Firebase's Authentication API

In this lesson, we'll figure out a way to establish communication between the Firebase API and our Firebase class.

WE'LL COVER THE FOLLOWING

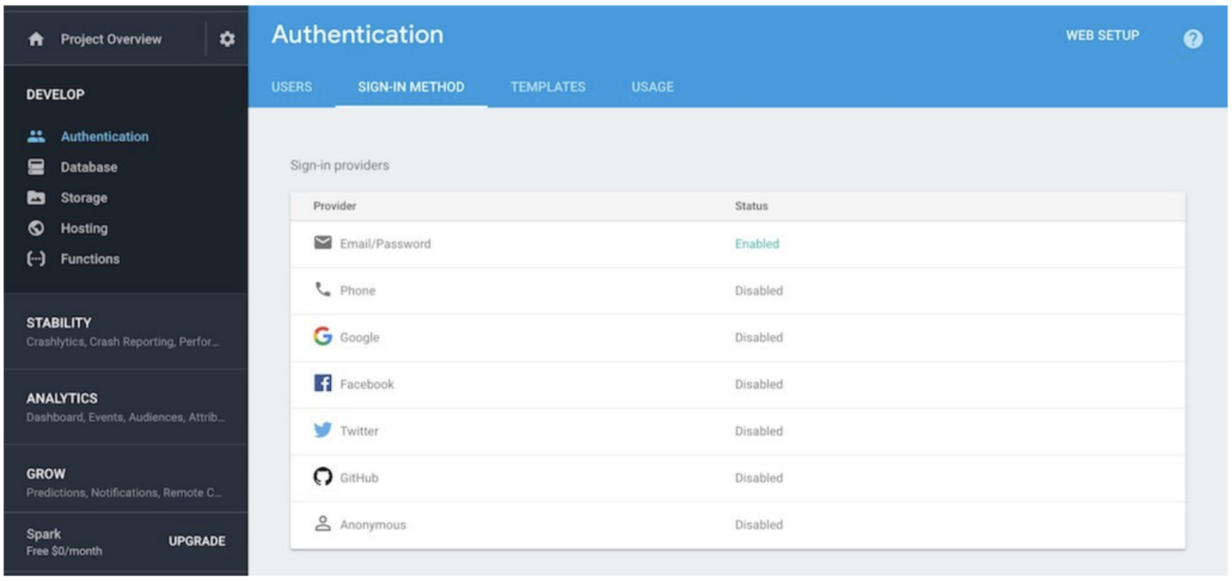
- Enabling Authentication
- Setting Up the Authentication API
  - Sign-Up
  - Sign-In
  - Error Handling
  - Reset/Change Password

In the previous lessons, we created a Firebase project on the official Firebase website. Now, we will implement the interface of our Firebase class which will enable communication between the class and the Firebase authentication API. In the upcoming lessons, we will be using the interface of the Firebase class in our React components.

Enabling Authentication

First, we need to activate one of the available authentication providers on Firebase's website. To do this, follow these instructions:

- On the project's Firebase dashboard, click on the **Authentication** option in the **DEVELOP** section.
- Select the **Sign-In Method** menu item.
- Set the authentication with **Email/Password** to *Enabled*.



Provider	Status
Email/Password	Enabled
Phone	Disabled
Google	Disabled
Facebook	Disabled
Twitter	Disabled
GitHub	Disabled
Anonymous	Disabled

Setting Up the Authentication API

Second, we will implement the **authentication API** for our Firebase class. To do that, we first need to import and instantiate the package from Firebase which is responsible for all the authentication.

Make the following changes in your `src/components/Firebase/` **`firebase.js`** file:

```
import app from 'firebase/app';
import 'firebase/auth';

const config = {
  apiKey: process.env.REACT_APP_API_KEY,
  authDomain: process.env.REACT_APP_AUTH_DOMAIN,
  databaseURL: process.env.REACT_APP_DATABASE_URL,
  projectId: process.env.REACT_APP_PROJECT_ID,
  storageBucket: process.env.REACT_APP_STORAGE_BUCKET,
  messagingSenderId: process.env.REACT_APP_MESSAGING_SENDER_ID,
};

class Firebase {
  constructor() {
    app.initializeApp(config);

    this.auth = app.auth();
  }
}

export default Firebase;
```

Firebase/firebase.js

Let’s define all the authentication functions as class methods, step-by-step. They will serve our communication channel from the Firebase class to the Firebase API.

Sign-Up

First, the **sign-up** function (registration) takes email and password parameters for its function signature and uses an official Firebase API endpoint to create a user:

```
import app from 'firebase/app';
import 'firebase/auth';

const config = { ... };

class Firebase {
  constructor() {
    app.initializeApp(config);

    this.auth = app.auth();
  }

  // *** Auth API ***

  doCreateUserWithEmailAndPassword = (email, password) =>
    this.auth.createUserWithEmailAndPassword(email, password);
}

export default Firebase;
```

Firebase/firebase.js

Sign-In

We’ll also set up the **login/sign-in** function, which takes the email and password parameters:

```
import app from 'firebase/app';
import 'firebase/auth';

const config = { ... };

class Firebase {
  constructor() {
    app.initializeApp(config);

    this.auth = app.auth();
  }
  // *** Auth API ***

  doCreateUserWithEmailAndPassword = (email, password) =>
    this.auth.createUserWithEmailAndPassword(email, password);

  doSignInWithEmailAndPassword = (email, password) =>
    this.auth.signInWithEmailAndPassword(email, password);
}

export default Firebase;
```

Firebase/firebase.js

Error Handling

Since these endpoints are called asynchronously, they will need to be resolved later. Error handling must also be taken care of.

For instance, it is not possible to sign in a user who is not signed up yet because the Firebase API would return an error. In case of the **sign out** function, we don't need to pass any arguments to it, because Firebase knows about the currently authenticated user.

If no user is authenticated, nothing will happen when this function is called.

```
import app from 'firebase/app';
import 'firebase/auth';

const config = { ... };

class Firebase {
  constructor() {
    app.initializeApp(config);

    this.auth = app.auth();
  }

  // *** Auth API ***

  doCreateUserWithEmailAndPassword = (email, password) =>
    this.auth.createUserWithEmailAndPassword(email, password);

  doSignInWithEmailAndPassword = (email, password) =>
    this.auth.signInWithEmailAndPassword(email, password);

  doSignOut = () => this.auth.signOut();
}

export default Firebase;
```

Firebase/firebase.js

Reset/Change Password

Reset/Change Password

There are two more authentication methods to reset and change a password for an authenticated user:

```
import app from 'firebase/app';
import 'firebase/auth';

const config = { ... };

class Firebase {
  constructor() {
    app.initializeApp(config);

    this.auth = app.auth();
  }

  // *** Auth API ***

  doCreateUserWithEmailAndPassword = (email, password) =>
    this.auth.createUserWithEmailAndPassword(email, password);

  doSignInWithEmailAndPassword = (email, password) =>
    this.auth.signInWithEmailAndPassword(email, password);

  doSignOut = () => this.auth.signOut();

  doPasswordReset = email => this.auth.sendPasswordResetEmail(email);

  doPasswordUpdate = password =>
    this.auth.currentUser.updatePassword(password);
}

export default Firebase;
```

Firebase/firebase.js

That’s the authentication interface for our React components that will connect to the Firebase API.

In the next lesson, we will run and verify our app for Firebase Authentication.