BaaSを活用した健康促進アプリ ActFitの開発

Development of Health Improvemet Application “ActFit” using BaaS

宮　下　　洋

要　約　Webサービスやモバイルアプリケーションにおいて，市場競争の激化，顧客ニーズの変化，アジャイル開発の浸透により，開発スピードの高速化が求められている．この要求に応えるため，BaaS（Backend as a Service）やmBaaS（mobile Backend as a Service）といったクラウドサービスが注目されている．これらのサービスが提供するバックエンドサービスを適切に活用することで，高速なアプリケーション開発が実現できる．BaaSを選定する際の評価点には提供機能，拡張性，スケーラビリティなどがある．それを基に選定したGoogle社のFirebaseを用いて，スマートフォンアプリ“ActFit”を開発した．アプリケーション開発におけるBaaSの有効活用には，ユーザー登録，データベースセキュリティとクエリ，バッチ処理などでBaaSの特性を考慮した設計と利用が決め手となる．

Abstract　In the development of web services and mobile applications, there is a demand for increased development speed due to intensified market competition, evolving customer needs, and the proliferation of agile development. To meet this demand, cloud services such as BaaS (Backend as a Service) and mBaaS (mobile Backend as a Service) have attracted attention. Leveraging the backend services provided by these services enables rapid application development. Evaluation points when selecting BaaS include provided functions, extensibility, and scalability. We developed the smartphone application “ActFit” using Google’s Firebase, which was selected based on these criteria. The key to effective use of BaaS in application development is to design and use BaaS in consideration of its characteristics, when developing such functions as user registration, database security and queries, and batch processing.

1. はじめに

情報技術は急速に進化しており，新しい技術をベースとしたビジネスアイディアやビジネスモデルが瞬時に市場に出現する．そうした中でリリースされるサービスは，競争力の維持・向上が求められる．それを満たすためには，新機能の追加，UI/UXの改善，バグ修正等，ビジネスの変化に合わせて素早くアプリケーションをリリース[1]できるアジャイル開発が適している．アジャイル開発では，ユーザーからのフィードバックを受けつつ，「分析」「設計」「実装」「テスト」「リリース」というイテレーションを高速に繰り返し，段階的に開発を行う[2]ため，イテレーションの各工程には短縮化が望まれる．

このような背景から，BaaS（Backend as a Service）やmBaaS（mobile Backend as a Service）を利用した開発が注目されている．BaaSは文字通りアプリケーションにバックエンドサービスを提供している．BaaSが提供するバックエンドサービスを利用することにより，アプリケーションのバックエンド側の開発を最小限に抑えられ，それは開発の高速化に寄与する．一方，提供されるバックエンドサービスによっては制約や制限が存在するため，それらを考慮したアプリケーションの開発が求められる．

本稿では，2 章にてユニアデックス株式会社（以降，ユニアデックス）が開発したスマートフォン向けアプリケーションである ActFit について説明する．3 章では，BaaS について他のクラウドサービスとの比較を通して述べ，BaaS が提供する機能について Google 社が提供する Firebase を中心に説明する．最後に 4 章で，BaaS の選定におけるポイントと BaaS を用いた開発における考慮点について，Firebase を例に取って述べる．

2. 健康促進アプリケーション ActFit の概要

本章では，ユニアデックスが新規ビジネス創出活動の一環として開発したスマートフォン向けアプリケーションである ActFit について説明する．ActFit は，SDGs の開発目標の一つである「すべての人に健康と福祉を」[*1] を実現するために企画された健康促進サービスである．「誰もが自身の健康状態を適切に管理し，病気を予防して健康でいきいきと生活できることが当たり前の世界に」をビジョンとしてかかげ，未病を改善し生活習慣病を防ぐことを目的としている．アプリケーションの流れと画面を図 1 に示す．

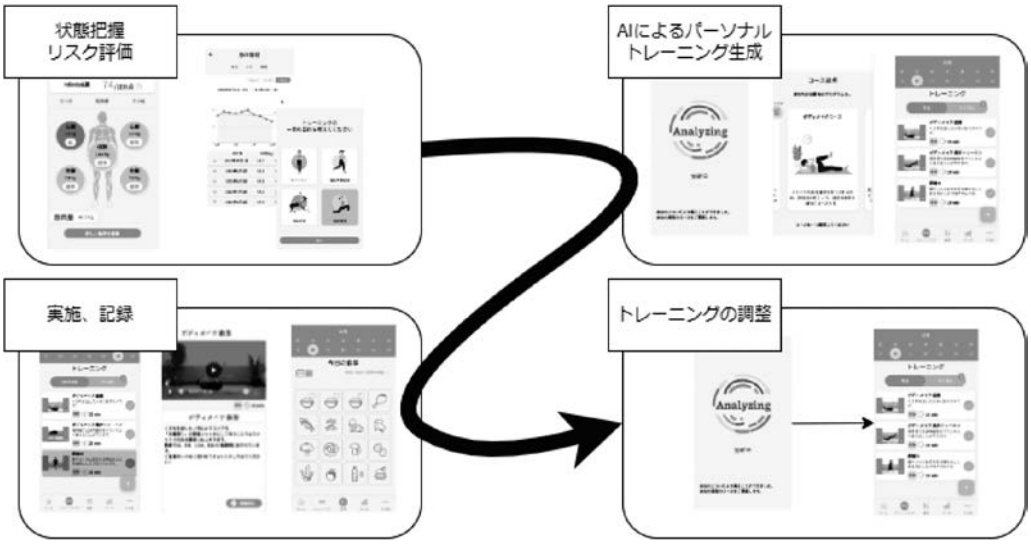

図 1　アプリケーション利用の流れ

図 1 の各ステップの詳細は以下の通りである．

1）状態把握 / リスク評価

アプリケーションから，健康診断情報，InBody[*2] で測定した体組成情報等のバイタル情報を取得し，これらの情報を基に体の状態を知り，隠れたリスクを把握する．それに加え，カウンセリングにより，ユーザーのモチベーションを評価する．

2）AI によるパーソナルトレーニング生成

体の状態，リスク，モチベーションを総合的に評価し，その人に合ったトレーニングメニューを AI が提案する．

3）実施，記録

トレーニング用動画の配信，トレーニングの実施と内容の記録を行う．

4）トレーニングの調整

実施状況に応じたトレーニングメニューの調整と各種フィードバックを行う．

また，本アプリケーションは企画の初期段階におけるプロトタイプという位置づけであり，「サービスとして価値確認（PoV：Proof of Value)」「ユーザーフィードバック収集（改善案策定)」「サービスの効果測定（健康促進に寄与するか)」を目的として作成されたため，「短期間」「少人数」「迅速な仕様変更への対応」でのアプリケーション開発が求められた．このような背景から，開発プロセスとしてアジャイル開発を採用し，アプリケーションのバックエンドとして BaaS を用いて開発することとなった．また，開発フレームワークには学習コスト，生産性，マルチプラットフォームへの対応の容易さを考慮し，Flutter[*3] を採用した．

3. BaaS の位置づけと主な機能

本章では，まず BaaS の位置づけを他のクラウドサービスとの比較を通して説明し，次に ActFit のバックエンドサービスとして採用した Google 社が提供する BaaS である Firebase を例に取り，BaaS が提供する主な機能について解説する．

3. 1　BaaS の位置づけ

クラウドサービスの主要な分類として IaaS（Infrastructure as a Service），PaaS（Platform as a Service），BaaS（Backend as a Service），SaaS（Software as a Service）がある．それぞれのクラウドサービスの特徴，適用対象となる主なアプリケーションの範囲，運用・保守コストを以下の 1）～ 4）に挙げる．また，アプリケーション開発の視点に立った場合のクラウドサービスの比較を表 1 に示す．

1）IaaS

インフラストラクチャを仮想化して提供するサービスであり，主に仮想マシン，ネットワーク，ストレージを提供する．

仮想マシン上で動作する OS 層以上についてアプリケーションに合わせて細かいチューニングができるため，特殊な OS やミドルウエアを前提とするアプリケーションや，オンプレミスアプリケーションのマイグレーションとして用いられることが多い．一方，OS 層以上について全て自前での運用となるため，運用・保守コストの負荷が高くなる傾向にある．

2）PaaS

実行基盤，ミドルウエア，開発におけるテスト，デプロイ，運用のための各種サービスを提供する．

ミドルウエア層以下がサービスとして提供されるため，開発者は開発のみに集中でき，高い生産性が期待される．ミドルウエア以下の層で特殊なカスタマイズが不要なアプリケーション全般が適用範囲となる．自動スケーリングに対応しているサービスもあり，自前の運用・保守コストの負荷が IaaS と比べ低くなる傾向にある．

3）BaaS

一般的なアプリケーション開発に用いる認証，データベース，ストレージ，ログ管理，分析，通知，API エンドポイント等のバックエンドのサービスを API として提供する．

バックエンドサービスが提供されるため，開発者はフロントエンド開発に集中でき，高い生産性が期待される．一方，BaaS で提供されるバックエンドサービスだけでは実装が難しい要件については，その要件に対応するために独自バックエンドサービスを開発することになる．そのため，BaaS が提供するバックエンドで対応できない要件が多く存在するケースでは BaaS のメリットを活かせない場合もある．また，BaaS が提供しているバックエンドの中には自動スケーリングに対応しているサービスもあり，運用・保守コストの負荷は低くなる傾向にある．

4）SaaS

ソフトウエアをサービスとして提供するもので，ユーザーはインターネット経由でアプリケーションにアクセスし，利用する．アプリケーションの適用範囲は，SaaS 上のアプリケーションが提供するカスタマイズ機能や API に依存するが，一般的には簡易的なアプリケーションに限られる．

表 1　アプリケーション開発から見たクラウドサービスの比較

サービス	主な開発範囲	自前の運用コスト	対象範囲
IaaS	バックエンド フロントエンド	高い	アプリケーション全般
PaaS	バックエンド フロントエンド	低い	特殊なミドルウエアや OS を前提としないアプリケーション全般
BaaS	フロントエンド	低い	一般的な Web，モバイルアプリケーション
SaaS	フロントエンド	低い	簡易的なアプリケーション

3. 2　BaaS が提供する主な機能

本節の各項では，BaaS が提供する主なバックエンドサービスについて，Google 社の BaaS である Firebase を例に解説する[*4]．Firebase の各種バックエンドサービスは，Google Cloud Project 上にデプロイされる形で提供される．つまり，クラウド上に Firebase Project を作成すると，内部的に Google Cloud Project が作成され，Firebase がその Google Cloud Project にデプロイされる[3]．このような構成となっているため，Google Cloud Project に任意のクラウドサービスを追加することにより，バックエンドサービスを容易に実装することができる．

3. 2. 1　Firebase Authentication

ユーザー認証機能を提供する．メールアドレスとパスワードを用いてユーザー認証を行うネイティブプロバイダに加え，フェデレーション連携[*5]にも対応している．また，Firebase Authentication を Identity Platform へアップグレードすることにより，ユーザー認証機能を Google Cloud が提供する Identity Platform に切り替えることができる．これにより，OIDC や SAML[*6] によるログイン，多要素認証，ブロッキング関数の設定等の機能が有効化されるため，認証に関するより多くの要件に対応できるようになる[*7]．

3. 2. 2　Cloud Firestore

NoSQL データベースサービスを提供する．データ形式はドキュメント型であり，ドキュメントの格納先はパスとして指定する[*8]．リアルタイムリスナー機能により，クラウド上のドキュメントに変化があった場合，その変更内容がドキュメントを監視しているアプリケーションに即座に通知される．また，Firebase SDK と連携したオフラインサポートにより，オフライン時のアプリケーションの変更がオンライン状態に遷移したときに自動的にクラウド側のドキュメントに同期される[5]．

データ操作のセキュリティについては，パスベースの独自構文での記述となる．ドキュメント単位だけではなく，コレクション，フィールド単位での制御など，きめ細かいアクセスルールを作成することができる[*9]．

3. 2. 3　Cloud Storage for Firebase

写真や音声，動画等のコンテンツ保存用のオブジェクトストレージサービスを提供する．内部的には Firebase 用の Cloud Storage のバケットが作成され，そのバケットにデータが保存される[7]．

データセキュリティについては，独自構文での記述となる．オブジェクト単位だけではなく，パス単位でのアクセス制限ができる．また，ルールを記述する際に firestore.get()，firestore.exists() 関数を使用することにより，Firestore 上に格納されたドキュメントを参照したアクセスルールの記述もでき[8]，きめ細かいセキュリティルールの作成ができる[*10]．

3. 2. 4　Cloud Functions for Firebase

独自の Web API など，アプリケーション独自のバックエンドを実装するための機能を提供する．Firebase がネイティブで提供するサービスだけでは実現が難しい要件については Cloud Functions for Firebase を用いて実装することとなる．関数のデプロイを行うと，内部的に Cloud Functions，Cloud Run，Cloud Pub/Sub，Cloud Scheduler，Secret Manager 等の関数実行に用いる Google Cloud のリソースが作成される[*11]．

Cloud Functions for Firebase は，ユーザーが https 経由で呼び出す Web API だけではなく，スケジュールで動作するバッチ処理プログラム，イベント等をトリガーとして実行するプログラムも作成できる[9]．また，自動スケーリングにより，一時的かつ急激な処理の増加に対してメンテナンスなしで対応できる．

3. 2. 5　Firebase Hosting

Web サーバのホスティング機能を提供する．アップロードしたファイルは，世界中に配置されている Google の CDN エッジの SSD キャッシュに保存され，コンテンツに最適な圧縮方法が自動的に選択されて配信される[10]．これによりユーザーがどこにいても，コンテンツを高速に配信することができる．また，カスタムドメインを用いたホスティングにも対応しているため，ブランドを象徴する独自のドメイン名を使用できる．一つのホスティング環境に対して複数のサイトを作成することができ，テストユーザーでデプロイ候補をテストするなどの用途にも使用できる．

3. 2. 6　Firebase Cloud Messaging

Android，iOS，Web に対してプッシュ通知を行う機能を提供する．この機能により，アプリケーションを開いていない利用者に対してお知らせなどの通知ができるようになる[11]．エンゲージメントが低下しているユーザーに対して通知を行うことで，エンゲージメントを高めていく効果が期待できる．

3. 2. 7　Google Analytics for Firebase

アプリケーションの使用状況，クラッシュ情報，ユーザーエンゲージメントについて分析するための機能及びダッシュボードを提供する．独自の分析イベントが定義でき，その独自イベントを利用してアプリケーション独自の分析もできる[12]．また，発行された分析イベントは Firebase Functions のトリガー条件としても設定できる．

3. 2. 8　Firebase Machine Learning

モバイルアプリで一般的に用いられるテキスト認識，翻訳，画像のラベル付け，ランドマーク認識などの AI 機能を API として提供する[13]．また，独自の機械学習モデルもサポートしており，作成した機械学習モデルを Firebase にデプロイすることにより，自動的にアプリケーション側にモデルを配信することができる．

3. 2. 9　Firebase A/B Testing

アプリケーションの A/B テストを支援する機能を提供する．A/B テストを使用することにより，UI の変更やプッシュ通知に対するユーザーのリアクションの分析が容易になる[14]．

4.　Firebase を用いたアプリケーション開発

本章では，まず ActFit のシステム構成について Firebase が提供しているバックエンドサービスの役割を中心に説明し，その後，BaaS 選定のポイントについて述べ，最後にアプリケーション開発を通して得られた Firebase を用いた開発における考慮事項について説明する．

4. 1　ActFit のシステム構成

本節では，ActFit のシステム構成について説明する．ActFit における Firebase サービス及び Google Cloud サービスの構成を図 2 に示し，役割を以下の 1）～ 8）で説明する．

1）Firebase Authentication

ユーザー認証のために使用する．ActFit では，実装コストと開発期間の制約から，フェデレーション連携は行わず，Firebase Authentication のネイティブ認証機能であるメールアドレスとパスワードによる認証を採用した．

2）Cloud Firestore

ユーザーに紐づく各種データ，トレーニング一覧などのマスターデータの格納先として使用する．

3）Cloud Storage for Firebase

トレーニングに紐づく画像など，今後のトレーニングの拡充に伴い増加する可能性のあるメディアファイルのホスティングとして使用する．

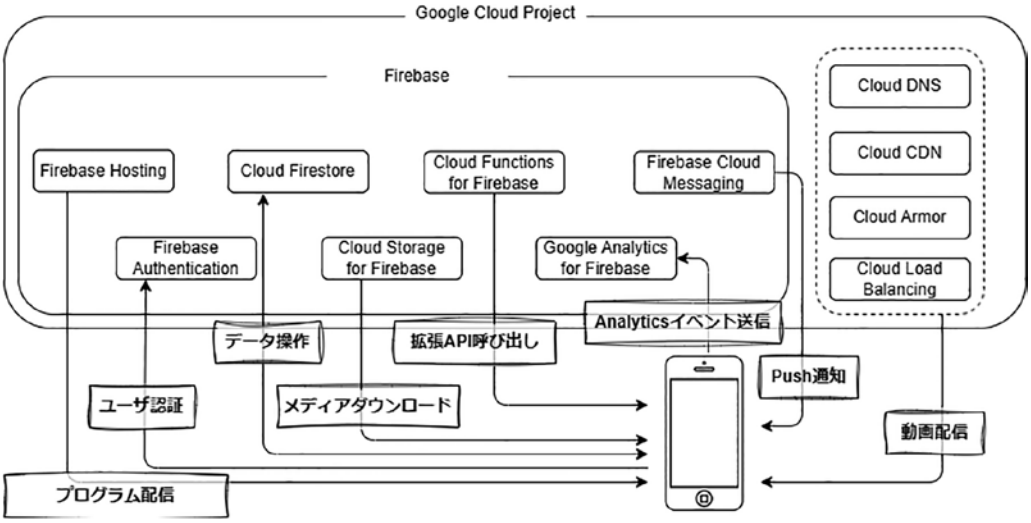

図 2　システム構成

4）Cloud Functions for Firebase

BaaS が提供する機能だけでは実現が難しい要件に対応するために使用する．ユーザーに提供する Web API 以外にも，以下 i）～vi）に挙げるバッチ処理，イベントドリブン処理を実装している．

i）　定期バックアップ処理
ii）　定期 email 未確認ユーザーの削除
iii）　定期プッシュ通知送信
iv）　ユーザー登録イベントを契機とした内部管理データのアップデート
v）　ユーザー削除イベントを契機としたデータ削除
vi）　ドキュメント追加削除イベントを契機とした集計データのアップデート

5）Firebase Hosting

本アプリケーションは，BIPROGY グループ社員が会社貸与のスマートフォンから利用することを前提としており，デプロイを簡略化するため，モバイル向け Web サイトをスマートフォン向けアプリのように使えるようにする PWA（Progressive Web Apps）として実装している．アプリケーションのホスティングとして Firebase Hosting を用いた．

6）Firebase Cloud Messaging

エンゲージメントを促すメッセージを通知するため，Firebase Cloud Messaging を利用する．但し，本アプリケーションのプラットフォームがブラウザであるため，Web Push に対応しており，ユーザーが通知を許可した端末のみが対象である．

7）Google Analytics for Firebase

アクティブユーザー数及びエンゲージメント時間の測定のため，Firebase Cloud Analytics を利用する．アクティブユーザー数やエンゲージメント時間の推移については Firebase のコンソール上のダッシュボードから確認・分析を行った．

8）Cloud CDN，Cloud Load Balancing，Cloud Armor，Cloud DNS

ログイン済みユーザーのみが動画を視聴できるようにするため，Cloud CDN の署名付き URL を用いたアクセス制御を有効化し，Cloud Functions for Firebase 上に署名付き URL を発行する Web API については Cloud Functions for Firebase にて実装した．

4. 2　BaaS を選択する際の評価ポイント

アプリケーションの開発では，開発のしやすさ，セキュリティ，運用費用を総合的に評価して開発・実行基盤サービスを選択する．本節では，BaaS として Firebase を選択するにあたり，特に注目したポイントを以下 1）～ 7）に挙げて説明する．

1）提供機能

作成しようとしているアプリケーションで使う機能の提供の有無や，バックエンドサービス間の連携の容易さ，統合管理コンソールの有無が評価ポイントに挙げられる．

2）拡張性

BaaS だけでは対応が難しい要件を実現するための拡張的な仕組みの有無や，他のクラウドサービスとの連携の有無，実装・連携の容易さが評価ポイントに挙げられる．

3）スケーラビリティ

Web サービスではアクセス数の一時的な増加への対応が求められる．そのため，BaaS においてもスケーラビリティを確保するための仕組みが提供されているかが評価ポイントに挙げられる．具体的には自動スケーリングの有無が考えられる．

4）セキュリティ

外部に向けたクラウドサービスでは，何らかの攻撃により悪用され，社会的信頼の損失，大量アクセスによる想定を超える高額な使用料の発生等のリスクが存在する．このようなリスクに対応するためのセキュリティ機能，リソース制限，ログ管理，アラート，アラートに基づく自動対処の仕組みの有無が評価ポイントに挙げられる．また，単純に機能の有無だけではなく，リスクに対応するための設定，アラートに基づく自動化のプログラムの作成の容易さについても考慮する．

5）価格体系

クラウドサービスを利用したアプリケーションの運用費用は，サービスの使用料，転送データ量に大きく依存する．そのため，アプリケーションを動作させる上でかかる各サービスの使用料，データ転送量の見積もりを行い，クラウド使用料が問題のない範囲に収まるかを評価する．

6）SDK 対応

多くの場合 BaaS を利用するための SDK が各クラウドベンダーから提供されているが，SDK が対応している言語についてはベンダーごとに異なる．そのため，開発で使用する言語及びフレームワークでの SDK の提供の有無が評価ポイントに挙げられる．

7）ドキュメント/知見者

開発を進めるにあたり，仕様等の確認/調査を行う．そのため，BaaS サービス/SDK に関するドキュメントの充実度と，周りに知見者がいるかが評価ポイントに挙げられる．

これらの評価ポイントを基に ActFit の開発において Firebase を評価した結果を表 2 に示す．

表 2　Firebase の評価結果

評価ポイント	Firebase の評価
提供機能	ActFit が提供する多くの機能を Firebase が提供するバックエンドを用いて実装でき，独自開発するバックエンドは少ない Firebase Console から一元的に管理できる
拡張性	Cloud Functions for Firebase を利用して拡張できる Firebase がデプロイされている Google Cloud Project にサービスを追加することで拡張できる
スケーラビリティ	多くのサービスで自動スケーリングに対応している
セキュリティ	API 及び予算単位でのクォータ，アラート，アラートをトリガーとした処理の設定ができる これらの機能により DoS 攻撃に対応できる
価格体系	標準的な使用を前提として，バックエンド費用を見積もった結果，予算内に納まった
SDK 対応	開発フレームワークである Flutter に対応した SDK が提供されている
ドキュメント/知見者	Firebase/SDK ともに日本語ドキュメントが提供されている 周りに Firebase を利用したアプリケーション開発の経験者がいた

4. 3　Firebase を用いた開発における考慮点

本節では，ActFit のアプリケーション開発を通して得られた，Firebase を開発のバックエンドとして用いる際の考慮事項について，実例を示しながら説明する．

4. 3. 1　ユーザー登録

ユーザー登録に関して，Firebase Authentication が提供する機能だけでは実現が難しい要件として，「ユーザーはサブスクリプションに紐づけられる形で登録されること」，「サブスクリプション毎にユーザー ID として許可するメールアドレスのドメインが制限できること」，「ユーザー ID として登録したメールアドレスが本人のものであることが確認できない場合はアプリケーションへのアクセスを拒否すること」があった．これらの要件を満たすため，以下 1）～ 4）に示す方法でユーザー登録機能を実装した．

1）Firebase Authentication の Identity Platform へのアップグレード

Firebase Authentication ではセルフサービスでのユーザー登録機能が無効化できないため，Firebase Authentication を Identity Platform にアップグレードし，セルフサービスによるユーザー登録を無効化した．

2）ユーザー登録用 Web API の作成

クラウドサービスが提供するセルフサービスでのユーザー登録を無効化したため，その代替として，ユーザー登録のための独自 Web API を，本項で挙げた要件を満たす形で Cloud Functions for Firebase 上に実装した．

3）メールアドレス未確認ユーザーの自動削除

他人のメールアドレスを用いたユーザー登録を回避するため，ユーザー登録後一定の期間内に本人のメールアドレスであることが確認できなかった場合，自動的にユーザーアカウントを削除する処理を Cloud Functions for Firebase 上の定期実行処理として実装した．

4）セキュリティルール

メールアドレス未確認ユーザーからの不正な処理の実行を防ぐため，各種 BaaS 及び Cloud Functions for Firebase 上に作成した Web API において，ユーザーのメールアドレスの確認状態を参照しアクセス/処理の実行の可否を制御するよう実装した．

4. 3. 2　データベースセキュリティ

Cloud Firestore では，ドキュメント及びドキュメントのコレクションはパスとして表現され，コレクションとドキュメントがパス上交互に出現する．また，セキュリティルールを記述する際の対象となるコレクションやドキュメントの指定はパスが基本となる．そのため，パスによってセキュリティ設定が分離できるよう，特定のユーザーに紐づけられるデータについては，データの種別ごとに「users/｛ユーザー ID｝/｛ドキュメント種別｝/」コレクション配下に格納し，ユーザー間で共有する読み取り専用のデータについては「commons/」コレクション配下に格納するように設計した．

セキュリティルールについては，そうしたパスの設計を前提として以下 1）～ 3）に示す方針で作成した．

1）メールアドレス未確認ユーザーに対するセキュリティルール

メールアドレス未確認ユーザーについては，ユーザーが本人であることを確認できていない状態であるためデータへのアクセスを拒否する．

2）ユーザーに紐づくデータのセキュリティルール

アクセスユーザーのユーザー ID とアクセス対象のドキュメント及びコレクションパス上のユーザー ID が一致する場合のみアクセスを許可する．

ドキュメント種別毎に，ドキュメント取得クエリに関する limit, offset, orderBy 等の制限や，ドキュメントに対する get, create, update, delete 操作の制限を設定する．

3）共有データ

ログインしているユーザーのみに共有データを格納する「commons/」コレクション配下のドキュメントに対する読み取り権限を設定する．

4. 3. 3　集計データ

基本的な集約クエリについては Cloud Firestore においてもサポートされているが，サポートしている集約クエリは基本的なものに限られている．サポートされていない集約を行う場合，集約する全てのデータに対する読み取りアクセスが前提となる．Cloud Firestore の料金体系はドキュメントの読み取り数に依存することから，集約の内容や集約するドキュメント数，集約の実行頻度によっては想定した以上のクラウド費用が発生する可能性がある．そのため，サポートされていない集約については，集約するドキュメントの更新をトリガーとして動作する集約結果更新処理を Cloud Functions for Firebase 上に作成することで対応した．

本アプリケーションにおける集約は固定的なものであったため，この対処で奏功したが，分析など動的な集約を行う場合は，集約用の DB サービスを用意して，Firestore へのデータ操作をトリガーとして同期をとり，実際の集約には集約用 DB を用いる，等の仕組みを要する．

4. 3. 4　Cloud Functions for Firebase の世代選択

Cloud Functions for Firebase には 2023 年 11 月時点で第 1 世代と第 2 世代の 2 種類の世代が存在する．第 2 世代の Cloud Functions for Firebase はバックエンドとして Cloud Run を用いており，第 1 世代と比べ実行時間の制約の緩和や同時並行実行数の設定，インスタンスの設定，スペックの自由度の拡大等の柔軟性と効率性に関する設定ができるようになった[*12]．このような理由により，ActFit の開発には第 2 世代の Cloud Functions for Firebase を採用した．

4. 3. 5　独自 Web API セキュリティ

Cloud Functions for Firebase 上に作成した Web API はインターネット上に公開されるため，セキュリティを考慮し以下 1）～ 4）に示す設計方針を執った．

1）ログイン状態・メールアドレス確認状態の確認

ログインユーザーのみが実行する Web API については，Web API のコード内の先頭でログイン状態，メールアドレス確認状態を参照し，未ログインもしくはメールアドレス未確認の場合，すぐに終了し，エラーを返す．

2）App Check の利用

Firebase プロジェクト上で App Check を有効化したうえで，Web API の呼び出しに対して App Check によるアプリケーションのチェックをラインタイムオプションとして設定する[*13]．

3）フォーマットチェック

無効なリクエストによるセキュリティリスクを回避するため，Web API のコード内の先頭で API 呼び出し時にクライアントから送信されたデータのフォーマットをチェックし，仕様に沿わないフォーマットの場合，すぐに終了し，エラーを返すように実装する．

4）エラーメッセージ

Web API で何らかの理由でエラーが発生した場合に返すエラーメッセージについては，それによって情報の流出や，攻撃の手掛かりを与える可能性のある情報を含めないよう実装する．例えば，ユーザー登録用 Web API において，エラーメッセージ内に既に登録済みであることを示すメッセージが含まれると，ユーザーの登録状況が流失し，攻撃の手掛かりを与えてしまうため，そのような情報をエラーメッセージ内に含めないよう実装する．

4. 3. 6　Cloud Functions for Firebase のランタイムオプション

Cloud Functions for Firebase において実行するプログラムのランタイムオプションの設定に関して考慮した点について，以下 1）～ 4）で説明する．

1）リージョン設定

Firestore，Cloud Functions for Firebase ともにリージョンリソースとしてデプロイされるため，異なるリージョンにデプロイした場合，ネットワーク上離れた場所にサービスが存在することとなる．その状態で，Cloud Functions for Firebase 上で実装した処理の中に Firestore 上のデータへのアクセスが含まれるとデータ転送のオーバーヘッドが発生し，処理終了までに非常に長い時間がかかる場合がある．そのため，データアクセスを伴って Cloud Functions で行う処理については，アクセスするデータを格納しているリージョンと同一のリージョンとなるようデプロイした．

2）タイムアウト

テスト実行を通して処理にかかる時間を計測し，その結果を基に処理内容に応じたタイムアウト時間を設定した．それにより，処理が終了する前にタイムアウトしてしまうことを防ぎつつ，何らかの問題により時間がかかりすぎている処理の停止ができる．

3）スペック

スペックについては机上での見積もりが難しいため，テストで1インスタンスでの同時実行数の上限まで実行し，その際のリソース使用量を基にメモリ容量とCPU数を設定した．

4）インスタンス数，同時実行数の設定

第2世代のCloud Functions for Firebaseでは最大インスタンスを設定しない場合，デフォルト値である100が使用される[17]．そのままの状態で外部から攻撃されると，クラウド使用料が想定以上に高額となる可能性がある．このようなことから，ユーザー数，呼び出し頻度，1件当たりの処理時間，安全マージンを考慮し，インスタンスの最大数及びインスタンスごとの同時実行数を設定した．また，最小インスタンス数を指定しない場合，デフォルト値の0が使用されるため，コールドスタートによるレイテンシーが発生する．これを回避するため，最小インスタンス数についても利用状況に合わせて設定した．但し，最小インスタンス数を1以上に設定すると処理を行っていない状態でもクラウド費用が発生するため，注意を要する[*14]．

4.3.7　バッチ処理

バックアップなど運用上の定期的な処理についてはCloud Functions for Firebaseの定期実行処理として実装した．また，Firestore上のデータにアクセスする処理についてはI/O待ちの時間が多くなるため，同時に実行できる処理を非同期処理関数として切り出して同時並行に実行させることにより，I/O待ちの時間を低減する実装とした．

今回実装したバッチ処理については，こうした実装上の工夫によりCloud Functionsの最大実行時間である60分以内で終了するものであったが，それでも最大実行時間内に終了しないバッチ処理がある場合は，処理を複数のCloud Functionsに分割するなどのさらなる工夫が求められる．

4.3.8　Google Cloudサービスの利用

動画の配信については，「ログインしているユーザーのみが視聴できること」，「キャッシュを用いて効率的に配信すること」というFirebaseが提供するバックエンドサービスだけでは実現が難しい要件に対し，Firebaseプロジェクトをデプロイしているの Google Cloud Project上にCloud CDNを中心とした配信に用いるクラウドサービスをデプロイし，それらサービスとFirebaseを組み合わせることで実装した．

具体的には，動画配信で使用するCloud CDN側で有効期限内の署名付きURLのみアクセスを許可する設定とし[19]，Cloud Functions for Firebase上に有効期限付きの署名付きURLを発行するWeb APIを作成することで実現した．

5.　お　わ　り　に

アプリケーション開発の高速化，運用負荷低減の要請から，BaaSを利用したアプリケーショ

ン開発が注目されている．BaaS を用いることにより，高速な開発とデリバリー，運用負荷の軽減が実現できる．一方，アプリケーション開発における BaaS の有効活用には BaaS の特性を考慮した利用・設計が決め手となる．

本稿では，BaaS の有効活用のため，BaaS の提供範囲となるアプリケーション，BaaS の選定ポイント，BaaS を用いたアプリケーション開発におけるポイントを筆者が開発に携わった ActFit を例に取り解説した．BaaS が提供するバックエンドサービスの拡充，SDK の拡充，運用・スケーラビリティの容易さから，今後 BaaS を用いたアプリケーション開発が広がっていくものと推察される．本稿が BaaS を用いたアプリケーション開発の一助になれば幸いである．

最後に本稿執筆にあたり，ActFit の企画の立ち上げにご支援・ご協力いただいた社内外の関係者，アプリケーションの設計・実装にご支援いただいた皆様，執筆についてご助言・ご指導いただいた方々に御礼申し上げる．

＊1　SDGs（Sustainable Development Goals）とは，「持続可能な開発のための 2030 アジェンダ」に記載された国際目標である．「地球上の誰一人として取り残さない」ことを理念とし，人類，地球及びそれらの繁栄のために設定された行動計画であり，17 のゴールと 169 のターゲットで構成されている．「すべての人に健康と福祉を」という目標は，その三番目の目標である．
https://sdgs.un.org/goals

＊2　InBody は株式会社インボディ・ジャパンが提供する体組成計であり，部位別の筋肉量，脂肪量，体水分量やミネラル量等を正確に測定できる．
https://www.inbody.co.jp/inbody-series/

＊3　Flutter は Google 社が開発したマルチプラットフォーム対応のオープンソースのアプリケーションフレームワークであり，アプリケーション開発には言語として Dart を用いる．

＊4　本稿の Firebase 及び Google Cloud が提供するサービスに関する説明は 2023 年 11 月時点で公開されている情報を基にしている．

＊5　フェデレーション連携とは，アカウント認証を連携させる仕組みであり，この機能を使用することでアプリケーションの認証を X（旧 Twitter）や Google 等の認証サーバで行うことができる．

＊6　OIDC（OpenID Connect）と SAML（Security Assertion Markup Language）は ID プロバイダ（IdP）がユーザー検証とアクセス制御を実装するための認証プロトコルである．

＊7　Firebase Authentication と Identity Platform との違いの詳細については，Identity Platform と Firebase Authentication の違い（参考文献[4]）を参照のこと．

＊8　SQL に比べてサポートされている集約関数が少ないこと，join などの操作ができないことから，複雑なクエリが要求される場合には Google Cloud が提供するその他のサービスとの組み合わせを検討する．

＊9　但し，ルールセットのサイズ，ルール内での関数の呼び出し回数等に制限があるので[6]，セキュリティルールをシンプルに記述するためのパスやドキュメント設計が求められる．

＊10　但し，セキュリティルールに関して＊9 の Firestore と同様の制限があるので，セキュリティルールをシンプルに記述するためのパス設計が求められる．

＊11　但し，Google Cloud Project 上のこれらのサービスについては Firebase Console からは確認できないため，設定や実行状況の詳細は Google Cloud コンソールから確認する．

＊12　世代間の機能比較の詳細については，Cloud Functions バージョンの比較（参考文献[15]）を参照のこと．

＊13　Cloud Functions for Firebase 上で作成する Web API で App Check を適用するには，Firebase プロジェクトで App Check を有効化した上で onCall() に引数で指定するランタイムオプションの enforceAppCheck を true に設定する．詳しくは，Cloud Functions の関数で App Check の適用を有効にする（参考文献[16]）を参照のこと．

＊14　アイドル時の費用の詳細については，Cloud Functions の料金（参考文献[18]）を参照のこと．

参考文献　［1］　平鍋健児，「変化を味方につけるアジャイル開発」，SEC journal Vol.11 No.4 Mar., pp.22-25，2016

［2］ 加藤智之，「開発型事業のアジャイル P2M ―アジャイル P2M の導入―」，Journal of International Association of P2M Vol.13 No.2，pp.46-59，2019
［3］ Firebase プロジェクトについて理解する，Firebase ドキュメント，Google，2023 年 10 月．
https://firebase.google.com/docs/projects/learn-more?hl=ja
［4］ Identity Platform と Firebase Authentication の違い，ID プラットフォーム，Google，2023 年 12 月．
https://cloud.google.com/identity-platform/docs/product-comparison?hl=ja
［5］ Cloud Firestore，Firebase ドキュメント，Google，2024 年 1 月．
https://firebase.google.com/docs/firestore?hl=ja
［6］ 使用量と上限―セキュリティルール，Firebase ドキュメント，Google，2024 年 1 月．
https://firebase.google.com/docs/firestore/quotas?hl=ja#security_rules
［7］ Cloud Storage for Firebase，Firebase ドキュメント，Google，2024 年 1 月．
https://firebase.google.com/docs/storage?hl=ja
［8］ Cloud Firestore による機能強化― Firebase Cloud Storage セキュリティ ルールで条件を使用する，Firebase ドキュメント，Google，2024 年 1 月．
https://firebase.google.com/docs/storage/security/rules-conditions?hl=ja#enhance_with_firestore
［9］ Cloud Functions トリガー，Cloud Functions，Google，2023 年 12 月．
https://cloud.google.com/functions/docs/calling?hl=ja
［10］ Firebase Hosting，Firebase ドキュメント，Google，2024 年 1 月．
https://firebase.google.com/docs/hosting?hl=ja
［11］ Firebase Cloud Messaging，Firebase ドキュメント，Google，2022 年 8 月．
https://firebase.google.com/docs/cloud-messaging?hl=ja
［12］ Google アナリティクス，Firebase ドキュメント，Google，2023 年 10 月．
https://firebase.google.com/docs/analytics?hl=ja
［13］ Firebase Machine Learning，Firebase ドキュメント，Google，2023 年 10 月．
https://firebase.google.com/docs/ml?hl=ja
［14］ Firebase A/B Testing，Firebase ドキュメント，Google，2024 年 1 月．
https://firebase.google.com/docs/ab-testing?hl=ja
［15］ Cloud Functions バージョンの比較，Cloud Functions，Google，2023 年 12 月．
https://cloud.google.com/functions/docs/concepts/version-comparison?hl=ja
［16］ Cloud Functions の関数で App Check の適用を有効にする，Firebase ドキュメント，Google，2024 年 1 月．
https://firebase.google.com/docs/app-check/cloud-functions?hl=ja#node.js-2nd-gen_1
［17］ 最大インスタンス数を構成する―最大インスタンス数の上限の設定，Cloud Functions，Google，2023 年 12 月．
https://cloud.google.com/functions/docs/configuring/max-instances?hl=ja#setting_maximum_instances_limits
［18］ Cloud Functions の料金，Cloud Functions，Google．
https://cloud.google.com/functions/pricing?hl=ja
［19］ 署名付き URL と署名付き Cookie の概要，Cloud CDN，Google，2023 年 1 月．
https://cloud.google.com/functions/pricing?hl=ja

※ 上記注及び参考文献に含まれる URL のリンク先は2024年2月1日時点での存在を確認．

執筆者紹介 宮 下 洋（Hiroshi Miyashita）
1998年日本ユニシス(株)入社，2004年ユニアデックス(株)転籍．2012 年まで JP1 を用いた運用管理システムの構築に従事．2013 年より日本ユニシス(株)の総合技術研究所に出向し，AI 研究に取り組む．2017 年から国立情報学研究所に出向し，researchmap.V2 に AI 機能を実装．2018 年から新規ビジネス企画に従事．