React has `useState`.

A React Developer has two states:

https://cederdorff.com/post-app-with-firebase/

Client Server Architecture

APP

REST API

DATA SOURCE

FETCH(GET, POST, PUT/PATCH, DELETE)

REQUEST

JSON DATA

RESPONSE

posts

users

CLIENT

SERVER

Firebase

Firebase

Table of Contents

1. Introduction to Firebase
2. Client Server
3. API
4. JSON
5. CRUD
6. REST
7. HTTP Requests
8. Realtime Database & REST API
9. Firebase Authentication
10. Firebase Security Rules

When someone asks you how to get data from a database in a js code

javascript.plainenglish.io/firebase-cloud-functions-tutorial-creating-a-rest-api-8cbc51479f80

What is Firebase?

Platform, a suite of tools & Backend-as-a-Service
for Web & App Development

100 SECONDS OF

https://www.youtube.com/watch?v=vAoB4VbhRzM

Rasmus
Firebase
made with mematic

Introducing Firebase

https://www.youtube.com/watch?v=iosNuIdQoy8

Build better apps

Cloud Firestore
Store and sync app data at global scale

ML Kit BETA
Machine learning for mobile developers

Cloud Functions
Run mobile backend code without managing servers

Authentication
Authenticate users simply and securely

Hosting
Deliver web app assets with speed and security

Cloud Storage
Store and serve files at Google scale

Realtime Database
Store and sync app data in milliseconds

Improve app quality

Crashlytics
Prioritize and fix issues with powerful, realtime crash reporting

Performance Monitoring
Gain insight into your app's performance

Test Lab
Test your app on devices hosted by Google

App Distribution BETA
Distribute pre-release versions of your app to your trusted testers

Grow your business

In-App Messaging BETA
Engage active app users with contextual messages

Google Analytics
Get free and unlimited app analytics

Predictions
Smart user segmentation based on predicted behavior

A/B Testing BETA
Optimize your app experience through experimentation

Cloud Messaging
Send targeted messages and notifications

Remote Config
Modify your app without deploying a new version

Dynamic Links
Drive growth by using deep links with attribution

MANY DEVICES
ONE PLATFORM

my firebase doesn't work. Can you help me?
read the documentation
But...
read the documentation

Do we need Backend Developers?

Realtime Database & REST API

Store and sync data in real time

REST API or SDK

https://firebase.google.com/products/realtime-database

Realtime Database REST API

https://firebase.google.com/docs/database/rest/start

```
export default function PostsPage() {
    const [posts, setPosts] = useState([]);
    const [showLoader, dismissLoader] = useIonLoading();

    async function getPosts() {
        const response = await fetch("https://race-rest-default-rtdb.firebaseio.com/posts.json");
        const data = await response.json();
        // map object into an array with objects
        const postsArray = Object.keys(data).map(key => ({ id: key, ...data[key] }));
        return postsArray;
    }

    async function getUsers() {
        const response = await fetch("https://race-rest-default-rtdb.firebaseio.com/users.json");
        const data = await response.json();
        // mapp object into an array with objects
        const users = Object.keys(data).map(key => ({ id: key, ...data[key] }));
        return users;
    }
```

javascript.plainenglish.io/firebase-cloud-functions-tutorial-creating-a-rest-api-8cbc51479f80

“

The Firebase Realtime Database is a cloud-hosted database. Data is stored as JSON and synchronized in realtime to every connected client.

We can use any Firebase Realtime Database URL as a REST endpoint. All we need to do is append .json to the end of the URL and send a request from our favorite HTTPS client.

”

https://firebase.google.com/docs/database/rest/start

- By following these REST principles, the Firebase Realtime Database REST API provides a simple and consistent way to work with the database.
- It allows developers to perform common operations using familiar HTTP methods, making it easier to integrate and interact with the database from different programming languages or platforms.

- In the context of Firebase Realtime Database REST API, you can perform CRUD operations using HTTP methods like POST, GET, PUT, PATCH, and DELETE to create, read, update, and delete data in the Firebase Realtime Database.
- This allows you to interact with the database and manipulate data effectively based on your application's needs.

my firebase doesn't work. Can you help me?
read the documentation
But...
read the documentation

How data is structured

```
{
    "posts": {
        "00": {
            "title": "I simply love loops.js 💫",
            "image": "https://images.unsplash.com/photo-1465
        },
        "01": {
            "title": "Slow Mornings in Bed ☕",
            "image": "https://images.unsplash.com/photo-1642
        },
        "02": {
            "title": "Every Day is Flower Day 🌷",
            "image": "https://images.unsplash.com/photo-1642
        }
    }
}
```

- One big object with objects.
- All objects have a unique key
- `"posts"`, `"00"`, `"01"` and `"02"` are all keys.
- `"posts"` is the key of the collection of post objects (list of posts).
- `"00"`, `"01"` and `"02"` are keys of a post object.
- `"title"` and `"image"` are keys inside of every post object. `"title"` and `"image"` contains values.
- Remember a property contains a key and a value.

```
{
    "posts": {
        "00": {
            "title": "I simply love loops.js 💫",
            "image": "https://images.unsplash.com/photo-1465
        },
        "01": {
            "title": "Slow Mornings in Bed ☕",
            "image": "https://images.unsplash.com/photo-1642
        },
        "02": {
            "title": "Every Day is Flower Day 🌷",
            "image": "https://images.unsplash.com/photo-1642
        }
    }
}
```

- Just think of an array of objects (posts array with post objects).
- We are just using an object to hold all the objects instead of an array.
- It's kind of a dictionary. The posts object contains unique prop names (keys) which easily can be "looked up".
- Then every post object (with title and image props) can be returned:
  - `["posts"]["02"]`
  - `["posts"]["00"]`
  - `["posts"]["01"]`
  - `["posts"]["01"]["title"]`

key and name of the collections

```
{
    "posts": {
        "00": {
            "title": "I simply love loops.js 💫",
            "image": "https://images.unsplash.com/photo-1465
        },
        "01": {
            "title": "Slow Mornings in Bed ☕",
            "image": "https://images.unsplash.com/photo-1642
        },
        "02": {
            "title": "Every Day is Flower Day 🌷",
            "image": "https://images.unsplash.com/photo-1642
        }
    }
}
```

key of the objects

GitHub

raw.githubusercontent.com/c

raw.githubusercontent.com/cederdorff/race/master/data/expo-posts.json

Raw Parsed


```
[
  {
    "id": "-M1Abcdefg123",
    "caption": "Beautiful sunset at the beach",
    "createdAt": 1687215634430,
    "image": "https://images.unsplash.com/photo-1566241832378-917a0f30db2c?ixlib=rb-
    4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1pYWdlfHx8fGVufDB8fHx8fA%3D%3D&auto=format&fit=crop&w=500&q=80",
    "user": {
      "id": "ZfPTVEMQKf9vhNiUh0bj",
      "image": "https://www.baaa.dk/media/b5ahrlra/maria-louise-bendixen.jpg?
      anchor=center&mode=crop&width=800&height=450&rnd=13279292165033000&format=webp",
      "mail": "mlbe@eaaa.dk",
      "name": "Maria Louise Bendixen",
      "title": "Senior Lecturer"
    },
    "location": {
      "city": "Aarhus",
      "country": "Denmark",
      "latitude": 56.1249541422341,
      "longitude": 10.218312555111716
    }
  },
  {
    "id": "-M1Xyz987pqr",
    "caption": "Exploring the city streets of Aarhus",
    "createdAt": 1687615634430,
    "image": "https://images.unsplash.com/photo-1559070169-a3077159ee16?ixlib=rb-
    4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1pYWdlfHx8fGVufDB8fHx8fA%3D%3D&auto=format&fit=crop&w=500&q=80",
    "user": {
      "id": "fTs84KRoYw5pRZEWCq2Z",
      "image": "https://share.cederdorff.dk/images/race.webp",
      "mail": "race@eaaa.dk",
      "name": "Rasmus Cederdorff",
      "title": "Senior Lecturer"
    },
    "location": {
      "city": "Aarhus",
      "country": "Denmark",
      "latitude": 56.157489440852906,
      "longitude": 10.20089048914843
    }
  },
  {
    "id": "-M2Def456stu",
    "caption": "Delicious food at the restaurant",
    "createdAt": 1687015634430,
    "image": "https://images.unsplash.com/photo-1548940740-204726a19be3?ixlib=rb-
    4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1pYWdlfHx8fGVufDB8fHx8fA%3D%3D&auto=format&fit=crop&w=500&q=80",
    "user": {
      "id": "fjpRTRTjZHwrq3tTLHri",
```


Firebase

raw.githubusercontent.com/c

raw.githubusercontent.com/cederdorff/race/master/data/expo-posts-users.json

Raw Parsed


```
{
  "posts": {
    "-M1Abcdefg123": {
      "caption": "Beautiful sunset at the beach",
      "createdAt": 1687215634430,
      "image": "https://images.unsplash.com/photo-1566241832378-917a0f30db2c?ixlib=rb-
      4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1pYWdlfHx8fGVufDB8fHx8fA%3D%3D&auto=format&fit=crop&w=500&q=80",
      "uid": "ZfPTVEMQKf9vhNiUh0bj",
      "location": {
        "city": "Aarhus",
        "country": "Denmark",
        "latitude": 56.1249541422341,
        "longitude": 10.218312555111716
      }
    },
    "-M1Xyz987pqr": {
      "caption": "Exploring the city streets of Aarhus",
      "createdAt": 1687615634430,
      "image": "https://images.unsplash.com/photo-1559070169-a3077159ee16?ixlib=rb-
      4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1pYWdlfHx8fGVufDB8fHx8fA%3D%3D&auto=format&fit=crop&w=500&q=80",
      "uid": "wrliEUxQxHdkH5uQ3a05OKksnt03",
      "location": {
        "city": "Aarhus",
        "country": "Denmark",
        "latitude": 56.157489440852906,
        "longitude": 10.20089048914843
      }
    },
    "-M2Def456stu": {
      "caption": "Delicious food at the restaurant",
      "createdAt": 1687015634430,
      "image": "https://images.unsplash.com/photo-1548940740-204726a19be3?ixlib=rb-
      4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1pYWdlfHx8fGVufDB8fHx8fA%3D%3D&auto=format&fit=crop&w=500&q=80",
      "uid": "fjpRTRTjZHwrq3tTLHri",
      "location": {
        "city": "Aarhus",
        "country": "Denmark",
        "latitude": 56.157489440852906,
        "longitude": 10.20089048914843
      }
    },
    "-M3Ghi789vwx": {
      "caption": "Exploring the city center of Aarhus",
      "createdAt": 1687615704430,
      "image": "https://images.unsplash.com/photo-1612624629424-ddde915d3dc5?ixlib=rb-
      4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1pYWdlfHx8fGVufDB8fHx8fA%3D%3D&auto=format&fit=crop&w=500&q=80",
      "uid": "pqzGY1MnHYm3I4Ca79Xn",
      "location": {
        "city": "Aarhus",
        "country": "Denmark",
```

Firebase Authentication

Sign Up, Sign In and Authenticate Users

https://firebase.google.com/products/auth

Authentication

https://firebase.google.com/docs/auth/web/start

console.firebase.google.com/u/0/project/expo-post-app/authent

Firebase

Expo Post App

Project Overview

Generative AI

Build with Gemini NEW

Project shortcuts

Realtime Database

Storage

Authentication

Firestore Database

Extensions

What's new

App Hosting NEW

Product categories

Build

Run

Analytics

All products

Spark
No cost $0/month

Upgrade

Authentication

Users | Sign-in method | Templates | Usage | Settings | Extensions

Sign-in providers

Add new provider

Provider	Status
Email/Password	Enabled

Advanced

SMS multi-factor authentication

Allow your users to add an extra layer of security to their account. Once enabled, integrated and configured, users can sign in to their account in two steps, using SMS. Learn more

MFA and other advanced features are available with Identity Platform, Google Cloud's complete customer identity solution built in partnership with Firebase. This upgrade is available on both the Spark and Blaze plans.

Upgrade to enable

Sign up new users

- Create User: Use `createUserWithEmailAndPassword` method to sign up new users.
- Handle Success: Access the user object on successful sign-up.
- Handle Errors: Catch and manage errors appropriately.

Sign up new users

```
function handleSignUp() {
  // Create user with email and password
  createUserWithEmailAndPassword(auth, mail, password)
    .then(userCredential => {
      // User Created and signed in
      const user = userCredential.user; // User
      console.log("Signed in as", user.email);
      router.replace("/"); // Redirect to home
    })
    .catch(error => {
      // Handle errors
      let errorMessage = error.code.split("/")[1];
      errorMessage = errorMessage.replaceAll("-", " ");
      setMessage(errorMessage);
    });
}
```

Sign in existing users

- Create a form that allows existing users to sign in with email address and password
- Pass validated email and password to `signInWithEmailAndPassword` method
- Handle Success: Access the user object on successful sign-up
- Handle Errors: Catch and manage errors appropriately

Authentication state observer

expo-post-app

_layout.jsx

app > _layout.jsx > RootLayoutNav > useEffect() callback

```
function RootLayoutNav() {
  useEffect(() => {
    onAuthStateChanged(auth, user => {
      if (!user) {
        // No user is signed in.
        router.replace("/sign-in");
      }
    });
  });

  return (
    <RootSiblingParent>
      <StatusBar style="light" />
      <ActionSheetProvider>
        <Stack>
          <Stack.Screen name="(tabs)" options={{ headerShown:
```

Authentication state observer

- Attach Observer to Authentication Object:
  - Necessary for pages that require user information
  - Observes changes in user's sign-in state
- Use `onAuthStateChanged` Method:
  - Attaches the observer to the global authentication object
  - Called whenever the user's sign-in state changes
- Retrieve User Information:
  - Observer gets user info when sign-in is successful

Manage Users in Firebase

console.firebase.google.com/u/0/project/expo-post-app/authentic

Firebase

Expo Post App

Project Overview

Generative AI

Build with Gemini NEW

Project shortcuts

Realtime Database

Storage

Authentication

Firestore Database

Extensions

What's new

App Hosting NEW

Product categories

Build

Run

Analytics

All products

Spark
No cost $0/month

Upgrade

Authentication

Users | Sign-in method | Templates | Usage | Settings | Extensions

Cross-origin redirect sign in on Google Chrome M115+ is no longer supported and will stop working on 24 June 2024.

Search by email address, phone number or user UID

Add user

Identifier	Providers	Created	Signed in	User UID
dob@eaaa.dk		3 May 2024	20 Jun 2024	4Q0YnA3Ox1XJXqOHWnmHf...
race@eaaa.dk		3 Jul 2023	30 Jun 2024	wrliEUxQxHdkH5uQ3a05OKks...

Rows per page 50 1 – 2 of 2

console.firebase.google.com/u/

Expo Post App

Realtime Database

Data | Rules | Backups | Usage | Extensions

https://expo-post-app-default-rtdb.firebaseio.com

```
https://expo-post-app-default-rtdb.firebaseio.com/
posts
users
  4Q0YnA3Ox1XJXqOHWnmHf0TnkcJ3
    image: "https://www.eaaa.dk/media/bdojel41/dan-okkels-brendstrup.jpg?anchor
    mail: "dob@eaaa.dk"
    name: "Dan Okkels Brendstrup"
    title: "Lecturer"
  JovNcr99C05elOL100k2
  XavNcr99C05erOLd0011
  ZfPTVEMQKf9vhNiUh0bj
  fjpRTRTjZHwrq3tTLHri
  nlvRcr44C05kuOL166k5
  pqzGY1MnHYm3I4Ca79Xn
  wrliEUxQxHdkH5uQ3a05OKksnt03
    image: "https://share.cederdorff.dk/images/race.webp"
    mail: "race@eaaa.dk"
    name: "Rasmus Cederdorff"
    title: "Senior Lecturer"
```

Database location: United States (us-central1)

Manage Users in Firebase

expo-post-app-v2-steps

profile.jsx

app > (tabs) > profile.jsx > Profile > getUser

```
27  export default function Profile() {
28    const [name, setName] = useState("");
29    const [title, setTitle] = useState("");
30    const [mail, setMail] = useState("");
31    const [image, setImage] = useState("");
32
33    // url to fetch (get and put) user data from Firebase Realtime Database
34    const url = `https://expo-post-app-default-rtdb.firebaseio.com/users/${auth.currentUser?.uid}.json`;
35
36    useEffect(() => {
37      setMail(auth.currentUser.email); // set mail to the current user email
38      getUser(); // fetch user data from Firebase Realtime Database
39    }, []);
40
41    async function getUser() {
42      const response = await fetch(url);
43      const userData = await response.json();
44
45      if (userData) {
46        // if userData exists set states with values from userData (data from Firebase Realtime Database)
47        setName(userData?.name); // set name to the value of the name property from userData
48        setTitle(userData?.title); // set title to the value of the title property from userData
49        setImage(userData?.image); // set image to the value of the image property from userData
50      }
51    }
52
```

Manage Users in Firebase

```
// url to fetch (get and put) user data from Firebase Realtime Database
const url = `https://expo-post-app-default-rtdb.firebaseio.com/users/${auth.currentUser?.uid}.json`;

async function handleSaveUser() {
  const userToUpdate = { name: name, mail: mail, title, image }; // create an object to hold the user to upda

  // send a PUT request to update user data in Firebase Realtime Database
  const response = await fetch(url, {
    method: "PUT",
    body: JSON.stringify(userToUpdate)
  });
  // if the response is ok, log the user data
  if (response.ok) {
    const data = await response.json();
    console.log("User data: ", data);
    Toast.show("Your profile has been updated");
  } else {
    Toast.show("Sorry, something went wrong");
  }
}
```

Firebase Security Rules

Used to control access to your Firebase services, such as Firestore, Realtime Database, and Cloud Storage. They allow you to specify conditions that must be met for read and write operations.

https://firebase.google.com/docs/rules

Firebase Security Rules

Key points:

- **Granular control:** You can set rules based on user authentication status, request data, or resource data.
- **Authentication**: Typically used with Firebase Auth to restrict data access to authenticated users.
- **Condition-based access**: You can define specific conditions, such as checking user roles or ownership before allowing access.
- **Real-time enforcement**: Rules are applied instantly to every database request.

Firebase Security Rules

Firebase Security Rules

No Security

All users can access - read and write to the database

FB-REST-RACE – Realtime D

console.firebase.google.com/u/0/project/fb-rest-race/database/fb-rest-race-default-rtdb/rules

FB-REST-RACE

Realtime Database

Rules playground

```
1  {
2    "rules": {
3      "users": {
4        ".read": true,
5        ".write": true
6      },
7      "posts": {
8        ".read": true,
9        ".write": true,
10         ".indexOn": ["uid"]
11     }
12   }
13 }
```

Basic Authentication

Allow read and write only for authenticated users:

```json
{
  "rules": {
    ".read": "auth != null",
    ".write": "auth != null"
  }
}
```

User-based Access

Allow each user to read/write their own data:

```json
{
  "rules": {
    "users": {
      "$user_id": {
        ".read": "auth != null && auth.uid === $user_id",
        ".write": "auth != null && auth.uid === $user_id"
      }
    }
  }
}
```

In this example, users can only access and modify data within their own node (e.g., `/users/{uid}`).

User-based Access & Validation

Allow each user to read/write their own data & data (values) are validated:

Role-based Access

Allow admins to read/write any data, but restrict normal users:

```json
{
  "rules": {
    "users": {
      "$user_id": {
        ".read": "auth != null && (auth.token.admin == true || auth.uid === $user_id)",
        ".write": "auth != null && (auth.token.admin == true || auth.uid === $user_id)"
      }
    }
  }
}
```

Here, an admin (defined by a custom token) can access all user data, while normal users can only access their own.

Code
Every
Day