令和 3 年度　卒業論文

Firebase を活用した Web アプリケーション のバックエンド開発手法

近畿大学工学部　情報学科

情報物理研究室

1810990061　高尾　玲

第 1 章　はじめに

1.1　研究の背景：クラウドサービスの今後と Firebase の需要

Firebase とは，モバイルにも対応した Web アプリケーションのバックエンド開発に必要な機能（1.ホスティング，2.ユーザ認証，3.データベース）を提供する開発者向けのクラウドサービスのことであり，現在は Google 社が提供している[1]．このようなサービスは BaaS または mBaaS と呼ばれ，開発者は Firebase を利用する事で，自身でアプリケーションに必要な Web サーバやデータベースなどのバックエンド部分の開発を行わなくても，Firebase の API を通じて必要な機能を利用する事ができるようになる．つまり，Firebase を利用する事でバックエンド部分の開発を省略し，効率的にアプリケーション開発を進める事ができるようになるという事だ．

この Firebase のようなネットワーク経由でデータやソフトウェアを提供するサービスのことをクラウドサービスという．利用者はクラウドサービス事業者が運用するコンピュータの処理能力を必要な分だけ利用する事ができ，導入する事で業務の効率化やコストダウンを図る事が可能となる[2]．近年では企業におけるクラウドサービスの利用が増加傾向にあり，2019 年では 64.7%もの企業がクラウドサービスを導入している[3]．また，クラウドサービスの導入による効果があったと回答した企業は 85.5%となっており，導入する事で効果を得やすい事がわかる．ファイル保管やデータ管理，認証システムなど，多種多様なサービスが存在しており，今後はさらに Firebase のようなクラウドサービスを導入する企業が増加するだろう．

近年では，金融系のシステム開発において，従来の開発方法から革新的なデジタルテクノロジーである API を活用した開発手法への変化が起きており，API の連携による金融サービスの高度化が加速的に進んでいる．この状況に伴い，金融以外の業種でのアプリケーション開発方法も API を活用した開発へシフトしていくと考えられる[4]．このような事からも，API を提供している Firebase のようなクラウドサービスは今後のアプリケーション開発において主流の手法になるのではないだろうか．

情報物理研究室では認知行動療法や高次脳機能障害者のリハビリを支援する Web アプリケーションの開発に取り組んでおり，今年度も研究室の数名のメンバーがそれぞれアプリの開発に取り組んでいる．そして開発されたアプリケーションは様々なユーザーに利用されることになるのだが，今年度はユーザーごとに各アプリケーションを利用して得られた結果を確認できるようにしたい，という意見が出た．しかし，現状ではそれぞれのユーザーが各メンバーの開発したアプリケーションを利用する事で得られた結果を保存しておく事ができず，そもそも各メンバーはそれぞれ違った開発に取り組んでいるため，たとえ結果を保存できるようになったとしても，その得られた結果をまとめて確認する事ができなくなっている．

このような背景から，本研究にて Firebase について学習し，Firebase の持つ機能を活用した Web アプリケーションのバックエンド開発手法についてのまとめを作成する，という方針に決まった．

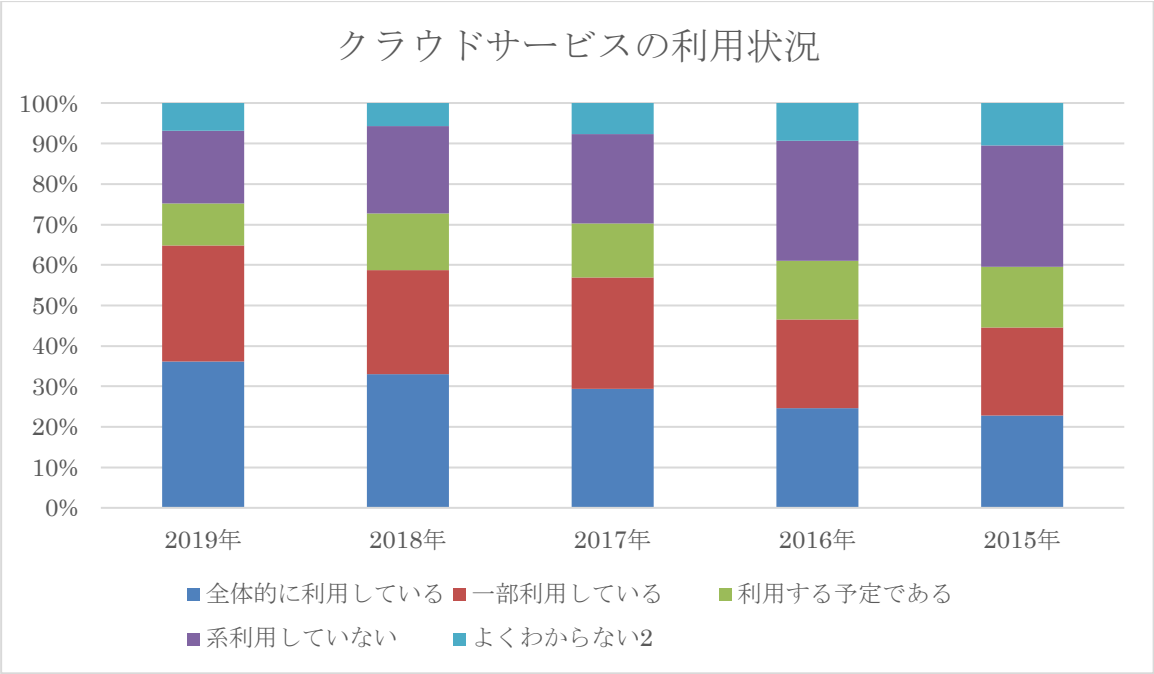

図 1　クラウドサービスの利用状況[3]

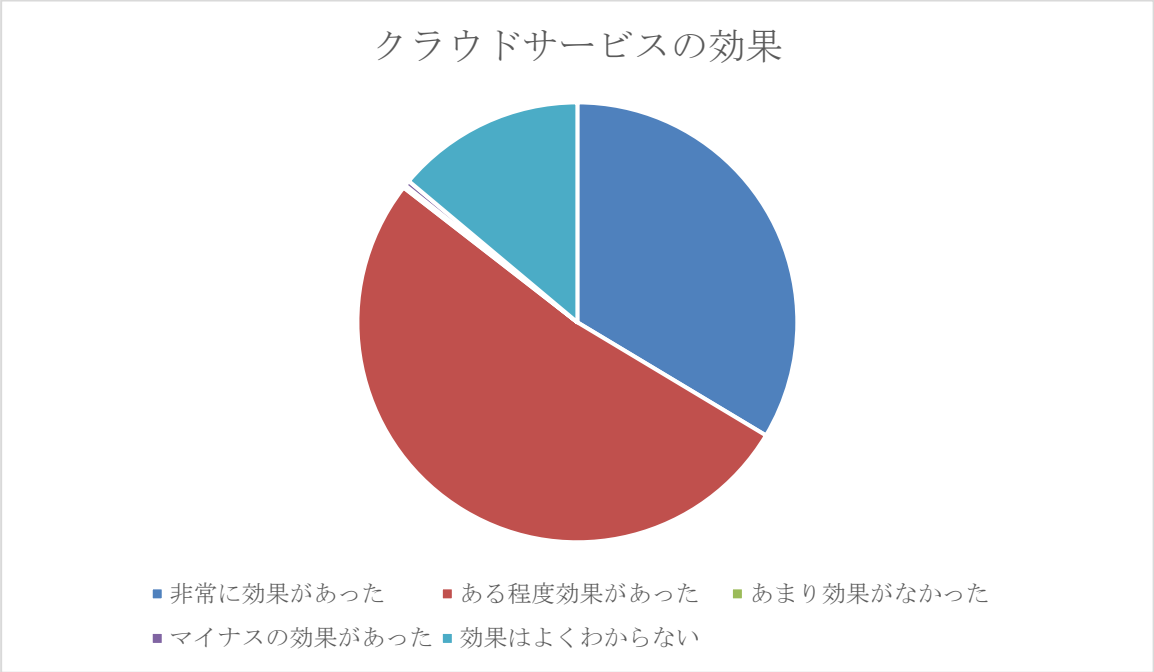

図 2　クラウドサービスの効果[3]

1.2　研究の目的

Firebase の持つ機能や JavaScript について学習し，ユーザ認証やデータベースの作成といった部分のコーディングや Firebase コンソールの利用方法といった，Web アプリケーションのバックエンド部分の開発に必要なことについてのまとめを作成する．また，Firebase を認知行動療法 3 コマまんがアプリに組み込む．

1.3　代表的な mBaaS

ここでは，Firebase 以外の代表的な mBaaS についていくつか記述する．

1.3.1　AWS Mobile Hub

AWS Mobile Hub とは，AWS が提供しているサービスをまとめて設定できる機能である[5]．データベースには DynamoDB，プッシュ通知は Amazon SNS などを利用する事ができる．しかし，詳細な設定は AWS 上で行う必要があるため，AWS に慣れておく事が求められる．

1.3.2　Azure App Service - Mobile Apps

Web サイトを構築するための Microsoft 製クラウドプラットフォームである[6]．Microsoft 製のため関連するサービスとの親和性が高く，ActiveDirectory との認証やオンプレミスデータとの連携といった強みを持っている．しかし，Azure Web Apps での Web サービスの開発には JavaScript を使用する事ができない．

1.4　PHP フレームワーク Laravel との比較

本研究では，当初は Web アプリケーションの開発に Laravel を利用することを検討していた．Laravel とは，PHP で書かれた Web アプリケーションフレームワークのことである[7]．同じ PHP フレームワークである Symphony をベースに開発され，2011 年にリリースされている．特徴としては，MVC モデルの採用と拡張性・自由度の高さが挙げられる．MVC モデルとは，処理を Model（データ処理），View（画面表示），Controller（全体制御）の 3 つに分類し，それぞれで開発を進める開発方法のことである．MVC モデルを採用する事で，処理の内容と記述する場所が明確になるため学びやすく扱いやすい開発を行う事ができる．また，Laravel が持つ機能は比較的多いため拡張性と自由度が高く，プログラムを自由に記述する事ができる．しかし，この自由度の高さによってコードの複雑化を招きやすいというデメリットがあり，プログラムの内容を研究室のメンバーで共有する事が難しくなる可能性がある．このような特徴を持つ Laravel に対し，Firebase は Firebase コンソールに記載されているスクリプトを HTML に記述するだけで Firebase を組み込む事ができ，メンバー同士でプログラムの内容を共有しやすい．また，Firebase の持つリアルタイム型のデータベースにより不安定なネットワーク環境下でも高速なデータベースへのアクセスを実現できるという点が，メンバーの開発する Web アプリケーションにマッチしていると考えられる．

このような背景から，本研究では Laravel の採用を見送り，Firebase を採用するに至った．

1.5　Firebase を利用した Web アプリケーション

今後も市場の拡大と競争の激化が予想される mBaaS の 1 つである Firebase だが，私たちにとって身近な環境にも Firebase が利用されている Web アプリケーションが存在する．ここでは，Firebase を活用した Web アプリケーションの開発事例として，「Komerco」を紹介する．

「Komerco」とは料理に関する商品を取り扱うオンラインマルシェとして，2018 年に Cookpad がリリースした Web アプリケーションである[8]．ユーザはこのアプリを利用することで調味料や食材，料理道具やカトラリーなど，料理に関する商品を「Komerco」を通してクリエイターから購入する事ができるうえ，クリエイターとして商品を出品することもできる．

この「Komerco」ではバックエンドのほぼ全てを Firebase で運用しており，現在も Firebase をより有効に活用できるように試行錯誤を重ねているようだ．現在では，「ユーザ」，「商品」，「ショップ」などのコレクションを Firestore で管理し，クライアントサイドジョインを採用することで商品のリスト表示を可能にしている[9]．クライアントサイドジョインとはクライアント側でコレクションを別々に取得して結合する手法のことであり，Firestore のストレージ消費を抑え冗長化を防ぐ事ができる．

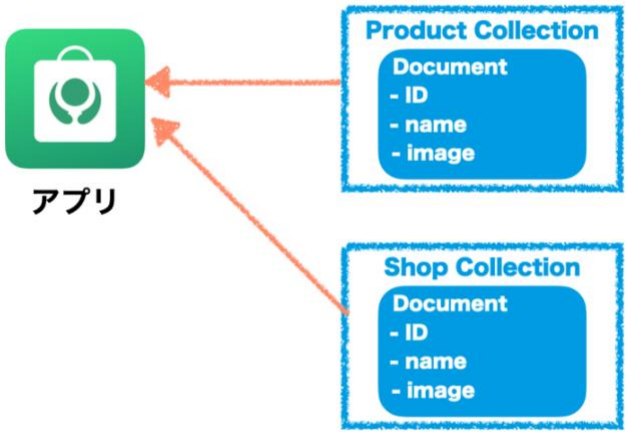

図 3　Komerco のクライアントサイドジョイン

第 2 章　準備

この章にて，本論文で扱う専門用語について記述し説明しておく．

2.1　Firebase とは

Firebase とは，モバイルや Web アプリケーションのバックエンド開発が可能なプラットフォームを提供する開発者向けクラウドサービスのことであり，クラウドサービスの形態としては mBaaS または BaaS に位置付けられている．2011 年に Firebase 社がサービスを開始し，2014 年に Google 社が買収，現在はそのまま Google 社がサービスの提供を行っている．

Firebase の最大の特徴は，リアルタイム型のデータベースを使用する事ができる点である．一般的なデータベースとは違い，Web アプリケーションがデータベースのローカルコピーを保持し，データベースへの書き込みが発生した場合，そのローカルコピーに書き込まれる． そのローカルコピーの変更は自動的にサーバ上のデータベースと同期される．このローカルコピーはオフラインでもアクセスする事ができるため，安定したインターネット環境でなくても高速な応答を得る事ができる．また，オフラインで発生したローカルコピーへの変更は，サーバと接続できた時点で自動的に同期される[10].

2.2　BaaS/mBaaS とは

BaaS とは，Backend as a Service の略であり， Web アプリケーションのうちバックエンドの部分の機能を提供するサービスのことを指す[11]．また、モバイル向けの BaaS のことを mBaaS といい，mobile Backend as a Service の略である．これらのサービスを利用することでサーバの設計と運用・管理を省略する事ができるようになる．近年ではスマートフォンのようなモバイルデバイスの普及が拡大しており，mBaaS を用いた Web アプリケーション開発の需要は加速するだろう[12].

2.3　バックエンドとは

バックエンドとは，システムやソフトウェアを構成する要素のうち，ユーザーが直接触れない機能や処理を行う要素のことを指す[13]．Web アプリケーションにおいては，ユーザーが文字などを入力する部分であるフロントエンドに対して，ユーザーによって入力された内容のデータ処理やデータベースへの保存などを行う部分のことを指す．

2.4　Firebase Authentication

Firebase Authentication とは，Firebase が提供している機能のうち，ユーザー認証に関する機能のことを指す[14]．ユーザー認証とは，Web アプリケーションアプリの管理者がそのアプリを利用するユーザーがログインしているか、もしくはログアウトしているかを確認する機能のことである．管理者は，ユーザーのログイン方法について，メールアドレスを利用する方法やソーシャルログインを利用する方法に設定する事ができる．Firebase はユーザー認証以外にもデータベースやストレージに関する機能を提供しているが，それらの機能を持たせたアプリケーションを利用する際にはそのアプリにログインしておく事が前提となるため，ユーザー認証機能の搭載は必須であると考えられる．

また，Firebase Authentication を利用しユーザーを管理する事で，悪意のあるユーザーによるデータの書き換えなどの問題もある程度防止する事ができる．

2.5　ソーシャルログイン

ソーシャルログインとは，ユーザーが既に持っている SNS アカウントを使って Web サイトや Web アプリケーションにログインできる機能のことを指す．利用できる SNS は Google や Twitter，Facebook などがあるが，本研究では Google を利用している．また，Web アプリケーションが持つユーザ認証において利用されるソーシャルログインには OAuth や OpenID Connect が多く用いられている[15].

2.6　Firebase Realtime Database

Firebase Realtime Database とは，Firebase が提供している機能のうち，データベースに関する機能のことを指す[16]．この機能を利用することで，Web サイトや Web アプリケーションの開発者は自身でデータベースサーバーを用意する必要がなくなるため，開発者のサーバー構築にかかるコストを削減できる．

Firebase Realtime Database は，JSON 形式でデータを管理する NoSQL データベースである．リレーショナルデータベースではないため複雑なデータベース処理には向かないが，シンプルかつ大量なデータを高速に処理する必要がある場合には適したデータベースであると言える．

2.7　NoSQL データベース

NoSQL データベースとは，データの形が一つではなくいくつかの形式が同時に存在する可能性があるデータに対して，テーブル構造を固定しなくてもそのままの形式で格納する事ができるデータベースである．格納するデータに対して一貫性の制限を一部解除することで，大容量のデータ格納，データへの高速なアクセス，柔軟なデータモデルを必要とするアプリケーションに対して最適化されている．データの格納と取得の方法がリレーショナルデータベースとは異なり，リレーショナルデータベース管理システム以外のデータベース管理システムのことを総称して NoSQL と呼ばれている[17][18].

2.8　リレーショナルデータベース

リレーショナルデータベースとは，データを事前に定義された列と行を持つテーブルに格納して管理し，それぞれのテーブルを互いに関連づけることで複雑なデータの関連性を扱うことができるデータベースである[19]．テーブルの行をレコード，列をフィールドと呼び，レコードはキーと呼ばれる重複しない ID を持っている．

2.9　Firebase　セキュリティルール

Firebaseセキュリティルールとは，データベースに保存されているデータへのアクセスを制御できる機能のことを指す[20]．デフォルトの状態では，データへのアクセスを制限するロックモードか全てのユーザにデータへのアクセスを許可するテストモードのどちらかを選択できる．Realtime Database ルールでは，JavaScript に似た JSON 構造の構文を使用されており，任意の構造に書き換えることでアプリケーションに適格なルールを作成する事ができる．

第 3 章　本論

3.1　Firebase のセッティングと使い方

3.1.1　Firebase でプロジェクトを作成する

ブラウザから https://firebase.google.com にアクセスし「使ってみる」と書かれたボタンをクリックする．その後，Google アカウントでログインしていなければ右上のアイコンからログインする．
「プロジェクトを作成」または「プロジェクトを追加」と書かれたボタンをクリックする．するとプロジェクト名の入力欄が出現するため，任意のプロジェクト名を入力し，「続行」と書かれたボタンをクリックする．
次に，Google アナリティクスの説明や構成についての文章が記述されているページに切り替わるため，それらの設定と同意を完了する．
「プロジェクトを作成」と書かれたボタンがもう一度出現するため，このボタンをクリックするとプロジェクトが作成される．また，作成したプロジェクトは Firebase コンソールから確認できる．

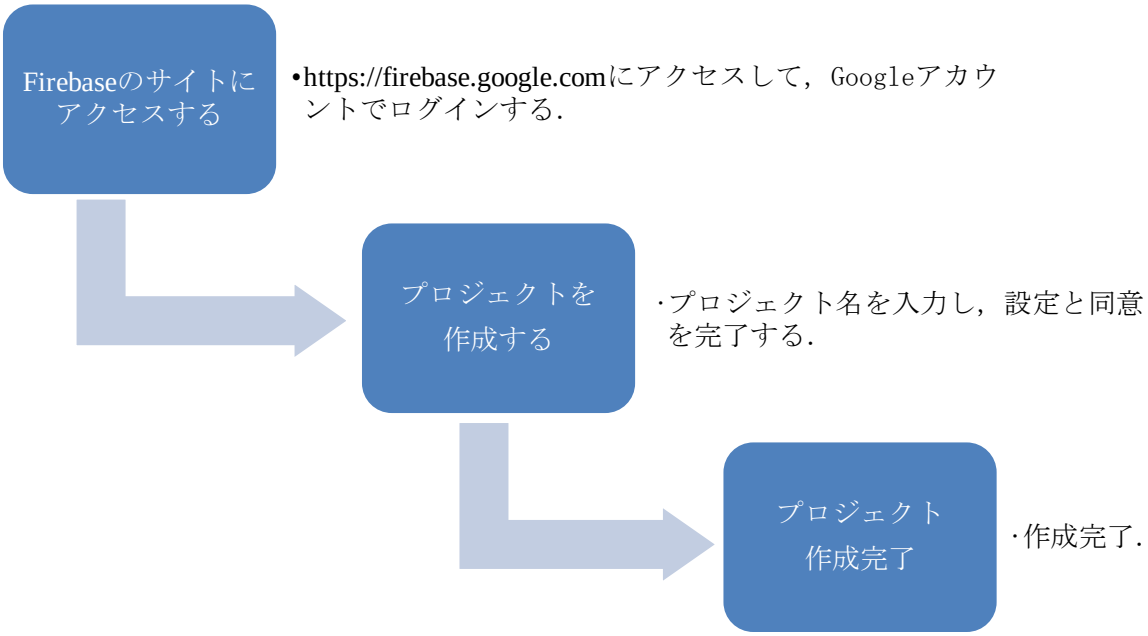

図 4　プロジェクト作成までの流れ

3.1.2.　アプリを追加する

Firebase で作成したプロジェクトのトップページの左上にある「+アプリを追加する」ボタンをクリックする．任意のプラットフォームを選択できるようになるため，本研究では Web を選択する．

アプリに任意の名前をつけて登録し，SDK は「<script>タグを使用する」を選択する．名前をつけるタイミングで Firebase Hosting の設定についての項目があるが，本研究では buturi.heteml.net というサーバを使用しているため設定しない．

これらの設定が終了した後コンソールに進むとプロジェクトにアプリが追加されている．

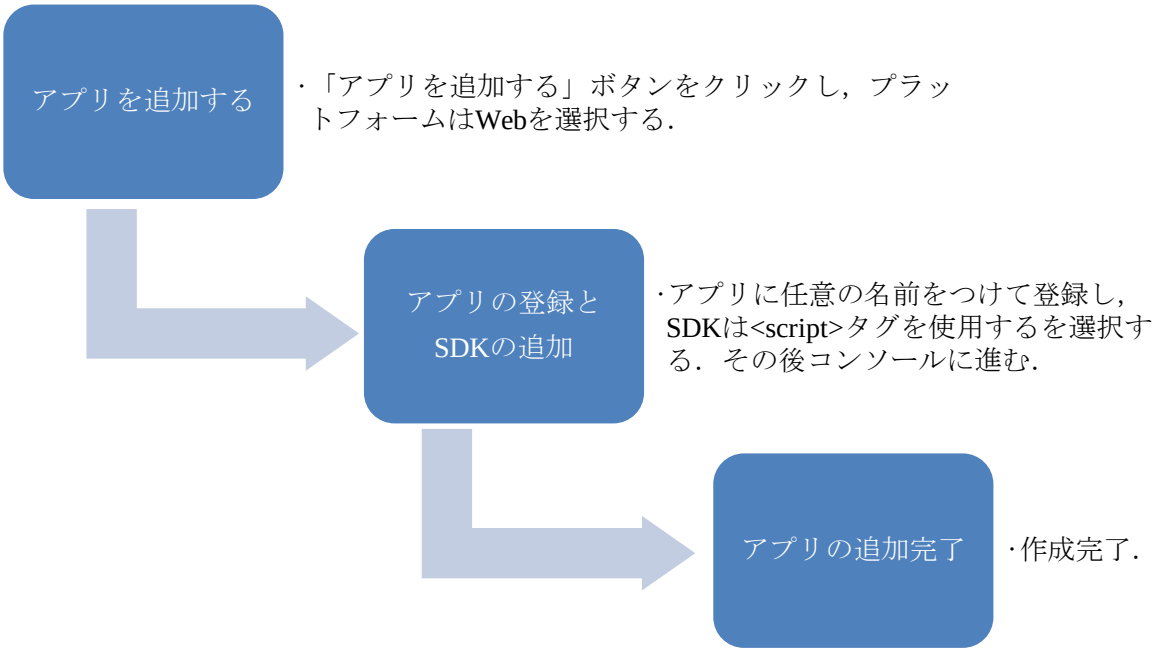

図 5　アプリ追加までの流れ

3.1.3. Firebase を HTML ファイルに組み込む

Web ページで Firebase を利用する方法は大きく 2 つ存在している．1 つは，HTML ファイルの中に<script>タグを使用してスクリプトを読み込ませて利用するという方法であり，もう 1 つは，Node.js のプロジェクトとして作成し，npm で Firebase の API をインストールして使用するという方法である．今研究では HTML ファイルを多く扱うため，前者の方法を採用している．

作成したプロジェクトのトップページの左上にあるハンバーガーメニューをクリックし，出現したツールボタンをクリックしてプロジェクトの設定を開く．
「全般」のページを下のほうにスクロールすると「SDK の設定と構成」という箇所があるため，ここで CDN を選択する．すると<script>タグで囲まれたプログラムが記載されているため，このプログラムをコピーしておく．また，このプログラム中で定義されている firebaseConfig の内容は作成したアプリごとに違っているため注意が必要である．
コピーしたスクリプトを任意の Firebase を組み込みたい HTML ファイルのプログラムにペーストしておく．
Firebase Authentication をセキュリティ面で安全に利用するためには，プログラムをサーバ上にアップロードする必要がある．そのためスクリプトをコピーしたプログラムをサーバ上にアップロードしておく．
また，アップロード先のサーバをあらかじめ Firebase で承認しておく必要がある．Firebase コンソールから Authentication のページを開き，Sign-in-method のページに移動する．承認済みドメインについての項目にある「ドメインを追加」をクリックする．するとサーバのドメイン名を入力する箇所が出現するため，入力して追加する．

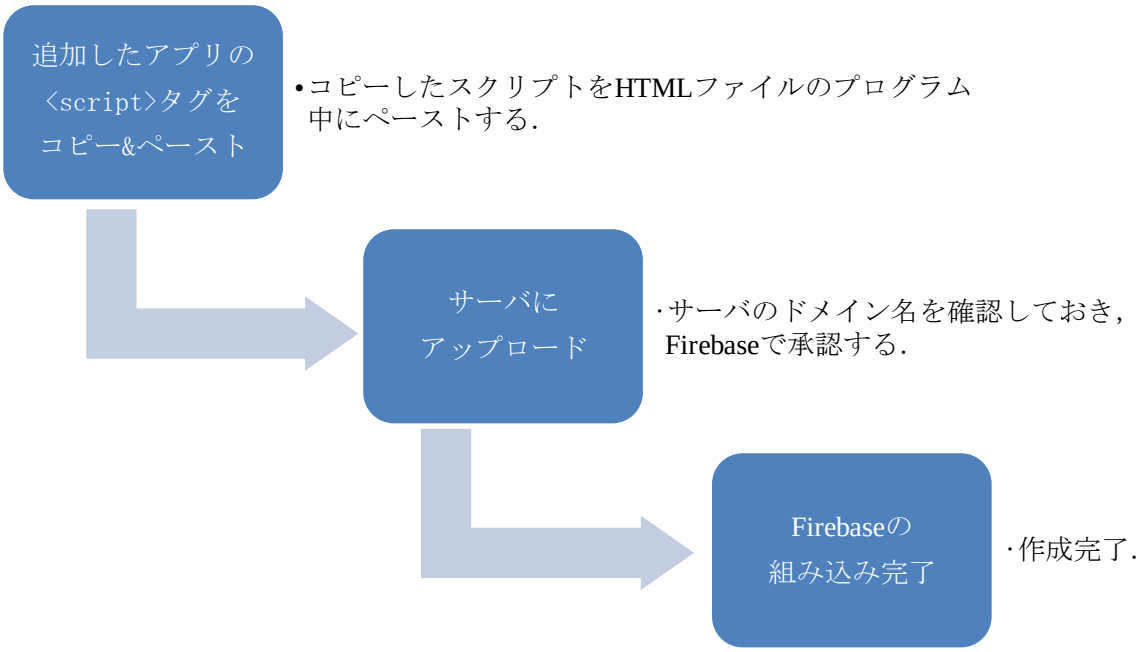

図 6　Firebase を HTML ファイルに組み込むまでの流れ

3.2　Firebase Authentication の利用

3.2.1.　アプリケーションの説明

作成した Web アプリケーション：http://buturi.heteml.net/student/2021/takao/auth-sample01.html

本研究にて，Firebase のユーザ認証の機能である Firebase Authentication を利用し，ユーザが Web アプリケーションにログインする際にソーシャルログインを使用できるプログラムを制作した．本 Web アプリケーションにおいて，ユーザは自身のメールアドレスもしくは Google アカウントを利用してログインできる．また，Web アプリケーションにログインしているユーザのアカウントは Firebase 上で確認する事ができる．

3.2.2.　アプリケーションの内容

Firebase Authentication を利用したユーザ認証を行う Web アプリケーションのソースコードの内容を説明する．

3.2.2.1　auth-sample01.html　1〜7 行目

```
  <!DOCTYPE html>
<html lang="ja">
<head>
 <title>Firebase sample 01</title>
 <script src="https://cdn.firebase.com/libs/firebaseui/3.5.2/firebaseui.js"></script>
 <link type="text/css" rel="stylesheet"
href="https://cdn.firebase.com/libs/firebaseui/3.5.2/firebaseui.css" />
</head>
```

上記が 1〜7 行目の内容にあたる．ここでは HTML の基本的なプログラムが記述されている．
5 行目にある firebaseui.js は Firebase Authentication SDK に基づいて作成されたライブラリのことであり，Web アプリケーションで使用されるドロップイン UI フローを提供している．

3.2.2.2　auth-sample01.html　8～14 行目

```
<body>
  <h1>Firebase Sample 01</h1>
  <div id="auth"></div>
  <script src="https://www.gstatic.com/firebasejs/9.1.2/firebase-app-compat.js"></script>
  <script src="https://www.gstatic.com/firebasejs/9.1.2/firebase-database-
compat.js"></script>
  <script src="https://www.gstatic.com/firebasejs/9.1.2/firebase-auth-compat.js"></script>
  <script src="https://www.gstatic.com/firebasejs/ui/5.0.0/firebase-ui-
auth__ja.js"></script>
```

上記が 8～14 行目にあたる．ここでは HTML で出力される部分と Firebase のライブラリのロードについての部分についてのプログラムが記述されている．

11～14 行目は Firebase を利用するにあたって必要な機能のライブラリをロードするプログラムである．11～13 行目にある compat とはバージョン 9 の互換ライブラリのことであり，バージョン 8 との互換性の問題を回避できる．また，10 行目については後ほど記述する．

3.2.2.3　auth-sample01.html　16～27 行目

```
<script>
  const firebaseConfig = {
    apiKey: "apiKey",
    authDomain: "authDomain",
    databaseURL: "databaseURL",
    projectID: "projectID",
    strageBucket: "atrageBucket",
    messagingSenderID: "messagingSenderID",
    appID: "appID",
    measurementID: "measurementID"
  };
  firebase.initializeApp(firebaseConfig);
```

上記が 16～27 行目にあたる．ここでは Firebase コンソールに記載されているソースコードと Firebase の初期化についてのプログラムが記述されている．

17～26 行目は 3.1.3 で記述した通り，Firebase コンソールで確認できるソースコードをコピー&ペーストしたものである．また，セキュリティの都合上この部分のプログラムの内容は伏せている．

27 行目は Firebase の初期化についてのプログラムである．Firebase オブジェクトである「initializeApp」メソッドを呼び出し，引数には Firebase の設定情報を指定することで Firebase のセットアップが完了する．

3.2.2.4　auth-sample01.html　30～51 行目

```
const ui = new firebaseui.auth.AuthUI(firebase.auth());
 const uiConfig = {
  callbacks: {
   signInSuccessWithAuthResult: function(authResult, redirectUrl) {
    return true;
   },
  },
  signInFlow: 'popup',
  signInSuccessUrl: 'auth-sample02.html',
  signInOptions: [
   firebase.auth.EmailAuthProvider.PROVIDER_ID,
   firebase.auth.GoogleAuthProvider.PROVIDER_ID,
  ],
  tosUrl: 'sample01.html',
  privacyPolicyUrl: 'auth-sample01.html'
 };
 ui.start('#auth', uiConfig);
 </script>
</body>
</html>
```

上記が 30～51 行目にあたる．ここではユーザ認証についてのプログラムが記述されている．

30 行目は FirebaseUI の初期化についてのプログラムである．

32～36 行目はログイン成功時のコールバックについての設定を指定するプログラムである．ここでは，ログインに成功すると return 文で true を返す．

37～47 行目は，ユーザ認証の内容についてのプログラムである．

signInFlow では，ログインに使用するアカウントを選択する画面の動作を指定する事ができる．ここでは popup を使用しており、アカウント選択画面がポップアップして現れるようになっている．

signInSuccessUrl では，ログインに成功した後に表示するページを指定する事ができる．ここでは auth-sample02.html を指定している．また，このページについては後ほど説明する．

signInOptions では，ユーザがログインするために使用するプロバイダを指定する事ができる．ここではメールアドレスと Google アカウントを指定している．

tosUrl，privacyPolicyUrl を指定する事で，利用規約やプライバシーポリシーについてのページを指定する事ができる．

48 行目は，任意の箇所に FirebaseUI を適応させるプログラムである．ここでは 31～47 行目の uiConfig の内容を 10 行目の div 要素に適応させるよう指定している．

図 7　ログイン方法を選択するページ

図 8　「メールでログイン」を選択したページ

3.2.2.5　auth-sample02.html　8〜11 行目

ここからは auth-sample01.html の 38 行目にログイン後のアクセス先である auth-sample02.html について説明する．一部 auth-sample01.html のプログラムと重なるため，その箇所は省略している．

```
<body>
 <h1>Firebase Sample 02</h1>
 <p id="login"></p>
 <button onclick="signOut()">signout</button>
```

上記が 8〜11 行目にあたる．ここではログインしたユーザの名前を含めた文章の表示とサインアウトボタンについてのプログラムが記述されている．詳しい内容は後ほど記述する．

3.2.2.6　auth-sample02.html　32〜40 行目

```
firebase.auth().onAuthStateChanged(user => {
  if (user) {
    console.log(user);
    const signOutMessage = `
    <p>Hello, ${user.displayName}!</p>
        `;
    document.getElementById('login').innerHTML = signOutMessage;
  }
 });
```

上記が 32〜40 行目にあたる．ここではログインしたユーザの名前を含めた文章の表示についてのプログラムが記述されている．

32 行目の onAuthStateChanged を使用する事で，ログインしているユーザの情報を取得する際に Auth オブジェクトの状態を確認できる．

33 行目では if 文を使用しており，ユーザがログインしていれば「Hello，ユーザの名前」と表示する事ができる．ユーザの名前は 36 行目の$(user.displayName)によって取得できる．user.displayName とは，ログインしたユーザのプロフィール情報であり，このユーザの名前は 33 行目に console.log(user);と記述する事でデベロッパーツールのコンソールから確認する事ができる．

38 行目では，35〜37 行目の内容を HTML で表示している．innerHTML を使用することで signOutMessage の内容を HTML で表示する事ができる．

3.2.2.6　auth-sample02.html　42～58 行目

```
function signOut() {
  firebase.auth().onAuthStateChanged(user => {
   firebase
    .auth()
    .signOut()
    .then(() => {
     alert('signout');
     location.href = "./auth-sample01.html";
    })
    .catch((error) => {
     alert('signout-error');
     location.reload();
    });
  });
 }


 </script>
```

上記が 42～58 行目にあたる．ここでは Web アプリケーションのサインアウト機能についてのプログラムが記述されている．

11 行目で onclick プロパディを使用しており，サインアウトボタンをクリックすると 42～56 行目の signOut()が起動するようになっている．サインアウトが成功すれば「signout」というアラートが出現し，ログイン前のページに移動するようになっている．また，エラーが発生すれば「signout-error」というアラートが出現し，同じページをリロードするようになっている．

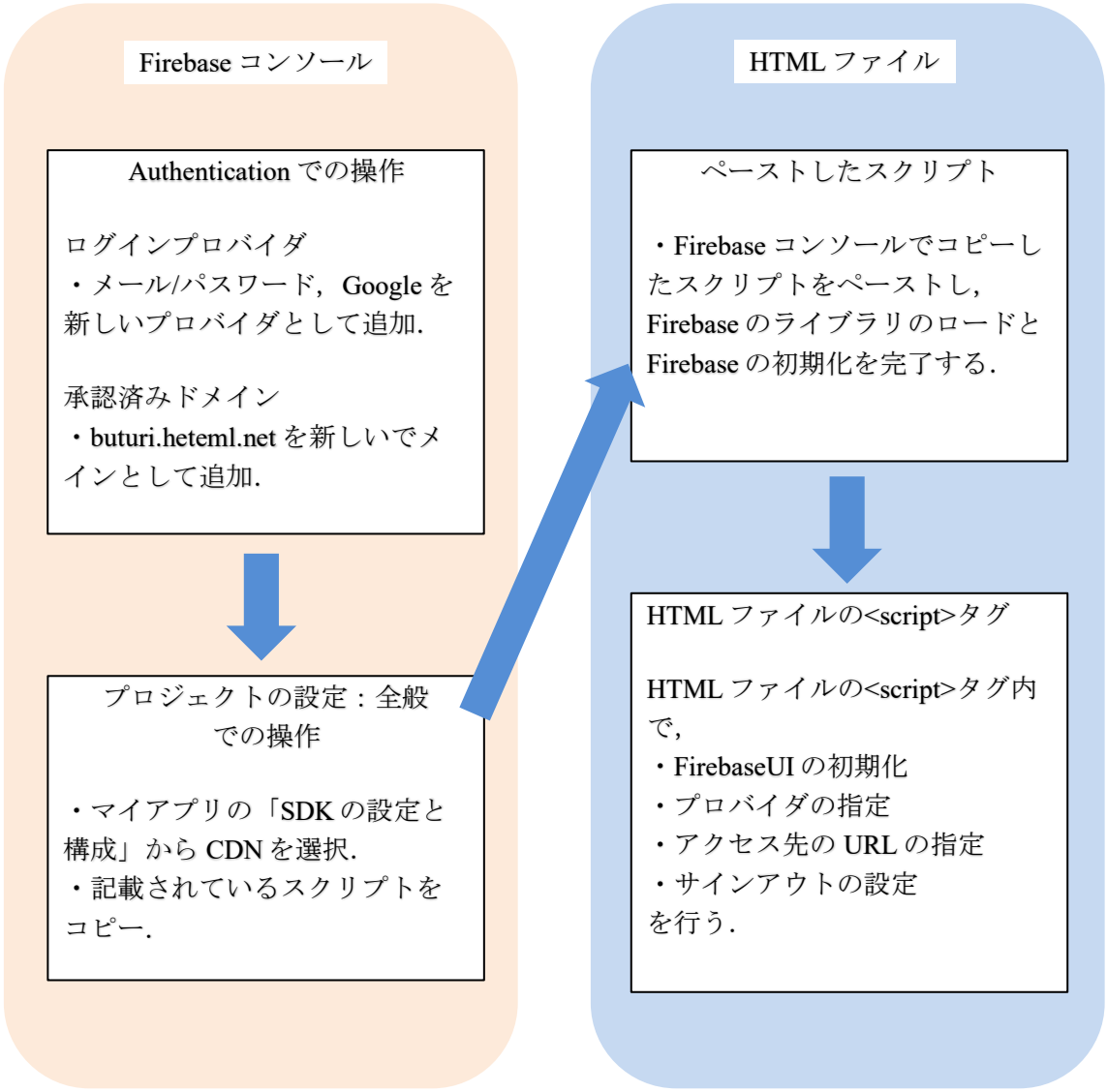

図 9　Authentication 導入の際のフローチャート

3.3　Firebase Realtime Database の利用

3.3.1.　アプリケーションの説明

作成した Web アプリケーション：
http://buturi.heteml.net/student/2021/takao/Firebase_realtimedatabase_2.html

本研究にて，Firebase の基本的なデータベースである Firebase Realtime Database を利用し，管理者が Web アプリケーション上でデータベースに格納されているデータを確認できるプログラムを制作した．本 Web アプリケーションにおいて，管理者はデータベースが持つデータを Web ページ上または Firebase 上で確認する事ができる．

3.3.2.　アプリケーションの内容

Firebase Realtime Database を利用したデータベースのデータの出力を行う Web アプリケーションのソースコードの内容を説明する．

図 10　HTML で表示されたデータベースのデータ

3.3.2.1　Firebase_realtimedatabase_2.html　8～26 行目

ここからは Firebase_realtimedatabase_2.html について説明する．一部 auth-sample01.html のプログラムと重なるため，その箇所は省略している．

```
<body>
    <h1> Firebase Realtime Database</h1>
  <div id="database"></div>

  <script type="module">
    import { initializeApp } from "https://www.gstatic.com/firebasejs/9.1.3/firebase-app.js";
    import { getDatabase, ref, onValue } from
"https://www.gstatic.com/firebasejs/9.1.3/firebase-database.js";
    import { getAnalytics } from "https://www.gstatic.com/firebasejs/9.1.3/firebase-
analytics.js";

    const firebaseConfig = {
      apiKey: "apiKey",
      authDomain: "authDomain",
      databaseURL: "databaseURL",
      projectID: "projectID",
      strageBucket: "strageBucket",
      messagingSenderID: "messagingSenderID",
      appID: "appID",
      measurementID: "measurementID"
    };
```

上記が 8～26 行目にあたる．ここではデータベースの内容の HTML での表示と Firebase のライブラリのロードと Firebase の初期化についてのプログラムが記述されている．

10 行目はデータベースが持つデータを HTML で表示するプログラムである．詳細は後ほど記述する．

12～15 行目は Firebase を利用するにあたって必要な機能のライブラリをロードするプログラムである．module とはバージョン 9 以降の Firebase の持つ機能であり，Firebase JavaScript SDK をモジュールバンドラーの最適化機能と連携させる事で Firebase コードの力を減少させる事ができる．

17～26 行目は Firebase コンソールで確認できるソースコードをコピー&ペーストしたものである．また，セキュリティの都合上この部分のプログラムの内容は伏せている．

3.3.2.2　Firebase_realtimedatabase_2.html　28～29 行目

```
const app = initializeApp(firebaseConfig);
const db = getDatabase(app);
```

上記が 28～29 行目にあたる．ここでは Firebase の初期化についてのプログラムが記述されている．

28 行目では，Firebase オブジェクトである「initializeApp」メソッドを呼び出し，引数には Firebase の設定情報を指定することで Firebase のセットアップが完了する．

29 行目では，Firebase オブジェクトである「getDatabase」メソッドを呼び出し，引数には 28 行目で定義した「app」を指定することでデータベースを参照している．

3.3.2.3　Firebase_realtimedatabase_2.html　31～44 行目

```
let people = ref(db, 'people/');
      onValue(people, function(snapshot) {
       let dataHtml = "";
       if(snapshot){
        snapshot.forEach(function(child){
         let data = child.val();
         dataHtml += data.name + "["  + data.mail + "," + data.age + "歳]" + "<br>";
        });
        console.log(dataHtml);
        $('#database').html(dataHtml);
       }
      });


  </script>
```

上記が 31～44 行目にあたる．ここでは Firebase Realtime Database に格納されているデータを HTML 上に表示するプログラムが記述されている．

31 行目はデータベースに格納されているデータを参照するプログラムである．「ref」メソッド用いることで，29 行目に定義した「db」から「people」の参照を変数に取り出している．

32～38 行目では value イベントを使用し，「people」のデータリスト全体を単一のスナップショットとして返している．35 行目の「forEach」メソッドは与えられた関数を配列の各要素に対して 1 回ずつ実行する．本プログラムでは，「people」のデータリストを 33 行目で定義された「dataHtml」に 37 行目で指定した形で加算代入している．

39 行目では，この時点での「dataHtml」に代入されている値を確認するために「console.log()」を記述している．

40 行目では，「dataHtml」の内容を HTML で表示するためのプログラムを記述している．この「dataHtml」の値は，10 行目の div 要素の箇所に HTML 上で表示するようになっている．

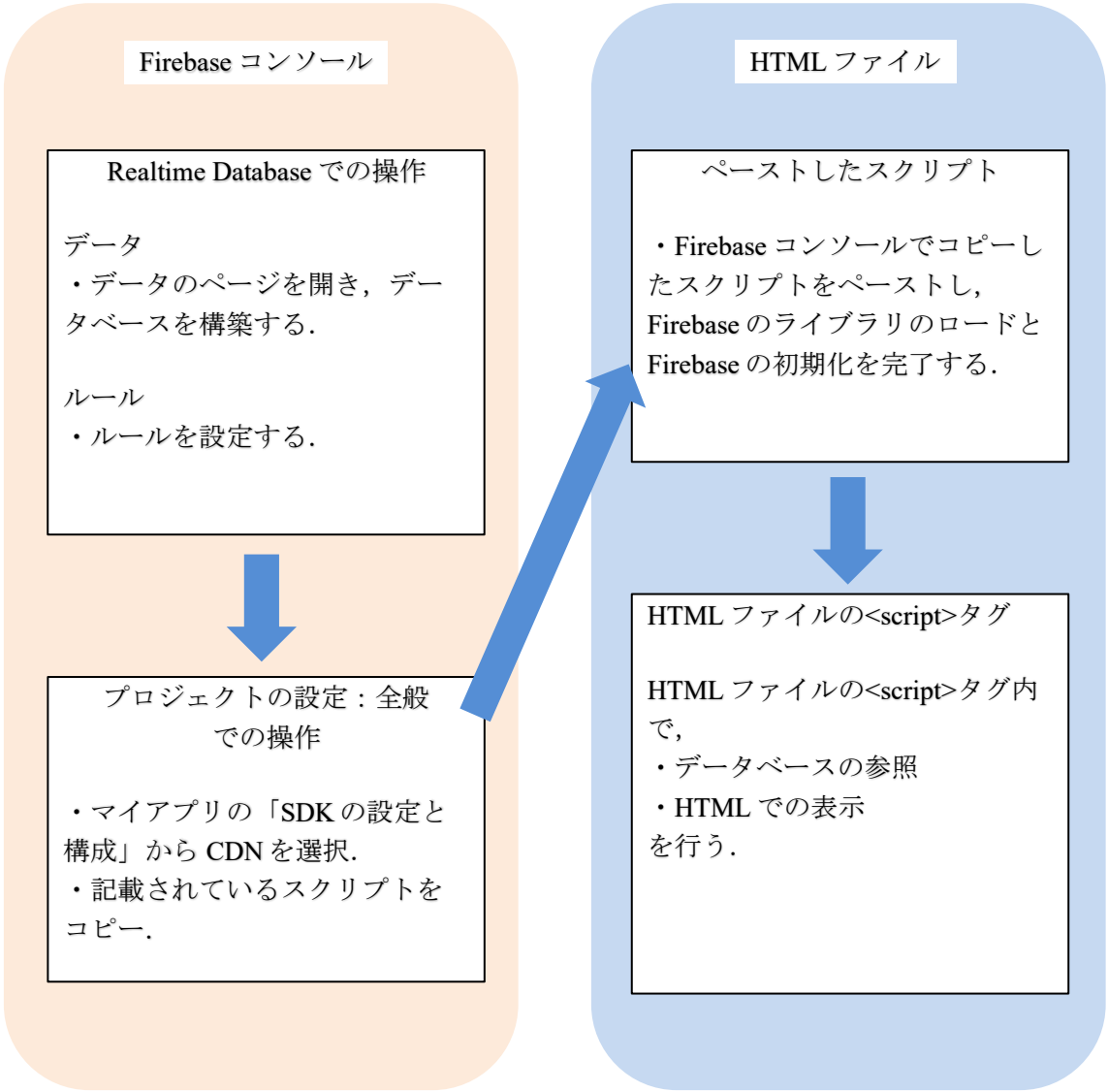

図 11　Realtime Database 導入の際のフローチャート

3.4　Firebase セキュリティルール

ここでは，Firebase セキュリティルール構成における一般的な脆弱性や独自のルールを確認してセキュリティを強化する方法，Web アプリケーションをデプロイする前に変更内容をテストする手順について記述する．

最初に Firebase でプロジェクトを立ち上げた際に，ルールを初期設定のままにしておくと期限切れになってしまい，Realtime Database が使えなくなってしまったため，その対処法も記述する．

3.4.1　Firebase セキュリティルールとは

Firebase セキュリティルールとは，悪意のあるユーザからデータを保護するために利用される．Firebase コンソールで，全てのユーザに対してアクセスを拒否するロックモードと，全てのユーザに対してアクセス権を付与するテストモードを選択する事ができ，開発後はより自由な構成が必要となる．

3.4.2　Firebase セキュリティルールの例

Firebase セキュリティルールの例をいくつか記述する．

3.4.2.1　オープンアクセス

全てのユーザがアクセスできるようになるルールである．しかし，Web アプリケーションをデプロイした後もこのルールのままだとセキュリティの面で危険なため，最終的にはルールを変更する事が推奨されている．

```
{
  // Allow read/write access to all users under any conditions
  // Warning: **NEVER** use this ruleset in production; it allows
  // anyone to overwrite your entire database.

  "rules": {
    ".read": true,
    ".write": true
  }
}
```

図 12　オープンアクセスのルール内容

3.4.2.2　認証されたユーザのアクセス

ログイン済みのユーザのみにアクセスを許可するルールであり，コーディング次第で特定のユーザのみアクセスを制限することも可能である．Web アプリケーションをデプロイした後はこのルールを適応させる事が望ましいと思われる．

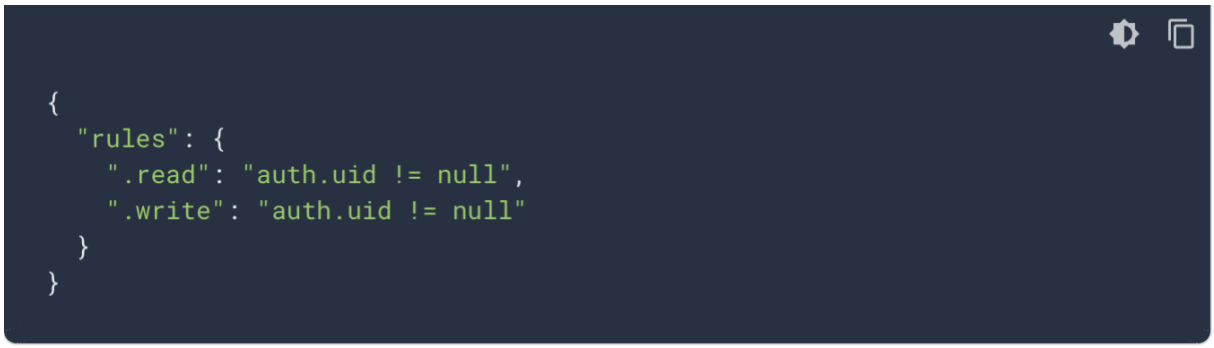

図 13　認証されたアクセスのルール内容

3.4.2.3　クローズドアクセス

全てのユーザに対して，データへのアクセスを拒否するルールである．Web アプリケーションの開発中における一般的なアプローチとされている．

図 14　クローズドアクセスのルール内容

第 4 章　結論

本研究では，Firebase Authentication と Firebase Realtime Database の概要と利用方法について学習しその内容について記述した．簡単なプログラムではあるが，実際に Web アプリケーションの制作を行う事で，Firebase の導入のしやすさや Web アプリケーションの管理・運用のしやすさなどの Firebase のメリットの部分を知る事ができた．しかし，初心者にとっては専門用語が多出する Firebase Documentation を利用した学習は少し難易度が高いというデメリットも身をもって体感した．このようなことから，本研究で行った「Firebase を活用した Web アプリケーションのバックエンドの開発手法」は初心者による Firebase の利用において有用なものになったのではないだろうか.

今後の課題として，認知行動療法 3 コマ漫画アプリに対して Firebase を組み込む事が挙げられる．この課題は本研究の目的の 1 つでもあったのだが，私の Firebase の学習に時間を多く費やしてしまい達成する事ができなかった．近年モバイルデバイスが加速的に普及していることから mBaaS の需要が今後さらに高まる事が予想されるため，mBaaS の 1 つである Firebase の価値はとても高いものになると考えられる．このようなことから，今後 IT 事業に携わっていく身としてさらに学習を進めた上でアプリへの組み込みに挑戦したい.

参考文献

[1]　Firebase とは　IT 用語辞典 e-Words　2021.9.13　https://e-words.jp/w/Firebase.html

[2]　クラウドサービスとは？　総務省
https://www.soumu.go.jp/main_sosiki/joho_tsusin/security/basic/service/13.html

[3]　企業におけるクラウドサービスの利用動向　総務省
https://www.soumu.go.jp/johotsusintokei/whitepaper/ja/r02/html/nd252140.html

[4]　新たなサービスのスピーディな提供を可能にするモバイル高速開発
青柳享　小林茂憲　小泉健　2017.3　https://jpn.nec.com/techrep/journal/g16/n02/pdf/160209.pdf

[5]　AWS Mobile Hub の機能　Amazon Web Service
https://docs.aws.amazon.com/ja_jp/aws-mobile/latest/developerguide/mobile-hub-features.html

[6]　App Service – Web アプリの構築とホスト　Microsoft Azure
https://azure.microsoft.com/ja-jp/services/app-service/#overview

[7]　Laravel とは何？Laravel の 11 個の特徴と使う際の注意点　Acrovision　2021.2.26
https://www.acrovision.jp/career/?p=2776

[6]　Komerco とは　Komerco　https://komer.co/about

[9]　Komerco と Firebase の話【前編】　Komerco　高橋　2021.4.21
https://techlife.cookpad.com/entry/2021/04/21/110000

[10]　Firebase とは？Google 社の強みを活かしたメリットや機能をご紹介　udemy　2020.4.14
https://udemy.benesse.co.jp/development/system/what-is-firabase.html

[11]　MBaaS(エムバース)とは？メリット・デメリット-モバイルアプリ開発サービスの特徴
SMARTCAMP　2020.12.22　https://boxil.jp/mag/a3651/

[12]　スマートフォンユーザの依存傾向を考慮した機能制限アプリケーションの開発
渡邊宏尚　水野凌太郎　土田栞　皆月昭則　2015

[13]　フロントエンドとバックエンドとは？意味の違いとエンジニアの仕事
WEB STAFF　2020.1.24
https://www.webstaff.jp/guide/trend/フロントエンドとバックエンドとは/

[14]　サーバーレス開発プラットフォーム Firebase 入門　掌田津耶乃　2019.4.29　P.47～P.160

[15]　OAuth/OpenID Connect 実装におけるセキュリティ状況の調査
菊田翼　真木康太郎　細谷竜平　八代哲　斎藤孝道　2019

[16]　サーバーレス開発プラットフォーム Firebase 入門　掌田津耶乃　2019.4.29　P.161～P.234

[17]　NoSQL（Not Only SQL）　野村総合研究所
https://www.nri.com/jp/knowledge/glossary/lst/alphabet/nosql

[18]　NoSQL とは？（NoSQL データベースの解説と SQL との比較）　Amazon Web Service
https://aws.amazon.com/jp/nosql/

[19]　リレーショナル・データベースとは　Oracle
https://www.oracle.com/jp/database/what-is-a-relational-database/

[20]　基本的なセキュリティ　ルール　Firebase　2021.9.1
https://firebase.google.com/docs/rules/basics?hl=ja