Firebase

Firebase とは

Google が提供しているモバイルおよびウェブアプリ開発プラットフォームです。
mBaaS(mobile Backend as a Service)の一つであり、アプリケーション開発におけるバックエンド環境を提供するサービスです。Firebase のサービスは主に「構築」、「リリースとモニタリング」、「エンゲージメント」にカテゴリが分かれます。

カテゴリ	概要
構築(ビルド)	ユーザ認証やデータベースおよびストレージなどアプリ開発で必要なバックエンド環境を提供
リリースと モニタリング	アナリティクスによるアプリ管理やアプリのクラッシュログの取得などアプリの品質改善のツールを提供
エンゲージメント	クライアントアプリ通知や A/B テストの実施機能など ユーザ体験の向上支援ツールを提供

・Firebase のサービス（一部抜粋）

サービス	概要
Realtime Database	NoSQL データベースでデータは JSON で管理される データはリアルタイムで同期される
Cloud Firestore	ドキュメント指向型の NoSQL データベース。Realtime Database の上位互換
Cloud Storage	写真や動画などのユーザ作成コンテンツのストレージサービス
Crashlytics	軽量なリアルタイムのクラッシュレポートツール
App Distribution	テスターにアプリを配布するサービス Crashlytics と連携することが可能
Cloud Messaging	連絡やその他のデータがあることをクライアントアプリに 通知を行うサービス
Google AdMob	アプリの収益化に利用できるモバイル広告プラットフォーム

※その他サービスについては Firebase の公式サイトを参照
https://firebase.google.com/?hl=ja

・Firebase のメリット・デメリット

メリット	・開発期間の短縮 アプリ開発に必要とされている機能が備わっています。 Firebase の機能を利用することで、開発期間を短縮することが可能 ・サーバ管理や保守の手間が省ける バックエンド処理を Firebase で行うことで、サーバの管理、保守が不要となる。
デメリット	・整合性が必要なデータが不得意 Firebase は NoSQL を採用しています。そのため整合性を求められるデータ管理には向いていません。 ・完全無料ではない Firebase は、従量課金制を採用しています。少ないデータのやり取りは無料で出来ますが膨大なデータを取り扱う場合料金が発生します。

※Firebase の料金についてはこちらを参照

https://firebase.google.com/pricing?hl=ja

ハンズオン　プロジェクトに Firebase を導入する

---

1. ハンズオン用プロジェクト(HandsOnFirebase)を開いて、動作チェックを行う

2. Firebase のプロジェクトを作成する　※Google アカウントのログインが必要
   https://console.firebase.google.com/?hl=ja

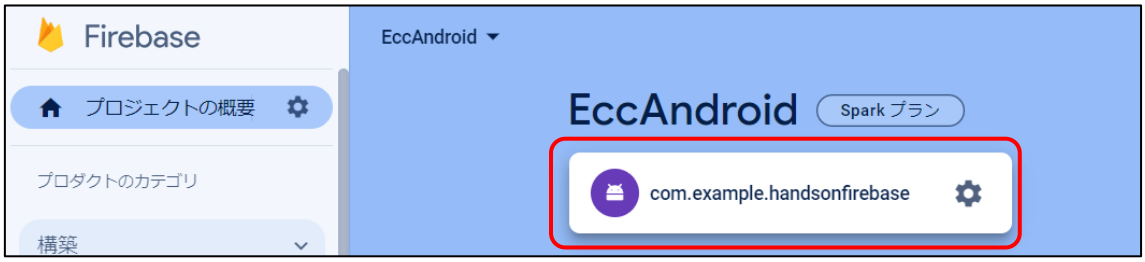
プロジェクト名	EccAndroid
Google アナリティクス(Firebase 向け)	有効にする
Google アナリティクスの構成	Default Account for Firebase

3. Firebase のプロジェクトに Android アプリを追加する

・画面の指示に従い、Android アプリに Firebase の設定を行う

Android パッケージ名	com.example.handsonfirebase
Google サービスの Gradle プラグイン	Kotlin DSL（build.gradle.kts）

※Android パッケージ名は、登録するアプリにより変わります



4. Android アプリを実行して、エラーが発生しないか確認する
   ※ここでエラーが発生する場合は、JSON ファイルや Gradle の設定を見直すこと

Cloud Firestore

Cloud Firestore とは Google Cloud インフラストラクチャ上に構築された、柔軟でスケーラブルな NoSQL クラウドデータベースです。主な特徴は以下の通りです。

特徴	説明
柔軟性 (NoSQL:Not Only SQL)	Firestore のデータモデルはドキュメント指向型です。データはドキュメントに格納され、コレクションにまとめられます。なお、ドキュメントにはサブコレクションを格納できるので複雑なネスト構造も管理することが出来ます。
高度な クエリ処理	データ(ドキュメント、コレクション)の取得でクエリが利用することが出来ます。またフィルタ処理と並び替え処理を組み合わせることも可能です。
リアルタイム アップデート	Realtime Database と同様に、データ同期を使用して、すべての接続端末のデータを更新します。また、シンプルな 1 回限りの取得クエリを効率的に実行するようにも設計されています。
オフラインサポート	アクティブに使用されるデータをキャッシュします。これによりアプリは、デバイスがオフラインになっている場合でもデータの書き込み、読み取り、聞き取り、クエリを実行できます。
スケーラビリティ (拡張性)	Google Cloud の強力なインフラストラクチャの優れた機能を提供します。そのため、最大規模のアプリからの最も過酷なワークロードに対応できるように設計されています。

NoSQL データベースとリレーショナルデータベースについて

データベース管理の主流はリレーショナルデータベースです。RDB はデータの整合性を維持すること(データに矛盾が発生しない仕組み)を最優先しているため、冗長性、データ構造の複雑化、パフォーマンスが犠牲になりやすいです。

それに対して NoSQL(Not Only SQL)は、RDBMS 以外のデータベース管理システムを表します。データモデルにはいくつかの種類が存在しており、データモデルによって実装に違いがあります。ただし、多くの NoSQL データベースでは JSON が使用されています。これは、データを名前と値のペアのコレクションとして表すオープンデータ交換形式です。

NoSQL データベースは、不確定なデータ、関連性のないデータ、または急速に変化するデータを処理するのに最適です。

	リレーショナルデータベース	NoSQL データベース
データモデル	データを行と列で構成されるテーブルに正規化。テーブル間の関係によってデータベースの参照整合性が維持される。	キーバリュー、ドキュメント、グラフ、列などのさまざまなデータモデルが存在している。
作業負荷	トランザクショナルで強固な一貫性を持つオンライントランザクション用に設計されています。	低遅延を含む多数のデータアクセスパターン用に設計されています。
ACID 特性	ACID 特性がすべて備わっている	ACID 特性の一部を緩和することで柔軟なデータモデルを実現しています
API	データの保存および取得のリクエストは、構造化クエリ言語（SQL）準拠のクエリを使用して伝えられる。	オブジェクトベースの API を使用して、データ構造の保存および取得を簡単に行うことができます。

・ACID 特性について

原子性：Atomicity	処理が中断しても中途半端にならない
一貫性：Consistency	データの内容が矛盾した状態にならない
分離性：Isolation	処理が同時実行されても副作用がない
永続性：Durability	記録した内容は消滅せずに保持され続ける

・リレーショナルデータベース

product 表

product_no	pname	category	price
0001	マルゲリータ	ピザ	1200

recipe 表

product_no	material_no	quantity
0001	00001	1
0001	00002	2
0001	00012	1
0001	00014	3
0001	00020	1

material 表

material_no	mname	origin	cost
00001	チーズ	北海道	50
00002	モッツァレラチーズ	北海道	700
00012	バジル	イタリア	30
00014	トマト	熊本	55
00020	トマトソース	NULL	30

・NoSQL（ドキュメント指向型）

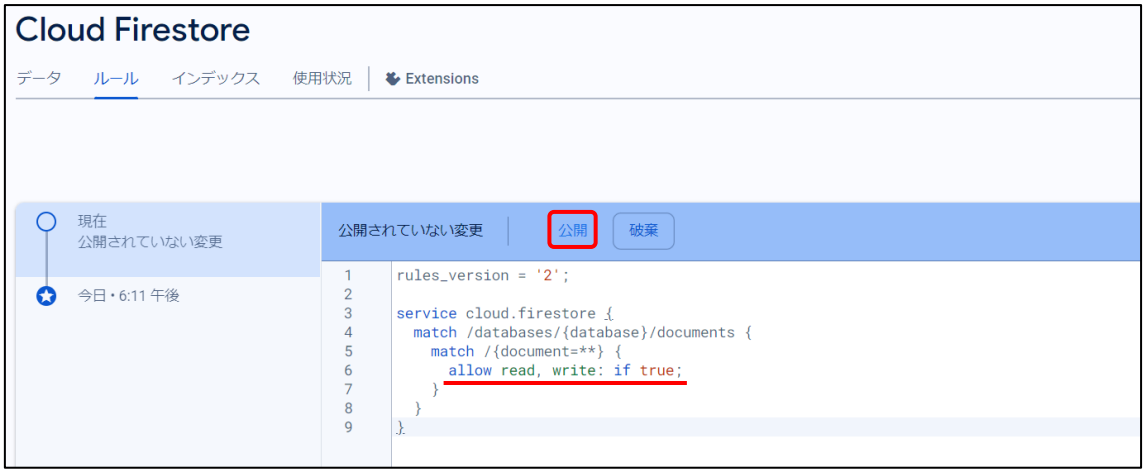
ハンズオン　Cloud Firestore のサービスを有効化する

1. Firebase のコンソール画面から「Firestore Database」を選択する。
   ※「Cloud Firestore」でも OK
2. 「データベースの作成」を選択

3. ダイアログに従い、データベースの新規作成を行う

ロケーション	asia-northeast2(Osaka)
データのセキュリティルール	本番環境モードで開始する

4. セキュリティルールをすべて許可に変更して、「公開」する



※本来は、アプリによってアクセスできる範囲などを決める必要があります！！

5. コンソール画面からテストデータを投入

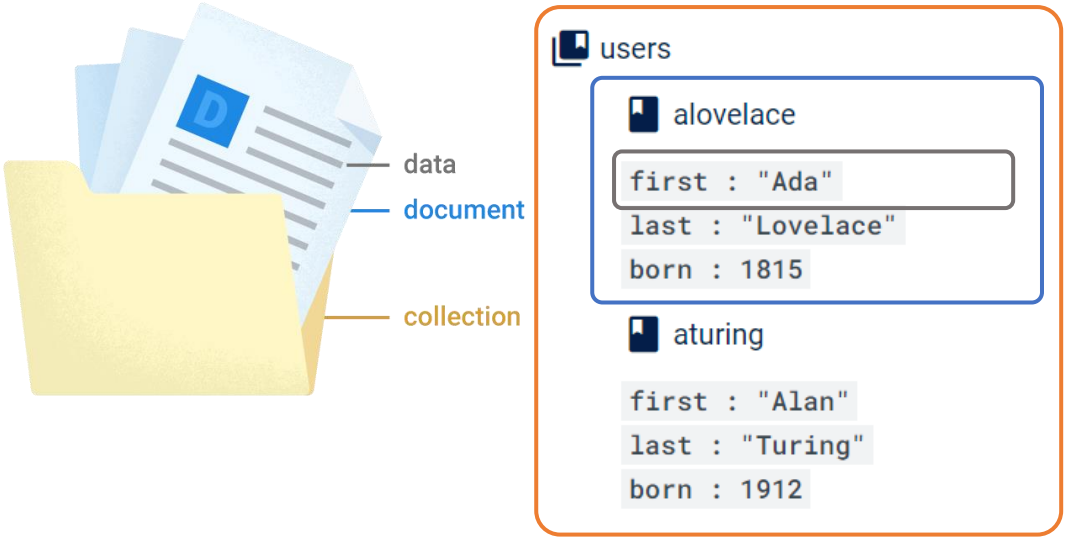
コレクション ID	todos		
└ドキュメント ID	自動 ID をクリック		
├フィールド	memo	string	はじめてのメモ
├フィールド	priority	number	2.5
└フィールド	createdBy	timestamp	カレンダーから現在日付を選択

Cloud Firestore データモデル

Cloud Firestore は NoSQL ドキュメント指向データベースです。SQL データベースとは違い、テーブルや行はありません。代わりに、データは「ドキュメント」に格納し、それが「コレクション」にまとめられます。

コレクション
コレクションは端的に言えばドキュメントのコンテナです。たとえば、users コレクションを作成して、さまざまなユーザを表すドキュメントを格納できます。
ドキュメント
ドキュメントは、値にマッピングされるフィールドを含む軽量のレコードです。各ドキュメントは名前で識別されます。いくつかの違いはありますが、一般的にドキュメントは軽量の JSON レコードとして扱うことができます。

・Cloud Firestore データモデルイメージ



Cloud Firestore の有効化および、データモデルの解説は以上です。次は、Android アプリケーションに Firestore の設定および、データ操作をハンズオンで進めていきます。

・Cloud Firestore でリアルタイム アップデートを入手する
https://firebase.google.com/docs/firestore/query-data/listen?hl=ja&authuser=0

ハンズオン　Firestore のデータを取得・同期する

1．アプリレベルの Gradle ファイルに Firestore のライブラリ依存関係を追記する

```
implementation("com.google.firebase:firebase-firestore")
```

2．MainActivity に Firestore のコレクション参照変数をクラス変数で宣言する

```
lateinit var todoRef: CollectionReference
```

3．onCreate メソッドで Firestore のコレクション参照をインスタンス化する

```
todoRef = Firebase.firestore.collection("todos")
```

4．リアルタイム同期に対応するため、変更検知されたら呼び出されるリスナーを実装する

```
todoRef.addSnapshotListener { value, error ->
    value?.let{
        val todos = it.toObjects(Todo::class.java)
        todoList.clear()
        todoList.addAll(todos)
        todoRecyclerView.adapter?.let{
            it.notifyDataSetChanged()
        }
    }
}
```

5．RecyclerView に設定している Adapter のリストデータをダミー用から Firestore 取得用リストに変更する

```
todoRecyclerView.adapter = TodoAdapter(
todoList,
```

コンソール上で、memo 項目を修正すると todos の変更を検知して、addSnapshotListener が呼ばれる。その内部で、todoList を修正しているので RecyclerView の内容が更新される。

・Cloud Firestore にデータを追加する
https://firebase.google.com/docs/firestore/manage-data/add-data?hl=ja&authuser=0

ハンズオン　データの新規登録を行う

1．ドキュメント ID を識別させるため、Todo データクラスにアノテーションを追加する

```
@DocumentId
val documentId: String? = null,
```

2．createFab の onClickListener ロジックを参照して、ダイアログ呼び出しから onDialogSubmitClick が呼び出される流れを把握する

3．onDialogSubmitClick で、Todo ドキュメントに新規データの追加処理を行う

```
// DocumentID が存在しない場合は、新規登録
todoRef.add(todo)
```