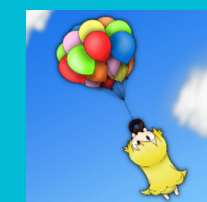


# 今日から始めるFirestoreのテスト



おりばー (@oliver\_diary)

# 自己紹介

# 自己紹介



# Daiki Minakawa ( @oliver\_diary )

- 新卒1年目です
- 個人ではRails使ったり  
職場ではElm触ったり
- サーバーサイドメインだったはずが最近ではフロントを触ることが多い...
- 趣味はカメラと音ゲーとアニメ

# どんなこと話す？

- Firestoreについてざっと
- Firestoreを使った一例
- Firestoreのテストを書いてみよう！
- CircleCIでテストを回そう！
- おわりに

**Firestore**についてざっと

# Firestoreについてざっと

- 公式には「Realtime Database よりも多彩で高速なクエリと高性能なスケーリングが特徴」と書かれている
- 要するに「すげー使いやすく」「すげー高速」で「すげースケーリングする」NoSQLデータベース
- データモデルとしては「データをドキュメントのコレクションとして保存」している

# Firestoreを使った一例

# Firestoreを使った一例

- みなさんはFirestoreを使ったことがありますか？
- 使ったことがあるのであれば、テストを書いたことがありますか？



# Firestoreを使った一例

- 一例
  - Userコレクションが存在し、アカウント登録時にドキュメントにEmailとNameが、それぞれ最小4文字、最大256文字で保存される
    - その仕様でAさんがまずRuleを書いた

# Firestoreを使った一例

- Rules of Aさん
  - Aさんが記述したRule  
(細かい部分は今回気にしない)

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /users/{userID} {
      allow read: if true
      ;

      allow update: if false
      ;

      allow create:
        if validateString(incomingData().name)
        && validateString(incomingData().email)
        ;
    }

    function validateString(text) {
      return text is string
      && 4 <= text.size()
      && text.size() <= 256
      ;
    }

    function incomingData() {
      return request.resource.data
      ;
    }
  }
}
```

# Firestoreを使った一例

- 一例
  - 後からNameを最小4文字、最大16文字にしたいという要望がきた
    - Bさんが引き継いでRuleを更新した

# Firestoreを使った一例

- Rules of Bさん
  - Bさんが変更したRule  
(validateStringの中身を変更)


```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /users/{userID} {
      allow read: if true
      ;

      allow update: if false
      ;

      allow create:
        if validateString(incomingData().name)
        && validateString(incomingData().email)
        ;
    }

    function validateString(text) {
      return text is string
      && 4 <= text.size()
      && text.size() <= 16
      ;
    }

    function incomingData() {
      return request.resource.data
      ;
    }
  }
}
```



# Firestoreを使った一例

- もちろんこれだと、Emailの文字数上限まで16文字になってしまい、この状態でリリースされたら大変なことに
- 元のRuleを書いたAさんが悪いのか？
- それともよくチェックをしてなかったBさんが悪いのか？

# Firestoreを使った一例

- コードの良し悪しはともかくとして、  
テストを書いておけば、最悪な状態（バグが存在した状態）  
でのリリースは避けられたはず

**Firestoreのテストを書いてみよう！**

# Firestoreのテストを書いてみよう！

- Firestoreのテストはローカルにエミュレータを立ててテストできる！
- なので、別途Firestoreのプロジェクトを作る必要はなし！
- みんな別々のエミュレータを利用するので、テストが壊れることもなし！
- CIでテストが簡単にできる！（後々の章で）



# Firestoreのテストを書いてみよう！

- 前準備として、firebase-toolsとemulatorのインストール・起動をさせておく
  - `npm i firebase-tools --save`
  - `node_modules/.bin/firebase setup:emulators:firestore`
  - `node_modules/.bin/firebase serve --only firestore`
    - エミュレータの起動

# Firestoreのテストを書いてみよう！

- `npm i mocha --save`
- test用のライブラリを追加
- `npm i @firebase/testing --save`
- Firebaseを使ったテストを書くときに必要

```
{
  "scripts": {
    "test": "mocha --timeout=10000"
  },
  "dependencies": {
    "@firebase/testing": "^0.12.0",
    "firebase": "^6.4.0",
    "firebase-tools": "^6.12.0",
    "mocha": "^6.2.0"
  }
}
```

# Firestoreのテストを書いてみよう！

- Aさんのfirestore.rules ファイルを作成

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /users/{userID} {
      allow read: if true
      ;

      allow update: if false
      ;

      allow create:
        if validateString(incomingData().name)
        && validateString(incomingData().email)
        ;
    }

    function validateString(text) {
      return text is string
      && 4 <= text.size()
      && text.size() <= 256
      ;
    }

    function incomingData() {
      return request.resource.data
      ;
    }
  }
}
```

# Firestoreのテストを書いてみよう！

- テストを test/test.js に記述していく

```
const firebase = require('@firebase/testing');
const fs = require('fs');

const projectId = 'firestore-emulator-example';
const rules = fs.readFileSync('firestore.rules', 'utf8');

function authedApp(auth) {
  return firebase.initializeTestApp({ projectId, auth }).firestore();
}

// エミュレータに残ってるデータお掃除と、Ruleの読み込みを行なっている
beforeEach(async () => { await firebase.clearFirestoreData({ projectId }); });
before(async () => { await firebase.loadFirestoreRules({ projectId, rules }); });
after(async () => { await Promise.all(firebase.apps().map(app => app.delete())); });

describe('Firestore tests', async () => {
  ...
});
```

# Firestoreのテストを書いてみよう！

- Aさんの実装時点でのテストケース

```
describe('Firestore tests', async () => {
  describe('正常系', async () => {
    // 本来であれば境界値でテストしたい
    it("Email 4~256文字, Name 4~256文字", async () => {
      const db = authedApp(null);
      const user = db.collection("users").doc("sampleUser");
      await firebase.assertSucceeds(
        user.set({ name: "Oliver", email: 'oliver@example.com' })
      );
    });
  });
  ...
});
```



# Firestoreのテストを書いてみよう！

- Aさんの実装時点でのテストケース

```
describe('Firestore tests', async () => {
  describe('正常系', async () => { ... });

  describe('非正常系', async () => {
    it("Email4~256文字, Name 3文字", async () => {
      const db = authedApp(null);
      const user = db.collection("users").doc("sampleUser");
      await firebase.assertFails(
        user.set({ name: "Oli", email: 'oliver@example.com' })
      );
    });
    it("Email3文字, Name 4~256文字", async () => { ... });
    it("Email257文字, Name 4~256文字", async () => { ... });
    it("Email4~256文字, Name 257文字", async () => { ... });
    ...
  });
});
```

# Firestoreのテストを書いてみよう！

- テスト実行！

```
└─$ npm run test

> @ test /Users/oliver/workspace/firestore_sample
> mocha --timeout=10000

Firestore tests
  正常系
    ✓ Email4~256文字, Name 4~256文字 (136ms)
  非正常系
    ✓ Email4~256文字, Name 3文字 (67ms)
    ✓ Email3文字, Name 4~256文字
    ✓ Email257文字, Name 4~256文字
    ✓ Email257文字, Name 4~256文字

5 passing (1s)
```

# Firestoreのテストを書いてみよう！

- Bさんのfirestore.rules ファイルを作成

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /users/{userID} {
      allow read: if true
      ;

      allow update: if false
      ;

      allow create:
        if validateString(incomingData().name)
        && validateString(incomingData().email)
        ;
    }

    function validateString(text) {
      return text is string
      && 4 <= text.size()
      && text.size() <= 16
      ;
    }

    function incomingData() {
      return request.resource.data
      ;
    }
  }
}
```



# Firestoreのテストを書いてみよう！

- Bさんの実装時点でのテストケース

```
describe('Firestore tests', async () => {
  describe('正常系', async () => {
    // 本来であれば境界値でテストしたい
    it("Email4~256文字, Name 4~16文字", async () => {
      const db = authedApp(null);
      const user = db.collection("users").doc("sampleUser");
      await firebase.assertSucceeds(
        user.set({ name: "Oliver", email: 'oliver@example.com' })
      );
    });
  });
  ...
});
```

# Firestoreのテストを書いてみよう！

- テスト実行！

```
└─$ npm run test

> @ test /Users/oliver/workspace/firestore_sample
> mocha --timeout=10000


Firestore tests
  正常系
    1) Email4~256文字, Name 4~16文字
  非正常系
    ✓ Email4~256文字, Name 3文字 (68ms)
    ✓ Email3文字, Name 4~16文字
    ✓ Email257文字, Name 4~256文字
    ✓ Email4~256文字, Name 17文字

4 passing (653ms)
1 failing

1) Firestore tests
   正常系
     Email4~256文字, Name 4~16文字:
     FirebaseError: 7 PERMISSION_DENIED:
false for 'create' @ L12
```

# Firestoreのテストを書いてみよう！

- テストが落ちたので、正しいfirestore.rulesを記述していく
- もっといいRuleの書き方や、Emailの細かいバリデーションとかは一旦無視で🙏

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /users/{userID} {
      allow read: if true
      ;

      allow update: if false
      ;

      allow create:
        if validateName(incomingData().name)
        && validateEmail(incomingData().email)
      ;
    }
  }
}

function validateName(text) {
  return text is string
  && 4 <= text.size()
  && text.size() <= 16
  ;
}

function validateEmail(text) {
  return text is string
  && 4 <= text.size()
  && text.size() <= 256
  ;
}

function incomingData() {
  return request.resource.data
  ;
}
}
```

# Firestoreのテストを書いてみよう！

- テスト実行！

```
└─$ npm run test  
  
> @ test /Users/oliver/workspace/firestore_sample  
> mocha --timeout=10000
```

Firestore tests

正常系

✓ Email4~256文字, Name 4~16文字 (104ms)

非正常系

✓ Email4~256文字, Name 3文字 (62ms)

✓ Email3文字, Name 4~16文字

✓ Email257文字, Name 4~256文字

✓ Email4~256文字, Name 17文字

5 passing (652ms)

**CircleCIでテストを回そう！**

# CircleCIでテストを回そう！

- ローカルでエミュレータを立ててテストをしてきたってことは、CIでも使えるはず...!
- ということで、CircleCIで動かしてみよう！（あと何分...？

# CircleCIでテストを回そう！

- ローカルでエミュレータを立ててテストをしてきたってことは、CIでも使えるはず...!
- ということで、CircleCIで動かしてみよう！（あと何分...?
  - 簡単にできるので、コードだけざっと説明します



# CircleCIでテストを回そう！

- キャッシュ戦略とか何もしてないけど許して🙏
- 特に難しいことはしてなく、エミュレータにjavaが必要なので、  
imageに  
circleci/openjdk:stretch-node-browsers-legacy  
を指定してるくらいです

```
version: 2
jobs:
  build:
    working_directory: ~/workspace
    docker:
      - image: circleci/openjdk:stretch-node-browsers-legacy
    steps:
      - checkout
      - run:
          name: Update npm
          command: 'sudo npm install -g npm@latest'
      - run:
          name: Install Dependencies
          command: 'npm install'
      - run: 'node_modules/.bin/firebase setup:emulators:firestore'
      - run:
          command: 'node_modules/.bin/firebase serve --only firestore'
          background: true
      - run: 'sleep 10'
      - run:
          name: Firestore Test
          command: 'npm test'
```



おわりに

# おわりに

- Firestoreは簡単に実装でき、簡単に利用できる反面、しっかりとRuleを書かなければ、データが壊れてしまうことを認識しないといけない
- Firestoreの機能もっとたくさん紹介したかったけど5分じゃ叶わず😭  
(FieldValueとかFirebase AuthのCustomClaimとの連携とか...)
- Qiitaとかに後日投稿します

ありがとうございました