

Programming Boot Camp #3

第3回：ユーザー登録、ユーザー認 証、データベース連携(Firebase編)

東京工業大学 2020/11/14

Ryo Imahashi

目次

- 前回までのふりかえり
- 今回やることの確認
- Firebaseってなに？
- Firebaseを使う準備をしよう
- ユーザー登録をしよう
- ユーザー認証をしよう
- データベース連携をしよう

前回までのふりかえり

- 第1回では、HTML、CSS、Javascriptの基礎を学びました。
 - [参考: 第1回資料](#)
- 第2回では、Vue.js(Nuxt.js)の基礎を学びました。
 - [参考: 第2回資料](#)

開発環境を整えよう

- ・ 今回が初参加の方は、ご自身のPCで開発をするための環境構築をしましょう。ギルドワークスのメンバーがサポートしますので、声を掛けてください。
 - [参考: 環境構築の資料](#)
- ・ 環境構築ができている方は、[GitHub上のリモートリポジトリ](#)から最新のコードを落としてきてください。
 - **titech-2020** ディレクトリでGitBash(Terminal)を開いて、以下のコマンドを実行しましょう。

```
git pull
```

前回作ったアプリの確認

- ふりかえりのため、前回作ったアプリを動かして確認しましょう。
 - GitBash(Terminal)で以下のコマンドを実行しましょう。

```
# から始まる行はコメントなので、実行されません

# 現在位置がtitech-2020ディレクトリの場合、移動する
cd titech-nuxt-day2-answer

# 必要なモジュールをインストール
npm install

# アプリを起動
npm run dev
```

- その後、<http://localhost:3000/> にアクセスしましょう。

トップ画面

TOTAL SAMPLE PROJECT

LIST & DETAIL

メンバーリスト と プロフィール



Copyright GuildWorks Inc.

<http://localhost:3000/>

メンバー一覧

TOTAL SAMPLE PROJECT

メンバーリスト

氏名	Email	担当	
仁和 泰也	niwa@example.com	リーダー	 Profile
川面 崇義	kawadura@example.com	メンバー	 Profile
石北 俊寛	kitaishi@example.com	メンバー	 Profile
岸 美貴	kishi@example.com	メンバー	 Profile
大内田 結貴	ohdauchi@example.com	メンバー	 Profile

Copyright GuildWorks Inc.

<http://localhost:3000/list>

メンバー詳細

TOTAL SAMPLE PROJECT

メンバープロフィール



NAME

仁和 泰也

f t m

こんにちは。仁和 泰也といいます。友人からはにわりんと呼ばれているので、にわりんと呼んでください。住まいは新宿で、大学4回生です。趣味は、写真を撮ることが好きなので、休日は山に登り、風景や草花をカメラに収めています。また、自粛生活が続いたとき、料理くらいできないと思い、夏から料理教室に通っています。得意料理はカレーライスです。

生年月日	2000年10月10日
血液型	A型
出身地	東京都
所属・部署	東京工業大学 情報理工学院
星座	天秤座
趣味	写真・料理
ニックネーム	にわりん

Copyright GuildWorks Inc.

<http://localhost:3000/user/0001>

気になるところ

- 表示内容が変わりません。
 - 表示しているデータは実際に登録したものではなく、サンプルデータです。
 - ユーザーの追加ができません。
 - プロフィールの更新ができません。
- ログインをしなくとも、誰でもデータを見ることができてしまします。

今回やることの確認

開発する機能や画面は、以下の通りです。

- ユーザー登録
- ログイン
- ログイン状態による画面の表示制御
- ログアウト
- プロフィール編集
- あなたのプロフィール
- メンバーリスト(実際に登録したデータを表示)
- メンバープロフィール(実際に登録したデータを表示)

ユーザー登録

TOTAL SAMPLE PROJECT

ユーザー登録

imahashi@guildworks.jp

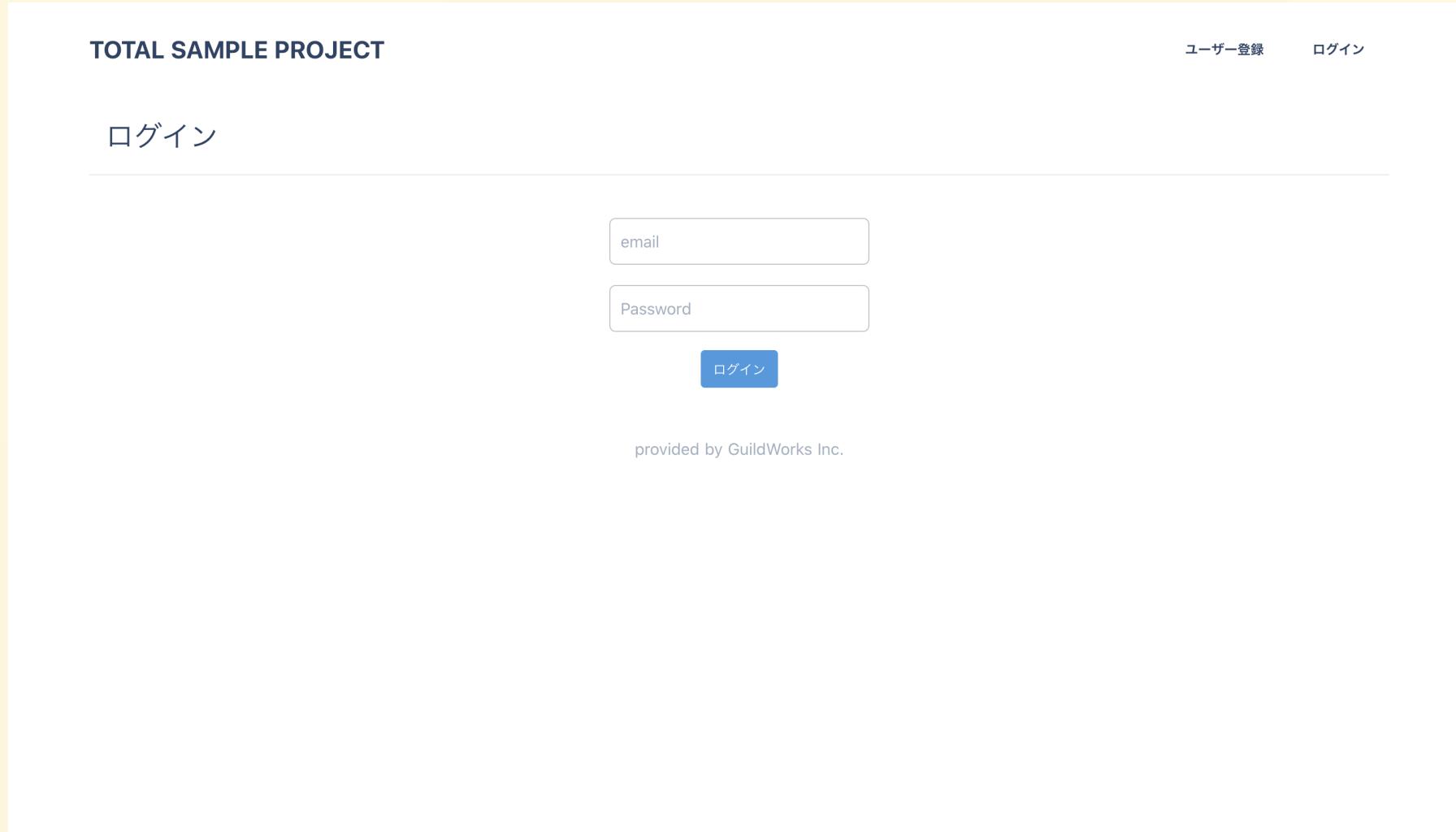
.....

登録する

provided by GuildWorks Inc.

The screenshot shows a user registration page. At the top left is the project name 'TOTAL SAMPLE PROJECT'. On the right are links for 'ユーザー登録' (User Registration) and 'ログイン' (Login). The main title 'ユーザー登録' is centered above the form. The form consists of two input fields: one containing the email 'imahashi@guildworks.jp' and another with several dots '.....'. Below the inputs is a blue button labeled '登録する' (Register). At the bottom of the page, it says 'provided by GuildWorks Inc.'

ログイン



The image shows a login form for a project titled "TOTAL SAMPLE PROJECT". The form is contained within a white rectangular box with a thin black border, set against a light yellow background. At the top left of the form, the text "TOTAL SAMPLE PROJECT" is displayed in a dark blue, sans-serif font. At the top right, there are two links: "ユーザー登録" (User Registration) and "ログイン" (Login), both in a smaller dark blue font. Below these links is the word "ログイン" in a larger, dark blue, bold font. The form itself consists of three input fields: a top field labeled "email", a middle field labeled "Password", and a bottom field labeled "ログイン" which is a solid blue button. Below the "ログイン" button, the text "provided by GuildWorks Inc." is visible in a small, gray font.

TOTAL SAMPLE PROJECT

ユーザー登録 ログイン

ログイン

email

Password

ログイン

provided by GuildWorks Inc.

ログイン状態による画面の表示制御、ログアウト

ログインしている時

TOTAL SAMPLE PROJECT

メンバーリスト

あなたのプロフィール

ログアウト

ログインしていない時

TOTAL SAMPLE PROJECT

ユーザー登録

ログイン

プロフィール編集

TOTAL SAMPLE PROJECT

メンバーリスト あなたのプロフィール ログアウト

プロフィール編集

登録

NAME
今橋 陵
✉ imahashi@guildworks.jp

役割
リーダー

今橋です。よろしくお願いします。

所属・部署 ギルドワークス

ニックネーム いまはし

出身地 山口県

生年月日 1992年7月24日

血液型 O型

星座 獅子座

趣味 猫と遊ぶ、料理

provided by GuildWorks Inc.

あなたのプロフィール

TOTAL SAMPLE PROJECT

あなたのプロフィール

メンバーリスト あなたのプロフィール ユーザー登録 ログアウト

NAME
今橋 陵
imahashi+3@guildworks.jp

あなたのプロフィール

編集

所属・部署 ギルドワークス

ニックネーム いまはし

出身地 山口

生年月日 1992年7月24日

血液型 O型

星座 獅子座

趣味 猫と遊ぶ・料理

こんにちは。ギルドワークスの今橋です。

今日はみんなで楽しくプログラミングをしましょう。

provided by GuildWorks Inc.

メンバーリスト(実際に登録したデータを表示)

TOTAL SAMPLE PROJECT

メンバーリスト あなたのプロフィール ユーザー登録 ログアウト

メンバーリスト

氏名	Email	担当	
今橋 陵	imahashi+3@guildworks.jp	メンバー	 Profile
上野 潤一郎	ueno@guildworks.jp	リーダー	 Profile
金 翔海	kim@guildworks.jp	メンバー	 Profile
京極 直丈	kyogoku@guildworks.jp	メンバー	 Profile

provided by GuildWorks Inc.

メンバープロフィール(実際に登録したデータを表示)

TOTAL SAMPLE PROJECT

メンバープロフィール

NAME
上野 潤一郎
✉ ueno@guildworks.jp

上野です。よろしくお願いします。

メンバーリスト あなたのプロフィール ユーザー登録 ログアウト

所属・部署	ギルドワークス
ニックネーム	うえじゅん
出身地	横浜
生年月日	1977年8月15日
血液型	内緒
星座	獅子座
趣味	Appleの新製品をチェックすること

provided by GuildWorks Inc.

- 今回修正していくアプリの現在の状態を確認しておきましょう。
 - 先ほど使ったGitBash(Terminal)で、[Ctrl]+[C]キーを押して、アプリを停止しておきましょう。
 - 以下のコマンドを実行しましょう。

```
pwd  # 現在位置の確認
cd .. # 一階層上がる
pwd  # もう一度、現在位置の確認
cd titech-nuxt-firebase-tutorial # 今回修正していくアプリのディレクトリに移動
npm install # 必要なモジュールをインストール
npm run dev # アプリを起動
```

- その後、<http://localhost:3000/> にアクセスしましょう。
 - 各画面や機能の雰形を追加してあるのが分かると思います。

- ・今回開発する機能に必要な、アプリケーションのバックエンド処理を使えるようにしましょう。
 - 第1回、第2回で作ったのはフロントエンド(アプリケーション)です。ブラウザ側で動作します。
 - バックエンド(アプリケーション)は、インターネットの先で動いて、必要なデータを返してくれます。サーバーサイド(アプリケーション)ともいいます。
 - フロントエンド-バックエンドという対比や、クライアントサイド、サーバーサイドという対比で語られることが多いです。
- ・このブートキャンプでは、バックエンドを1から開発することはせず、代わりにFirebaseというサービスを利用します。

Firebaseってなに？

- Firebase は Google が提供しているモバイルおよび Web アプリケーションのバックエンドサービスです。Firebase を使うことで、開発者はアプリケーションの開発に専念でき、バックエンドで動くサービスを作成する必要も管理する必要もありません。
- サーバーやデプロイが不要で、簡単にバックエンド処理を利用できるようになります。
- 支払い情報の登録不要で利用できる無料プランがありますので、それを使っていきましょう。

Firebaseを使う準備をしよう

- 以下のリンクにアクセスしましょう。
 - <https://console.firebaseio.google.com/?hl=ja&pli=1>
- Googleアカウントのログインを求められた場合、
 - 既にGoogleアカウントを持っている人は、ログインをしましょう。
 - Googleアカウントを持っていない人は、左下の「アカウントを作成」リンクをクリックして、アカウントを作成しましょう。

- こんな画面が表示されたらOKです。



- 「プロジェクトを作成」をクリックしましょう。

- プロジェクトに名前をつけて、「続行」をクリックしましょう。
 - 名前は何でもOKです。

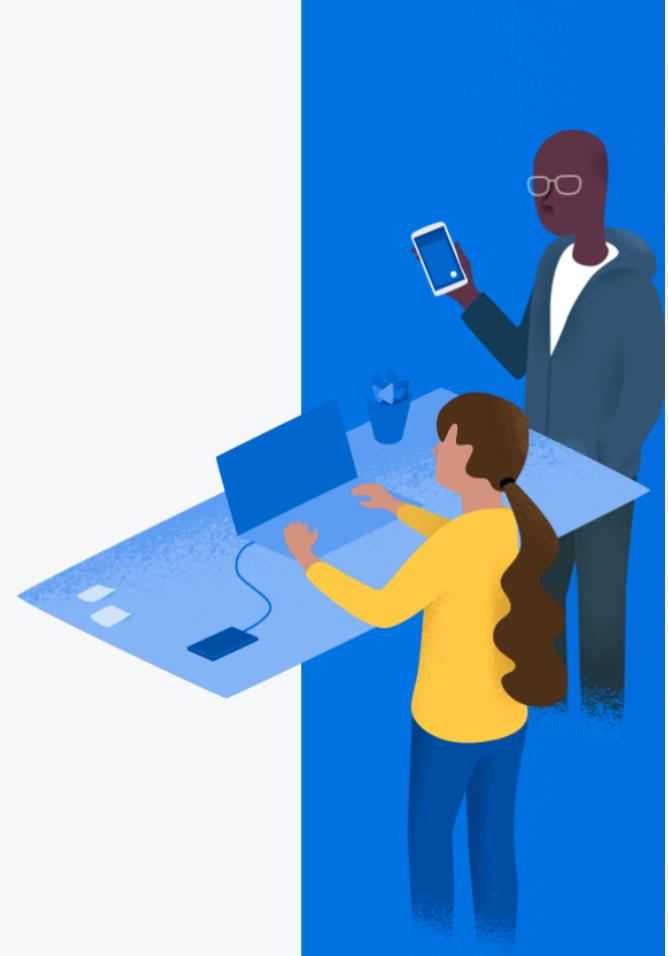
× プロジェクトの作成（手順 1/3）

まず
プロジェクトに名前をつける^⑨

プロジェクト名
titech-2020-imahashi

[titech-2020-imahashi](#) [guildworks.jp](#)

続行



- Googleアナリティクスを無効にして、「プロジェクトを作成」をクリックしましょう。

× プロジェクトの作成 (手順 2/2)

Google アナリティクス (Firebase プロジェクト向け)

Google アナリティクスは無料かつ無制限のアナリティクス ソリューションです。これにより、Firebase Crashlytics、Cloud Messaging、アプリ内メッセージング、Remote Config、A/B Testing、Predictions、Cloud Functions で、ターゲティングやレポートなどが可能になります。

Google アナリティクスによって以下の機能が有効になります。

- ×
- A/B テスト ⑦
- ×
- Firebase プロダクト全体でのユーザー ⑦
セグメンテーションヒターゲティング
- ×
- ユーザー行動の予測 ⑦
- ×
- クラッシュに遭遇していないユーザー ⑦
数
- ×
- イベントベースの Cloud Functions ト ⑦
リガー
- ×
- 無料で無制限のレポート ⑦

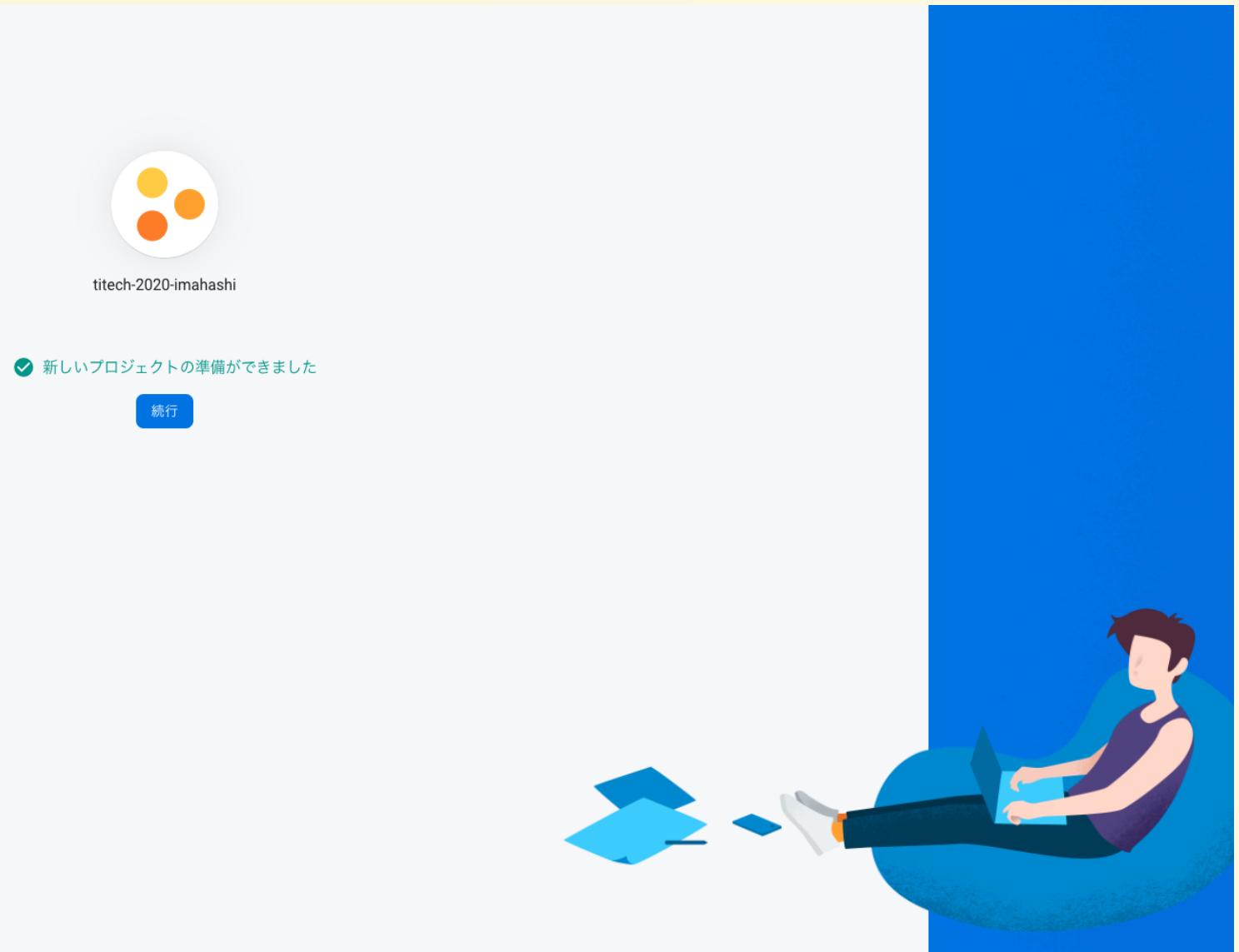
このプロジェクトで Google アナリティクスを有効にする
おすすめ

前へ

プロジェクトを作成



- プロジェクトが作成できたら、「続行」をクリックしましょう。



- これが作成されたプロジェクトの画面です。このプロジェクトを使って開発を進めていきます。

The screenshot shows the Firebase console interface for a project named "titech-2020-imahashi". The left sidebar contains navigation links for "プロジェクトの概要", "開発" (Authentication, Cloud Firestore, Realtime Database, Storage, Hosting, Functions, Machine Learning), "品質" (Crashlytics, Performance, Test Lab...), "分析" (Dashboard, Events, Conversions...), "拡大" (Predictions, A/B Testing, Cloud M...), and "Extensions". The main area features a large blue banner with the text "titech-2020-imahashi Spark プラン" and "アプリに Firebase を追加して利用を開始しましょう". It includes icons for iOS, Android, and web development, and a call to action "開始するにはアプリを追加してください". Below the banner, there are two cards: "Authentication" (ユーザーの認証と管理) and "Cloud Firestore" (リアルタイムの更新、強力なクエリ、自動スケーリング). At the bottom right, there is a link "すべての 開発 の機能を表示". A small modal window at the top right says "Firebase の機能、リサーチ、イベントに関する最新情報のメールを受け取る" with "登録" and "X" buttons.

今回開発するアプリでFirebaseの初期設定をしよう

- 画面左上の歯車アイコンをクリックして、「プロジェクトを設定」を選択しましょう。



- 画面下部マイアプリ欄の「</>」アイコンをクリックして、プロジェクトにWebアプリを作成しましょう。

The screenshot shows the GCP Project Settings page for the project "titech-2020-imahashi". The top navigation bar includes "ドキュメントに移動" and user profile icons. The main content area has tabs for "全般" (selected), "Cloud Messaging", "統合", "サービス アカウント", "データのプライバシー", and "ユーザーと権限".

プロジェクト

プロジェクト名	titech-2020-imahashi
プロジェクト ID	titech-2020-imahashi
プロジェクト番号	454271650
デフォルトの GCP リソース ロケーション	未選択
GCP 内の親組織 / フォルダ	guildworks.jp
ウェブ API キー	このプロジェクトにはウェブ API キーがありません

公開設定

これらの設定は公開されるプロジェクトのインスタンスを制御します

公開名	project-454271650
サポートメール	未設定

マイアプリ

プロジェクトにはアプリがありません
開始するにはプラットフォームを選択してください

Icons for iOS, Android, Web, and a blue circle with a white double-headed arrow (</>) are displayed.

- アプリのニックネームを入力して、「アプリを登録」をクリックしましょう。
 - ニックネームは何でもOKです。

× ウェブアプリに **Firebase** を追加

The screenshot shows the 'Add app' wizard in the Firebase console. Step 1 is titled 'アプリの登録' (App registration). It asks for the app's nickname, which is 'pbc-sample'. There is an optional checkbox to set up Firebase Hosting later. A blue button labeled 'アプリを登録' (Register app) is at the bottom. Step 2, 'Firebase SDK の追加' (Add Firebase SDK), is shown below it.

1 アプリの登録

アプリのニックネーム ②

pbc-sample

このアプリの **Firebase Hosting** も設定します。 詳細

Hosting は後で設定することもできます。いつでも無料で始めることができます。

アプリを登録

2 Firebase SDK の追加

- 選択中の箇所(apiKeyからappIdまでの行)をコピーしましょう。

2 Firebase SDK の追加

これらのスクリプトをコピーして <body> タグの下部に貼り付けます。この作業は Firebase サービスを使用する前に行ってください。

```
<!-- The core Firebase JS SDK is always required and must be listed first -->
<script src="https://www.gstatic.com/firebasejs/8.0.1.firebaseio.js"></script>

<!-- TODO: Add SDKs for Firebase products that you want to use
      https://firebase.google.com/docs/web/setup#available-libraries -->

<script>
  // Your web app's Firebase configuration
  var firebaseConfig = {
    apiKey: "AIzaSyAZk4BIcu84kIK72SEkHSnafuZTP-7TCS8",
    authDomain: "titech-2020-imahashi.firebaseio.com",
    databaseURL: "https://titech-2020-imahashi.firebaseio.com",
    projectId: "titech-2020-imahashi",
    storageBucket: "titech-2020-imahashi.appspot.com",
    messagingSenderId: "454271650",
    appId: "1:454271650:web:2252a5355e5f24d6052dfb"
  };
  // Initialize Firebase
  firebase.initializeApp(firebaseConfig);
</script>
```



ウェブ向け Firebase の詳細については、こちらをご覧ください: [使ってみる](#)

- Visual Studio Codeを開いて、コピーした設定を
`/titech-nuxt-firebase-tutorial/plugins/firebase.ts` に貼り付けましょう。

```
1 import firebase from 'firebase/app'
2 import 'firebase/auth'
3 import 'firebase/firestore'
4 import 'firebase/storage'

5
6 if (!firebase.apps.length) {
7   firebase.initializeApp({
8     apiKey: "AIzaSyAZk4BIcu84kIK72SEkHSnafuZTP-7TCS8", You,
9     authDomain: "titech-2020-imahashi.firebaseio.com",
10    databaseURL: "https://titech-2020-imahashi.firebaseio.com",
11    projectId: "titech-2020-imahashi",
12    storageBucket: "titech-2020-imahashi.appspot.com",
13    messagingSenderId: "454271650",
14    appId: "1:454271650:web:2252a5355e5f24d6052dfb"
15  })
16}
17
18 export default firebase
```

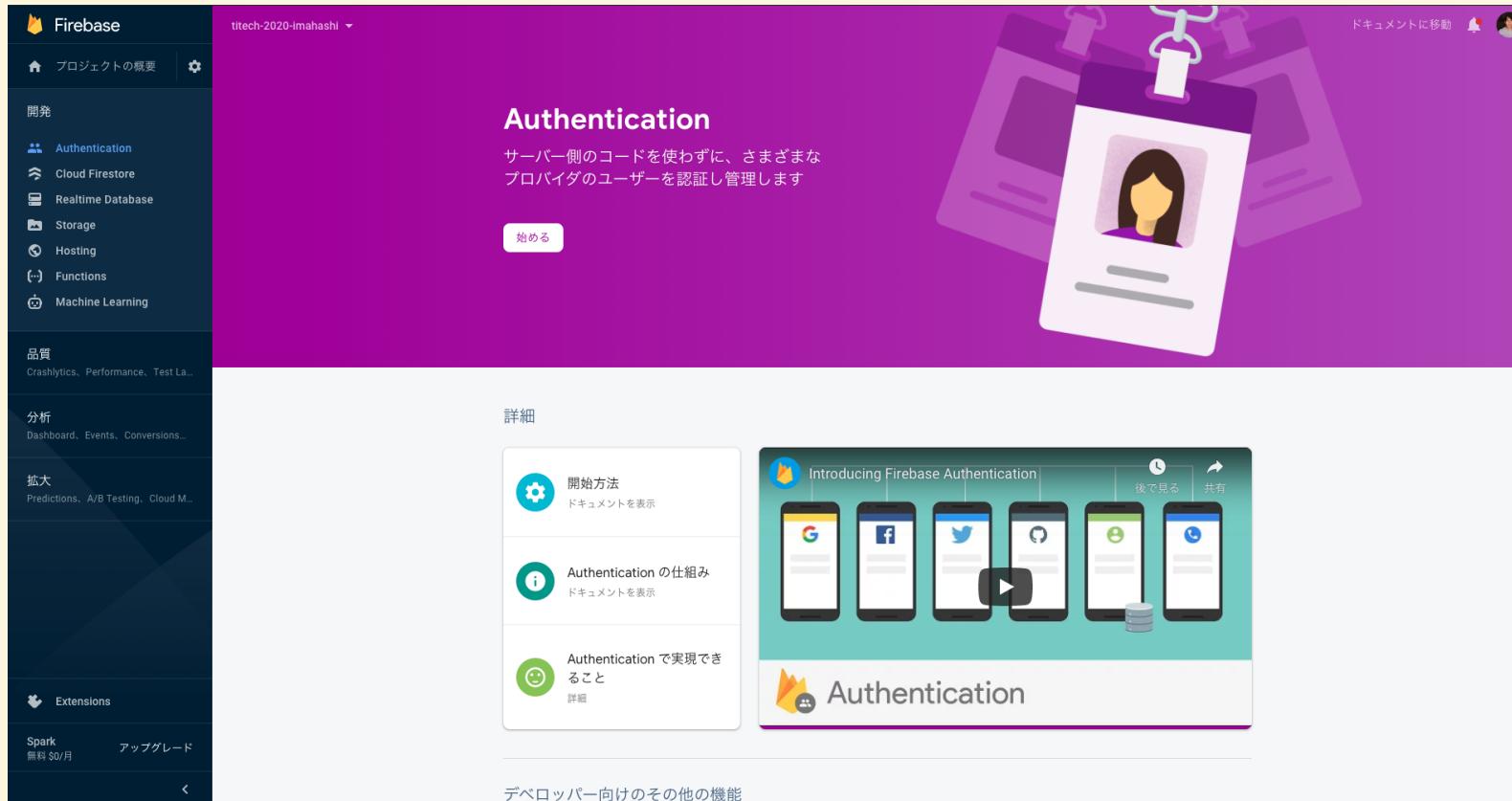
- これで、Firebaseの初期設定は完了です。

ユーザー登録機能を作ろう

- Firebase Authenticationを使います。
 - 認証機能を提供してくれるサービスです。
 - 様々な方法での認証をサポートしています。
 - メールアドレスとパスワード
 - Googleアカウント
 - Facebookアカウント
 - Twitterアカウント
 - など

Firebase側の設定

- 左側のメニューで「Authentication」を選んで、「始める」をクリックしましょう。



- 今回は、メールアドレスとパスワードによる認証を使うため、「メール/パスワード」をクリックしましょう。

titech-2020-imahashi ▾

ドキュメントに移動

?

Authentication

Users Sign-in method Templates Usage

ログイン プロバイダ

プロバイダ	ステータス
✉ メール / パスワード	無効
📞 電話番号	無効
.Google	無効
▶ Play ゲーム	無効
Game Center	無効
Facebook	無効
Twitter	無効
Github	無効
Yahoo!	無効
Microsoft	無効
Apple	無効
👤 匿名	無効

- 1つ目のスライダーをクリックして有効化し、「保存」をクリックしましょう。



- これで、メールアドレスとパスワードによる認証機能が使えるようになります。

プログラムの編集

- `/pages/signup.vue` を開きましょう。これが、ユーザー登録画面のファイルです。
- 「登録する」ボタンを押した時に実行される `submit()` 関数の中身が空になっています。

```
function submit() {  
  // TODO  
}
```

- この関数内で、Firebase Authenticationが提供している、メールアドレスとパスワードによるユーザー登録処理を呼び出すことで、ユーザー登録を行えるようにします。

- `/pages/signup.vue` 内でFirebaseを扱えるようにするために、`firebase`をインポートしましょう。

```
<script lang="ts">
import { defineComponent, reactive } from 'nuxt-composition-api'
import PageHeading from '@/components/page-heading.vue'
import firebase from '@/plugins/firebase.ts' // この行を追加
```

- 以下のリンクを開いてください。

[https://firebase.google.com/docs/auth/web/password-auth#create_a_password-based account](https://firebase.google.com/docs/auth/web/password-auth#create_a_password-based_account)

- メールアドレスとパスワードによるユーザー登録処理を呼び出すためのコードが記載されています。
- コードブロック右上の「Copy code sample」アイコンをクリックして、コピーしてください。

パスワード ベースのアカウントを作成する

パスワードを使用して新しいユーザー アカウントを作成するには、アプリのログインページで次の手順に沿って操作します。

- 新しいユーザーがアプリの登録フォームを使用して登録したら、アプリで必要な新しいアカウントの検証手順（新しいアカウントのパスワードが正しく入力されていることや、パスワードの複雑さの要件を満たしているかの確認など）を行います。
- 新しいユーザーのメールアドレスとパスワードを `createUserWithEmailAndPassword` に渡して、新しいアカウントを作成します。

```
firebase.auth().createUserWithEmailAndPassword(email, password).catch(function(error) {
  // Handle Errors here.
  var errorCode = error.code;
  var errorMessage = error.message;
  // ...
});
```

Copy code sample

- コピーしたコードを、 `submit()` 関数内に貼り付けてください。

```
const state = reactive({
  email: '',
  password: ''
})
function submit() {
  firebase.auth().createUserWithEmailAndPassword(email, password).catch(function(error) {
    // Handle Errors here.
    var errorCode = error.code;
    var errorMessage = error.message;
    // ...
  });
}
```

- 参考: コードを貼り付けてインデントが崩れた時は、複数の行を選択して[Tab]を押すと、選択した全ての行に対してインデントを追加できます。[Shift] + [Tab]でインデント削除もできます。

- そのままだと動かないで、`createUserWithEmailAndPassword()` の引数 `email` を `state.email` に、`password` を `state.password` に変更します。

```
const state = reactive({
  email: '',
  password: ''
})
function submit() {
  firebase.auth().createUserWithEmailAndPassword(state.email, state.password).catch(function(error) {
    // Handle Errors here.
    var errorCode = error.code;
    var errorMessage = error.message;
    // ...
  });
}
```

- 画面に入力した `email`、`password` が、`createUserWithEmailAndPassword()` に渡され、それらを含めたユーザー登録処理のリクエストが実行されるようになります。

- ユーザー登録完了後は、プロフィール編集画面に遷移するようにしておきましょう。

```
const state = reactive({
  email: '',
  password: ''
})
function submit() {
  firebase.auth().createUserWithEmailAndPassword(state.email, state.password)
  .then(() => (location.href = '/profile/edit')) // 改行してこの行を追加
  .catch(function(error) {
    // Handle Errors here.
    var errorCode = error.code;
    var errorMessage = error.message;
    // ...
  });
}
```

- 画面を操作して実際にユーザーを登録してみましょう。
 - メールアドレスとパスワードを入力し、登録するボタンを押してください。

TOTAL SAMPLE PROJECT

メンバーリスト あなたのプロフィール ユーザー登録 ログイン ログアウト

ユーザー登録

imahashi@guildworks.jp

.....

登録する

provided by GuildWorks Inc.



- Firebase Authenticationで、ユーザーが登録されたことが確認できます。

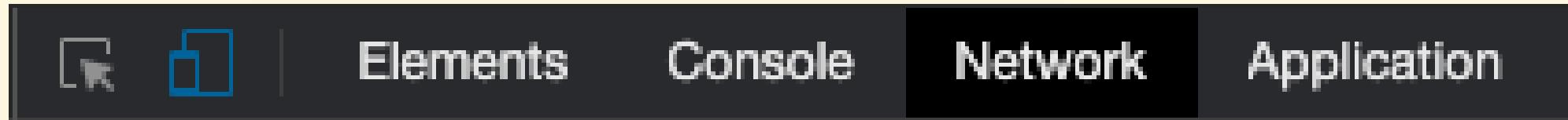


The screenshot shows the Firebase Authentication 'Users' page. At the top, there is a dropdown menu labeled 'titech-2020-imahashi'. Below it, the title 'Authentication' is displayed, followed by tabs: 'Users' (which is underlined in blue), 'Sign-in method', 'Templates', and 'Usage'. A search bar at the top right contains the placeholder text 'メールアドレス、電話番号、またはユーザー UID で検索'. To the right of the search bar are buttons for 'ユーザーを追加' (Add User) and settings. The main area displays a table with one row of data:

ID	プロバイダ	作成日	ログイン日	ユーザー UID ↑
imahashi@guildworks.jp	✉	2020/11/12	2020/11/12	nqjRTWnkcyQVrOT82XmlJUPQ4Lc2

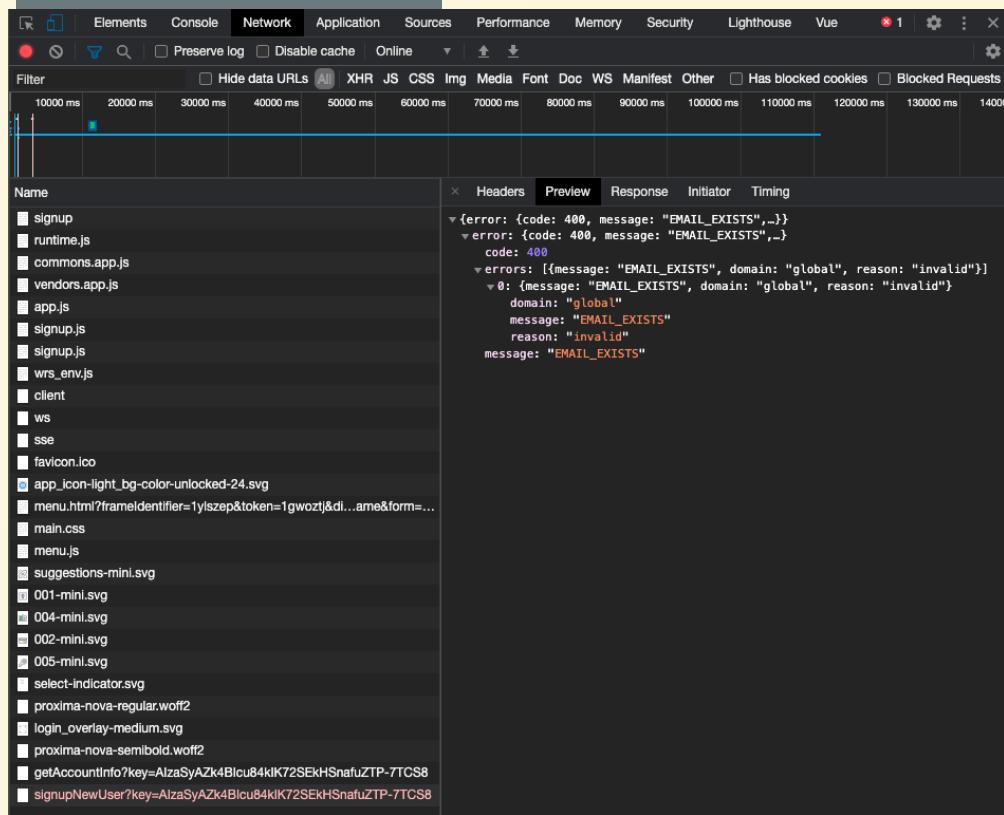
At the bottom of the table, there is a pagination message: 'ページあたりの行数: 50 ▾ 1 - 1 of 1 < >'.

- ・ ブラウザで開発者ツールを開きましょう(Chromeの場合、右クリックをして「検証」を選択)。
- ・ 上部のタブからNetworkタブを選択してください。



- ・ この状態で、先程と同じメールアドレスを使って再度ユーザー登録をしてみましょう。

- EMAIL_EXISTS というメッセージのエラーが起きています。



- 同じメールアドレスで複数のユーザーは登録できないように、
Firebase Authenticationのデフォルト設定で制限されているため
です。

- エラーが起きた場合、それが分かるようにしましょう。 `catch()` の中を以下のように修正してください。

```
.catch(function(error) {
  // Handle Errors here.
  alert('ユーザー登録が失敗しました。errorCode: ' + error.code + ', errorMessage:' + error.message)
});
```

- エラーが出ると、以下のようなポップアップが出るようになります。再度同じメールアドレスでユーザー登録をしてみてください。

localhost:3000 says

ユーザー登録が失敗しました。errorCode: auth/email-already-in-use,
errorMessage:The email address is already in use by another
account.

OK

ユーザー認証(ログイン)機能を作ろう

- ここでも、Firebase Authenticationを使います。
- `/pages/signin.vue` を開きましょう。これが、ログイン画面のファイルです。
- 「ログイン」ボタンを押した時に実行される `submit()` 関数の中身が空になっています。

```
function submit() {  
  // TODO  
}
```

- この関数内で、メールアドレスとパスワードによるログイン処理を呼び出すことで、ログインできるようにします。

- `/pages/signin.vue` 内でFirebaseを扱えるようにするために、`firebase`をインポートしましょう。

```
<script lang="ts">
import { defineComponent, reactive } from 'nuxt-composition-api'
import PageHeading from '@/components/page-heading.vue'
import firebase from '@/plugins/firebase.ts' // この行を追加
```

- 以下のリンクを開いてください。
https://firebase.google.com/docs/auth/web/password-auth#sign_in_a_user_with_an_email_address_and_password
 - メールアドレスとパスワードによるログイン処理を呼び出すためのコードが記載されています。
- コードブロック右上の「Copy code sample」アイコンをクリックして、コピーしてください。

メールアドレスとパスワードを使用してユーザーのログインを行う

パスワードを使用したユーザーのログイン手順は、新しいアカウントの作成手順と似ています。アプリのログインページで、次の手順に沿って操作します。

- ユーザーがアプリにログインしたら、そのユーザーのメールアドレスとパスワードを `signInWithEmailAndPassword` に渡します。

```
firebase.auth().signInWithEmailAndPassword(email, password).catch(function(error) {  
  // Handle Errors here.  
  var errorCode = error.code;  
  var errorMessage = error.message;  
  // ...  
});
```

ログインしたユーザーの詳細情報を取得する方法については、下記の「次のステップ」をご覧ください。

- コピーしたコードを、 `submit()` 関数内に貼り付けてください。

```
const state = reactive({
  email: '',
  password: ''
})
function submit() {
  firebase.auth().signInWithEmailAndPassword(email, password).catch(function(error) {
    // Handle Errors here.
    var errorCode = error.code;
    var errorMessage = error.message;
    // ...
  });
}
```

- コードを以下のように変更しましょう。

```
const state = reactive({
  email: '',
  password: ''
})
function submit() {
  firebase.auth().signInWithEmailAndPassword(state.email, state.password)
  .then(() => (location.href = '/users')) // ログイン完了後にユーザーリスト画面へ遷移させる
  .catch(function(error) {
    // Handle Errors here.
    alert('ログインが失敗しました。errorCode: ' + error.code + ', errorMessage:' + error.message)
  });
}
```

- `signInWithEmailAndPassword()` の引数 `email` を `state.email` に、
`password` を `state.password` に変更しています。
- ログイン成功時に `/users` (ユーザー一覧)へ遷移させています。
- ログイン失敗時のアラートを追加しています。

- 画面を操作して実際にログインしてみましょう。
 - メールアドレスとパスワードを入力し、ログインボタンを押してください。
 - 先程作成したユーザーのメールアドレスとパスワードに一致していれば、ユーザー一覧画面へ遷移します。
 - 先程作成したユーザーとメールアドレスが異なっていれば、`auth/user-not-found` エラーが起きます。
 - 先程作成したユーザーとメールアドレスが一致し、パスワードが異なっていれば、`auth/wrong-password` エラーが起きます。

ログイン状態による画面の表示制御

- ログイン状態に応じて、各画面を表示させるかどうかを制御していきましょう。
- ログイン状態の判定には、Nuxt.jsのmiddlewareという仕組みを使います。

<https://ja.nuxtjs.org/docs/2.x/directory-structure/middleware/>

- middlewareには、画面が描画される前に実行したい処理を記載します。

- `/middleware/Auth.js` を開きましょう。
 - まだ何の処理も書かれていません。

```
export default function () {  
}
```

- `/middleware/Auth.js` を以下のように変更しましょう。

```
import firebase from '@/plugins.firebaseio.ts'
export const skipAuthPaths = ['/signin', '/signup']

export default function ({ redirect, route, store }) {
  firebase.auth().onAuthStateChanged(function (user) {
    if (user) {
      // User is signed in.
      store.commit('signedIn', true);
    } else {
      // No user is signed in.
      store.commit('signedIn', false);
      if (!skipAuthPaths.includes(route.path)) {
        redirect('/signin')
      }
    }
  })
}
```

- `firebase.auth().onAuthStateChanged()` で現在ログインしているユーザーを取得し、ユーザーが取得できたかどうか(=ログインしているかどうか)で、その後の処理を分岐させています。
[https://firebase.google.com/docs/auth/web/manage-users#get the currently signed-in user](https://firebase.google.com/docs/auth/web/manage-users#get_the_currently_signed-in_user)
- `store.commit()` で、アプリケーションの状態を管理するための `store` という入れ物にログイン状態を保存します。
 - `store` はこの後で実装します。
- ログインしていない状態で認証が必要な画面を表示しようとした場合、`redirect('/signin')` でログイン画面に遷移させます。

- `/store/index.ts` を開いて、以下のように変更しましょう。

```
interface State {
  signedIn: boolean
}

export const state = () : State => ({
  signedIn: false
})

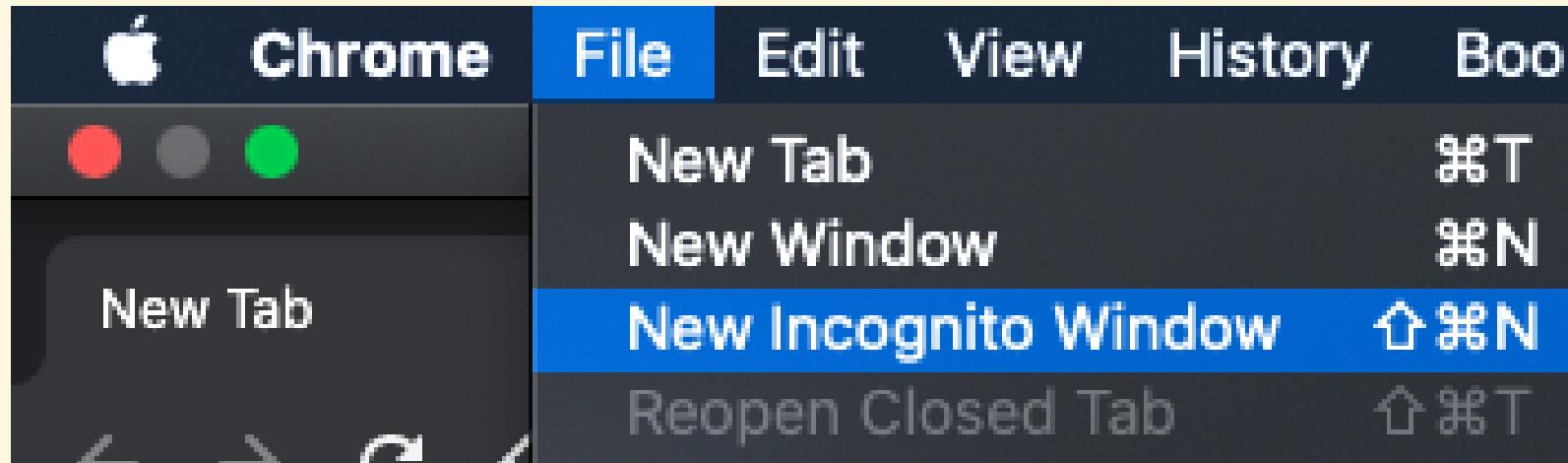
export const mutations = {
  signedIn(state: State, signedIn: boolean) {
    state.signedIn = signedIn
  }
}
```

- `/middleware/Auth.js` に記載した `store.commit('signedIn', true)` でログイン状態を保存できるようになります。

- 作成した `/middleware/Auth.js` がデフォルトレイアウトの描画前に実行されるように、`/layouts/default.vue` に1行追記しましょう。

```
<script lang='ts'>
import { defineComponent } from 'nuxt-composition-api'
export default defineComponent({
  middleware: [ 'Auth' ], // この行を追加し、Auth.jsを読み込む
  setup(_, { root: { $store } }) {
  }
})
</script>
```

- ・ブラウザをシークレットウィンドウで開きましょう。



- そのままでは先程ログインしたことをブラウザが記憶してしまっているため、これによってログイン前の状態で画面を開き直します。

- シークレットウィンドウで以下を確認しましょう。
 - <http://localhost:3000> (トップ)にアクセスすると、ログイン画面にリダイレクトされること。
 - メンバーリスト、あなたのプロフィールに遷移しようとしても、ログイン画面にリダイレクトされること。
 - ログインをすると、トップやメンバーリスト、あなたのプロフィールに遷移できるようになること。

メニューの表示制御

- `/layouts/default.vue` を開きましょう。
- Nuxt.jsの仕組みにより、このファイルに記載した内容がデフォルトのレイアウトとして各ページに反映されます。

<https://ja.nuxtjs.org/docs/2.x/concepts/views/#default-layout>

- ログイン状態に応じて、表示するメニューを変えましょう。
- `setup()` を以下のように修正しましょう。

```
setup(_, { root: { $store } }) {
  const isSignedIn = (): boolean => {
    return $store.state.signedIn
  }
  return {
    isSignedIn
  }
}
```

- ログイン状態を `isSignedIn` から判定できるようになります。
- 次のページのコードを参考に、`<template>` 内の `<a>` タグに `v-if` でログイン状態を判定する分岐を追加しましょう

```
<a  
  v-if="isSignedIn()"  
  href="/users"  
  class="text-blue-900 hover:text-blue-600 py-3 px-6 text-sm font-bold"  
>  
  メンバーリスト  
</a>  
<a  
  v-if="isSignedIn()"  
  href="/profile"  
  class="text-blue-900 hover:text-blue-600 py-3 px-6 text-sm font-bold"  
>  
  あなたのプロフィール  
</a>  
<a  
  href="/signup"  
  class="text-blue-900 hover:text-blue-600 py-3 px-6 text-sm font-bold"  
>  
  ユーザー登録  
</a>  
<a  
  v-if="!isSignedIn()"  
  href="/signin"  
  class="text-blue-900 hover:text-blue-600 py-3 px-6 text-sm font-bold"  
>  
  ログイン  
</a>  
<a  
  v-if="isSignedIn()"  
  href="#"  
  class="text-blue-900 hover:text-blue-600 py-3 px-6 text-sm font-bold"  
>  
  ログアウト  
</a>
```

ログアウト

- /layouts/default.vue の `<script>` 内を次のページのよう に変更しましょう。変更内容は以下の通りです。
 - firebaseをimportしています。
 - `setup()` 内に `signOut` を追加し、それをreturnして `template` から呼び出せるようにしています。
 - `firebase.auth().signOut()` では、ログアウト処理が実行されます。

https://firebase.google.com/docs/auth/web/password-auth#next_steps

```
import { defineComponent } from 'nuxt-composition-api'
import firebase from '@/plugins/firebase.ts'
export default defineComponent({
  middleware: ['Auth'],
  setup(_, { root: { $store } }) {
    const isSignedIn = (): boolean => {
      return $store.state.signedIn
    }
    const signOut = (): void => {
      firebase.auth().signOut()
    }
    return {
      isSignedIn,
      signOut
    }
  }
})
```

- ログアウトの `<a>` タグに `@click="signOut"` を追加しましょう。

```
<a  
  v-if="isSignedIn()"  
  href="#"  
  @click="signOut"  
  class="text-blue-900 hover:text-blue-600 py-3 px-6 text-sm font-bold">  
  ログアウト  
</a>
```

- ログアウトをクリックすると、ログアウト処理が実行されるようになります。

- メニューの表示が以下のように変わることを確認しましょう。
 - ログインしている時



- ログアウトした時



データベース連携をしよう

- Cloud Firestoreを使います。

<https://firebase.google.com/docs/firestore>

- データを保存するデータベースと、データを扱うための基本操作を提供してくれます。
- データは階層型の構造で格納されます。

● データの格納イメージ

titech-nuxt-sample	users	BOT6hdi032fe0pmzaTFrzyxEAVS2	
+ 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	BOT6hdi032fe0pmzaTFrzyxEAVS2	+ フィールドを追加	
	H8oRz6Gr13doFbPHM2Rec4oynwM2 1rIjMpbbLSN3XQXEjvx1du86gAr2 xEXCZWEwf6kK03p84zqQ17ppw1	comment: "こんにちは。ギルドワークスの今橋です。今日はみんなで楽しくプログラミングをしましょう。" email: "imahashi@guildworks.jp" iconUrl: "https://firebasestorage.googleapis.com/v0/b/titech-nuxt-sample.appspot.com/o/images%2Fprofile%2FBOT6hdi032fe0pmzaTFrzyxEAVS2?alt=media&token=338464db-b6a7-4fff-bbf9-fe69a4f07ef7" name: "今橋 陵" ▼ profile belongs: "ギルドワークス" birthday: "1992年7月24日" birthplace: "山口" bloodType: "O型" hobby: "猫と遊ぶ・料理" nickname: "いまはし" sign: "獅子座" role: "member"	

Firebase側の設定

- 左側のメニューで「Cloud Firestore」を選んで、「データベースの作成」をクリックしましょう。



- 「テストモードで開始する」を選んで「次へ」をクリック。

データベースの作成

1 Cloud Firestore のセキュリティ
保護ルール 2 Cloud Firestore のロケーション
を設定します

データ構造の定義後に、データのセキュリティを保護するルールを作成する必要があります。

[詳細](#)

本番環境モードで開始する
データはデフォルトで非公開になります。セキュリティルールで指定されているとおりに、クライアントの読み取り / 書き込み権限のみ付与されます。

テストモードで開始する
デフォルトでデータが開き、クリックセットアップが有効になります。セキュリティルールが更新されない場合、クライアントの読み取り / 書き込みアクセスは 30 日後に拒否されます。

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /{document=**} {
      allow read, write: if
        request.time < timestamp.date(2020, 12, 13);
    }
  }
}
```

! データベース参照を所有しているユーザーなら誰でも 30 日間、データベースのすべてのデータの表示、編集、削除を行えるようになります

Cloud Firestore を有効にすると、このプロジェクトで、関連する App Engine アプリから Cloud Datastore を使用できなくなります

キャンセル 次へ

- 「asia-northeast1」(東京)を選んで「有効にする」をクリック。



<https://firebase.google.com/docs/firestore/locations#location-r>

- データベースの準備ができました。

Cloud Firestore

データ ルール インデックス 使用状況

home icon

titech-2020-imahashi

+ コレクションを開始



A screenshot of the Cloud Firestore web interface. At the top, there's a navigation bar with tabs: 'データ' (selected), 'ルール', 'インデックス', and '使用状況'. Below the navigation is a header with a home icon and the project name 'titech-2020-imahashi'. A blue button '+ コレクションを開始' (Start Collection) is visible. The main area is mostly empty, featuring a large circular icon with server racks and a network connection, and the text 'データベースの準備ができました。データを追加してください。' (Database preparation is complete. Please add data.).

Cloud Firestore のロケーション: asia-northeast1

プロフィール編集(アイコン画像は除く)

- 自分のプロフィールを登録するため、プロフィール編集機能を作りましょう。
- <http://localhost:3000/profile/edit> を表示してください。
- `/pages/profile/edit.vue` を開きましょう。これが、プロフィール編集画面のファイルです。

- 最初に、firebaseをimportしておきましょう。

```
import firebase from '@/plugins.firebaseio.ts'
```

- 次は、プロフィール登録のために必要となる自分のユーザーIDを取得していきます。

- `// TODO: ユーザーID、メールアドレス取得` というコメントを削除して、そこに以下のコードを貼り付けましょう。

```
firebase.auth().onAuthStateChanged(function (user) {  
  if (user) {  
    // User is signed in.  
    userData.id = user.uid  
    userData.email = user.email  
    getUserData(user)  
  } else {  
    // No user is signed in.  
  }  
})
```

- `/middleware/Auth.js` で使ったのと同じ `onAuthStateChanged()` を使ってユーザーID、メールアドレスを取得し、必要な箇所にセットしています。

- `setProfile` を次のように変更してください。

```
const setProfile = (): void => {
  const data = {
    name: userData.name,
    email: userData.email,
    role: userData.role,
    iconUrl: userData.iconUrl,
    comment: userData.comment,
    profile: userData.profile,
  }
  firebase
    .firestore()
    .collection('users') // usersコレクションの、
    .doc(userData.id) // <ユーザーID>というドキュメントに、
    .set(data) // dataをセットする
    .then(() => {
      window.location.href = '/profile' // 完了後、プロフィール画面へ遷移
    })
}
```

- 実際にプロフィールを入力して、「登録」ボタンを押してみましょう。

TOTAL SAMPLE PROJECT

メンバーリスト あなたのプロフィール ログアウト

プロフィール編集

登録

NAME
今橋 陵
✉ imahashi@guildworks.jp

所属・部署 ギルドワークス

ニックネーム いまはし

出身地 山口県

生年月日 1992年7月24日

血液型 O型

星座 獅子座

趣味 猫と遊ぶ、料理

役割 リーダー

今橋です。よろしくお願いします。

provided by GuildWorks Inc.

- Cloud Firestoreにデータが登録されたことを確認しましょう。

The screenshot shows the Cloud Firestore interface with the following navigation path: Home > users > nqjRTWnkcyQVr... . The left sidebar shows a collection named "users". The main area displays a single document with the ID "nqjRTWnkcyQVrOT82Xm1JUPQ4Lc2". The document contains the following fields:

- comment: "今橋です。よろしくお願いします。"
- email: "imahashi@guildworks.jp"
- iconUrl: ""
- name: "今橋 陵"
- profile (expanded):
 - belongs: "ギルドワークス"
 - birthday: "1992年7月24日"
 - birthplace: "山口県"
 - bloodType: "O型"
 - hobby: "猫と遊ぶ、料理"
 - nickname: "いまはし"
 - sign: "獅子座"
 - role: "admin"

- 登録したデータをプロフィール編集画面に表示しましょう。
 - `getUserData` を次のように変更してください。

```
const getUserData = (user) => {
  firebase
    .firestore()
    .collection('users') // usersコレクションから、
    .doc(user.uid) // 指定したuidのドキュメントを
    .get() // 取得する
    .then((doc) => {
      if (doc.exists) {
        userData.name = doc.data().name
        userData.role = doc.data().role
        userData.iconUrl = doc.data().iconUrl
        userData.profile = doc.data().profile
        userData.comment = doc.data().comment
      }
    })
    .catch((err) => {
      console.log('Error getting user document', err);
    })
}
```

- プロフィール編集画面を表示すると、登録済みのデータが表示されるようになっています。

TOTAL SAMPLE PROJECT

メンバーリスト あなたのプロフィール ログアウト

プロフィール編集

登録

NAME
今橋 陵
✉ imahashi@guildworks.jp

役割
リーダー

今橋です。よろしくお願いします。

所属・部署 ギルドワークス

ニックネーム いまはし

出身地 山口県

生年月日 1992年7月24日

血液型 O型

星座 獅子座

趣味 猫と遊ぶ、料理

provided by GuildWorks Inc.

あなたのプロフィール画面へのデータ表示

- 同様に、あなたのプロフィール画面(<http://localhost:3000/profile>)でも登録済みのデータが表示されるようにしましょう。
- あなたのプロフィール画面のファイルは、
`/pages/profile/index.vue` です。
- プロフィール編集画面でやったことを参考に、ご自身でコードを修正してみてください。

- 答えはこちら。

<https://github.com/GuildWorks/titech-2020/blob/master/titech-nuxt-firebase-day3-answer/pages/profile/index.vue>

メンバーリスト(実際に登録したデータを表示)

- メンバーリスト画面(<http://localhost:3000/users>)でも登録済みのデータが表示されるようにしましょう。
- メンバーリスト画面のpageファイルは、`/pages/users/index.vue`ですが、今回はその中で読み込まれている
`/components/list-table.vue`を編集していきます。

- firebaseをimportしてください。

```
import firebase from '@/plugins.firebaseio.ts'
```

- setup() を次のように変更してください。

```
setup(_) {
  const userList = reactive<User[]>([])
  firebase
    .firestore()
    .collection('users') // usersコレクションから、
    .get() // 全てのドキュメントを取得する
    .then(function (querySnapshot) {
      querySnapshot.forEach(function (doc) { // 取得したドキュメントの配列のそれぞれの要素で、
        userList.push({ // 各フィールドの値を取り出して、userList配列にオブジェクトとして追加していく
          id: doc.id,
          name: doc.data().name,
          email: doc.data().email,
          role: doc.data().role,
          iconUrl: doc.data().iconUrl,
          comment: doc.data().comment,
          profile: doc.data().profile,
        })
      })
    })
  })
}

const userLink = (userId: string): void => {
  window.location.href = '/users/' + userId
}

return {
  userList,
  userLink,
}
},
```

- 登録したユーザーが表示されることを確認しましょう。

TOTAL SAMPLE PROJECT

メンバーリスト

氏名	Email	担当	
今橋 陵	imahashi@guildworks.jp	リーダー	 Profile

provided by GuildWorks Inc.

- ユーザーを追加して、複数ユーザーがいる場合の表示を確認しましょう。
 - ログアウトをして、別のメールアドレスでユーザー登録をしてください。
 - メールアドレスの存在チェックはしていませんので、適当なメールアドレスでもユーザー登録は可能です。
 - しかし、今後アプリを修正してメール送信機能をつけた場合にメールが送信されてしまうので、可能であれば自分のメールアドレスを使うことが望ましいです。

参考

- Gmailのエイリアス機能を使うと、1つのメールアドレスで複数のメールアドレスが使って便利です。
 - `imahashi@gmail.com` のアカウントを持っている場合、`imahashi+2@gmail.com`、`imahashi+third@gmail.com` といったように、`@` の前に`+`と好きな英数字を追加すれば、同じメールアドレスにメールが届きます。

- 追加したユーザーも表示されることが確認できました。

TOTAL SAMPLE PROJECT

メンバーリスト

メンバーリスト

氏名	Email	担当	
今橋 陵	imahashi@guildworks.jp	リーダー	 Profile
今橋 +2	imahashi+2@guildworks.jp	メンバー	 Profile

provided by GuildWorks Inc.

メンバープロフィール(実際に登録したデータを表示)

- メンバープロフィール画面でも登録済みのデータが表示されるようにしましょう。
 - パス(http://localhost:3000/users/_id)の `_id` 部分にユーザーIDが必要なので、ユーザーリスト画面から遷移して表示してください。
- メンバープロフィール画面のファイルは、 `/pages/users/_id.vue` です。

- firebaseをimportしてください。

```
import firebase from '@/plugins.firebaseio.ts'
```

- setup() を次のように変更してください。

```
setup(_, { root }: SetupContext) {
  const userData = reactive<User>({
    id: '',
    name: '',
    email: '',
    role: '',
    iconUrl: '',
    comment: '',
    profile: {
      belongs: '',
      nickname: '',
      birthplace: '',
      birthday: '',
      bloodType: '',
      sign: '',
      hobby: ''
    }
  })
  firebase
    .firestore()
    .collection('users')
    .doc(root.$route.params.id) // URLからIDを取得して、そのIDのドキュメントを取得する
    .get()
    .then((doc) => {
      if (doc.exists) {
        userData.id = root.$route.params.id
        userData.name = doc.data().name
        userData.email = doc.data().email
        userData.role = doc.data().role
        userData.iconUrl = doc.data().iconUrl
        userData.profile = doc.data().profile
        userData.comment = doc.data().comment
      }
    })
    .catch((err) => {
      console.log('Error getting document', err)
    })
  return {
    userData,
  }
},
```

- メンバープロフィールが表示されることが確認できます。

TOTAL SAMPLE PROJECT

メンバープロフィール

NAME
今橋 +2
✉ imahashi+2@guildworks.jp

今橋が2人目に作ったユーザーです。

メンバーリスト あなたのプロフィール ログアウト

所属・部署	ギルドワークス
ニックネーム	2号
出身地	大岡山
生年月日	2020年11月14日
血液型	O型
星座	蠍座
趣味	プログラミング

provided by GuildWorks Inc.

おまけ

- プロフィール編集画面で写真をアップロードできるようにします。
- Cloud Storage for Firebaseを使います。
<https://firebase.google.com/docs/storage>
 - 写真や動画などのファイルを保存してくれるサービスです。
- `/pages/profile/edit.vue`を開いてください。

- 既に、ドラッグアンドドロップで画像をアプリに取得させることはできるようになっています。
- `setIcon` を次のように変更しましょう。

```
const setIcon = (file: File): void => {
  const storageRef = firebase.storage().ref()
  // プロフィール画像アップロード先への参照を取得
  const fileRef = storageRef.child(
    'images/profile/' + userData.id + '/' + file.name
  )
  // プロフィール画像をストレージにアップロード
  fileRef.put(file).then(function (snapshot) {
    // ユーザーデータのURLを更新する
    snapshot.ref.getDownloadURL().then((url) => {
      userData.iconUrl = url
    })
  })
}
```

- 画像をドラッグアンドドロップした後に「登録」ボタンを押すと、アイコンが設定されます。

TOTAL SAMPLE PROJECT

あなたのプロフィール

編集

NAME
今橋 陵
imahashi@guildworks.jp

今橋です。よろしくお願いします。

所属・部署	ギルドワークス
ニックネーム	いまはし
出身地	山口県
生年月日	1992年7月24日
血液型	O型
星座	獅子座
趣味	猫と遊ぶ、料理

provided by GuildWorks Inc.

- Cloud Storageを開くと、画像が保存されていることが確認できます。

The screenshot shows the Firebase Storage interface for the project 'titech-2020-imahashi'. The left sidebar includes links for Authentication, Cloud Firestore, Realtime Database, Storage (which is selected), Hosting, Functions, and Machine Learning. The main area is titled 'Storage' and shows a table of files under the 'Files' tab. The table has columns for Name, Size, Type, and Last Updated. One file, 'cats.jpg', is listed with a size of 57.78 KB, type of image/jpeg, and last updated on 2020/11/14. To the right of the table, there is a preview of the 'cats.jpg' image, which is a black and white photo of two cats. Below the preview, detailed information about the file is provided: Name (cats.jpg), Size (59,163 バイト), Type (image/jpeg), Creation Time (2020/11/14 0:00:02), and Last Modified (2020/11/14 0:00:02).

	名前	サイズ	種類	最終更新日
	cats.jpg	57.78 KB	image/jpeg	2020/11/14

cats.jpg

名前
cats.jpg

サイズ
59,163 バイト

タイプ
image/jpeg

作成日時
2020/11/14 0:00:02

更新日時
2020/11/14 0:00:02

まとめ

- ✓ 今回は、Firebaseを使ってユーザー登録、ログイン、データの登録・読み取り・更新などができるようになりました。
- ✓ アプリの完成形は、`titech-nuxt-firebase-day3-answer` ディレクトリに入っています。分からなかったところがある場合は、こちらを確認してください。
- ✓ 第1回から第3回までの内容を理解すれば、今回のようなシンプルなアプリの開発はできるようになります。
- ✓ 興味がある人は、今回作ったアプリを自由にカスタマイズしたり、自分で新しくアプリを作ったりしてみましょう！

以上です！

お疲れさまでした！