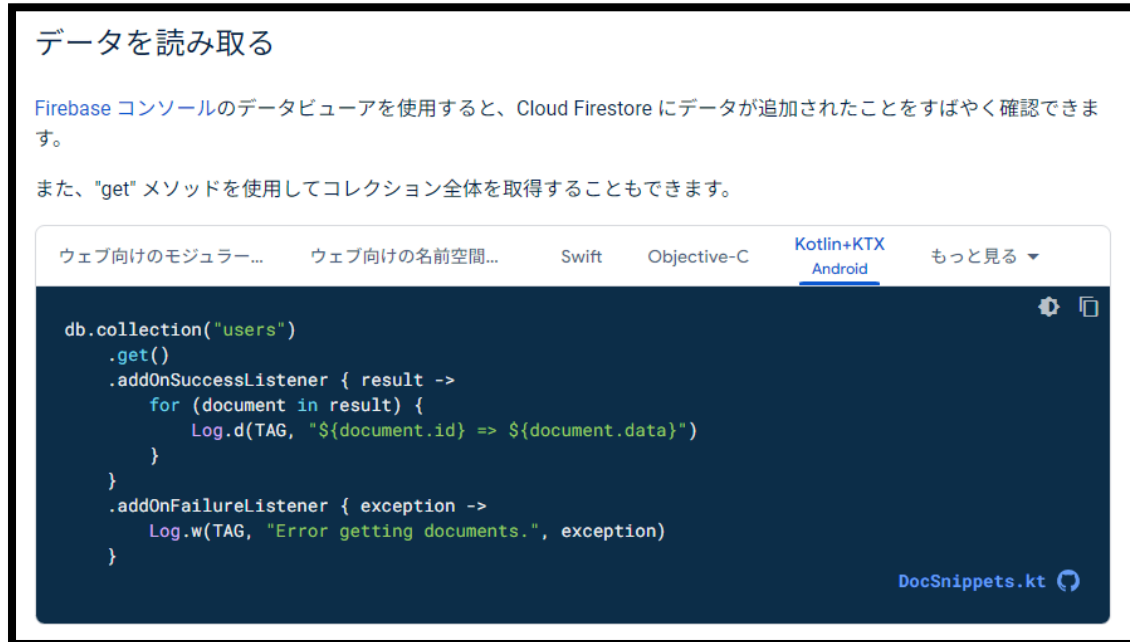
users > mk0ScYQQIZeG.

tryfirestore-cbddd4	users	mk0ScYQQIZeGmTQz9kQi
+ コレクションを開始	+ ドキュメントを追加	+ コレクションを開始
users >	mk0ScYQQIZeGmTQz9kQ1 >	+ フィールドを追加
		born: 1815
		first: "Ada"
		last: "Lovelace"

◆データの読み取り : .get()

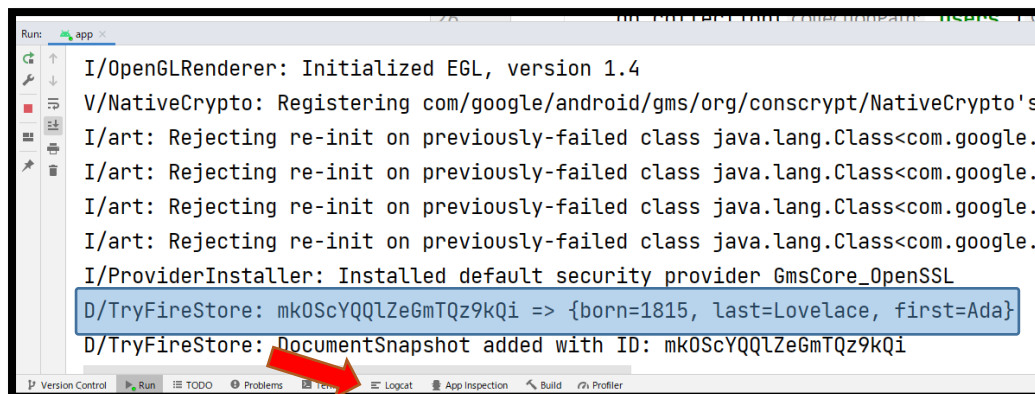
Firebase のデータを読み取る処理を記述します(onCreate メソッド内)

呼び出す場合は get() メソッドを使用します。



この状態でアプリを実行し、ログからデータを読み取れているか確認してみましょう。

Anroid Studio のログを Logcat タブを選択して確認してみましょう



取得が出来ていることを確認出来ます。