

ReactJS

Projeto WhatsApp Clone

Aula 1

Repositório projeto: <https://github.com/vicentecalfo/react-fire-chat>

Vicente Calfo

vicentecalfo@gmail.com

<https://www.linkedin.com/in/vicentecalfo/>

Objetivo do Projeto

Criar um chat em tempo real usando o React para o *front-end* e os serviços do Firebase para *back-end*.

Link de demonstração: <https://firechat-two.vercel.app>

O que é o React

- Biblioteca JavaScript de código aberto, mantida pelo Facebook, que é usada para criar interfaces de usuário (UI);
- Permite criar componentes reutilizáveis.

O que é o Firebase

- O Firebase é uma plataforma de desenvolvimento de aplicativos móveis e web do Google que oferece uma ampla variedade de serviços em nuvem;
- Serviços inclusos: armazenamento em nuvem, banco de dados em tempo real, autenticação de usuário, hospedagem, mensagens e análise de acessos.

Criação do projeto

```
npx create-react-app react-fire-chat-app
```

NPM x NPX

- NPM é usado para gerenciar pacotes e dependências em um projeto Node.js;
- NPX é usado para executar pacotes sem instalá-los globalmente no sistema;
- NPX é um utilitário que vem com o NPM a partir da versão 5.2.0.

Estrutura Inicial - App.js

```
import "./App.css";

function App() {
  return (
    <div className="App">
      <header>
        <div className="logo">
          <div>
            
          </div>
          <div>
            <h1>Firechat</h1>
            <h2>React Firebase Chat</h2>
          </div>
        </div>
        <div className="user"></div>
      </header>
    </div>
  );
}

export default App;
```

className

A razão pela qual se usa **className** em vez de **class** é que a palavra "class" é uma palavra-chave reservada em JavaScript, o que significa que já possui um significado definido na linguagem e, portanto, não pode ser usada como nome de um atributo em uma tag HTML.

CSS Modules - App.css

```
.App {  
  display: grid;  
  grid-template-rows: 60px 1fr;  
  height: 100vh;  
  width: 100vw;  
}  
  
.App header {  
  background-color: #f0f2f5;  
  display: grid;  
  grid-template-columns: auto 1fr;  
  align-items: center;  
  height: 60px;  
}  
  
.App header .logo {  
  display: grid;  
  align-items: center;  
  justify-content: center;  
  grid-template-columns: 60px 1fr;  
}  
  
.App header .logo img {  
  height: 50px;  
}  
  
.App header .logo h1 {  
  font-size: 15px;  
  margin: 0;  
}  
  
.App header .logo h2 {  
  font-size: 12px;  
  margin: 0;  
  font-weight: 400;  
}  
  
.App header .user {  
  display: flex;  
  flex-direction: row-reverse;  
}
```

Criando o Primeiro Componente - SignIn.jsx

```
import "../SignIn.css";

function SignIn({ auth }) {
  const signInWithGoogle = () => {
    console.log(auth);
  };
  return (
    <div className="sign-in">
      <button onClick={signInWithGoogle}>
        <div>
          
        </div>
        <div>Sign in with Google</div>
      </button>
    </div>
  );
}

export default SignIn;
```

CSS - SignIn.css

```
.sign-in {  
  display: grid;  
  height: calc(100vh - 60px);  
  background-color: #fff;  
  align-items: center;  
  justify-content: center;  
}  
  
.sign-in button {  
  display: grid;  
  grid-template-columns: 60px 1fr;  
  outline: none;  
  align-items: center;  
  border: none;  
  padding: 10px 10px 10px 5px;  
  border-radius: 5px;  
  background-color: #306ccf;  
  cursor: pointer;  
  color: #fff;  
}  
  
.sign-in button img {  
  width: 40px;  
}
```

Atenção

Em ReactJS, os componentes são essencialmente funções JavaScript regulares, no entanto, é necessário que eles sejam escritos com a primeira letra maiúscula para que possam ser reconhecidos pelo compilador e, assim, renderizados corretamente na aplicação.

Props

```
// Usando o componente  
<SignIn auth={auth} />
```

```
// Código do Componente  
function SignIn({ auth }) {  
  const signInWithGoogle = () => {  
    console.log(auth);  
  };  
  return (  
    <div className="sign-in">  
      <button onClick={signInWithGoogle}>Sign in with Google</button>  
    </div>  
  );  
}
```

Componente - Sala de chat - ChatRoom

```
function ChatRoom() {  
  return (  
    <>  
      <div className='chat-room'>  
        <main>  
          <!-- mensagens -->  
        </main>  
        <div className='form'>  
          <form>  
            <input  
              placeholder='Vamos conversar'  
              type='text'  
            />  
            <button type='submit'>  
              <img src='./sent.png' alt="Botão de enviar"/>  
            </button>  
          </form>  
        </div>  
      </div>  
    </>  
  )  
}  
  
export default ChatRoom
```

CSS - ChatRoom.css - 1/4

```
.chat-room {  
  display: grid;  
  height: calc(100vh - 60px);  
  align-items: center;  
  justify-content: center;  
  grid-template-rows: 1fr 60px;  
  position: relative;  
}  
  
.chat-room::before {  
  content: " ";  
  display: block;  
  position: absolute;  
  left: 0;  
  top: 0;  
  width: 100%;  
  height: 100%;  
  opacity: 0.6;  
  background-image: url("https://www.toptal.com/designers/subtlepatterns/uploads/dot-grid.png");  
}
```

CSS - ChatRoom.css - 2/4

```
main {  
  height: calc(100vh - 120px);  
  display: block;  
  overflow: auto;  
  width: 100vw;  
  padding: 30px;  
  position: relative;  
}  
  
main::-webkit-scrollbar {  
  width: 0.55rem;  
}  
  
main::-webkit-scrollbar-track {  
  background: #333333;  
}  
  
main::-webkit-scrollbar-thumb {  
  background: #ffca27;  
}
```


CSS - ChatRoom.css - 3/4

```
.form {  
  background-color: #f0f2f5;  
  display: grid;  
  height: 60px;  
  align-items: center;  
  position: relative;  
  padding: 0 30px;  
}  
  
form {  
  display: grid;  
  grid-template-columns: 1fr 60px;  
  grid-gap: 20px;  
}  
  
form input[type="text"] {  
  display: flex;  
  border: 1px solid #ccc;  
  border-radius: 5px;  
  height: 40px;  
  padding: 0 20px;  
  outline: none;  
}
```

CSS - ChatRoom.css - 4/4

```
form button[type="submit"] {  
    display: grid;  
    width: 40px;  
    height: 40px;  
    background: #ffca27;  
    border-radius: 50%;  
    border: none;  
    outline: none;  
    align-items: center;  
    justify-content: center;  
    cursor: pointer;  
}  
  
form button[type="submit"] img {  
    display: block;  
    width: 25px;  
}
```

Mensagens - ChatMessage.jsx

```
import './ChatMessage.css';

function ChatMessage(props) {
  const messageClass = "sent"; //received

  return (
    <>
      <div className={`message ${messageClass}`}>
        <div className="bubble">
          
          <div className="display-message">
            <strong>Vicente Calfo</strong>
            <p>Esta é nossa mensagem de placeholder</p>
            <small>28/04/2003</small>
          </div>
        </div>
      </div>
    </>
  );
}

export default ChatMessage;
```

CSS - ChatMessage.css - 1/3

```
.message {  
  display: flex;  
  align-items: center;  
  margin-bottom: 10px;  
}  
.message .display-message {  
  display: grid;  
  grid-template-rows: auto 1fr auto;  
}  
.message .display-message strong {  
  display: block;  
  font-size: 12px;  
  color: rgba(0, 0, 0, 0.5);  
}  
  
.message .display-message p {  
  padding: 0;  
  margin: 0;  
  font-size: 14px;  
}
```

CSS - ChatMessage.css - 2/3

```
.message .display-message small {  
  font-size: 10px;  
  text-align: right;  
  margin-top: 5px;  
  color: rgba(0, 0, 0, 0.5);  
}  
  
.bubble {  
  display: flex;  
  border-radius: 10px;  
  align-items: flex-start;  
  padding: 20px;  
  max-width: 450px;  
}  
  
.avatar {  
  width: 42px;  
  height: 42px;  
  border-radius: 50%;  
  margin: 2px 15px 2px 5px;  
  padding: 5px;  
  background-color: rgba(255, 255, 255, 0.7);  
}
```

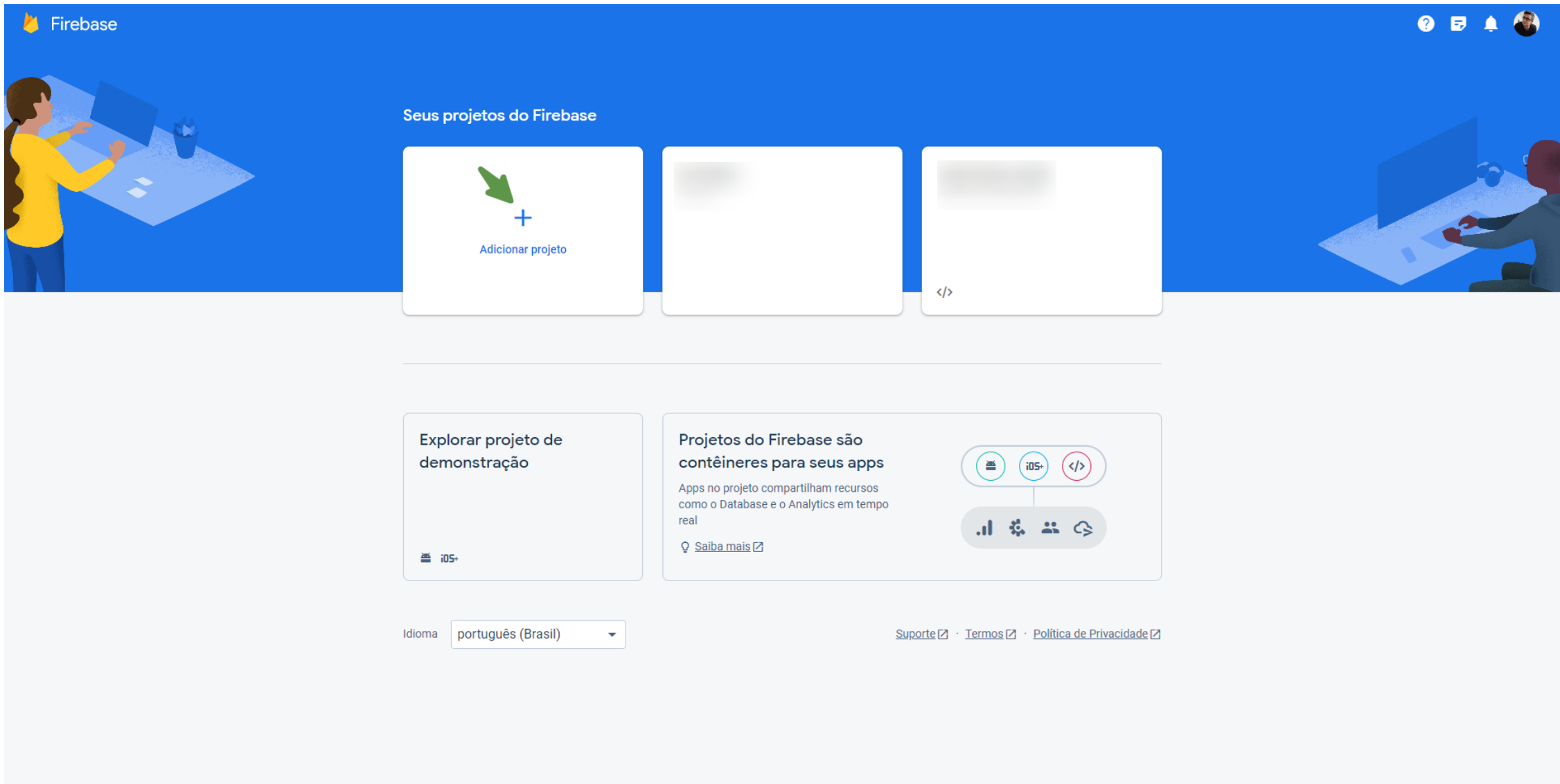
CSS - ChatMessage.css - 3/3

```
.sent {  
  flex-direction: row-reverse;  
}  
.sent .bubble {  
  flex-direction: row-reverse;  
  background-color: rgba(0, 0, 0, 0.05);  
  border-bottom-right-radius: 0px;  
}  
.sent p {  
  align-self: flex-end;  
}  
  
.received .bubble {  
  background-color: rgba(255, 211, 50, 0.5);  
  border-bottom-left-radius: 0px;  
}
```

Incluindo na sala de chat

```
import ChatMessage from "../ChatMessage";  
// ....  
return (  
  <>  
    <div className="chat-room">  
      <main>  
        <ChatMessage />  
      </main>  
      <div className="form">// form</div>  
    </div>  
  </>  
>);
```

Configuração do Firebase - Criar Projeto



The screenshot shows the Firebase console interface. At the top, the Firebase logo is on the left, and help, chat, and notification icons are on the right. The main heading is "Seus projetos do Firebase". Below it, there are three cards: the first has a green arrow pointing to a plus sign and the text "Adicionar projeto"; the second and third are blurred. Below the cards, there are two informational boxes. The left box is titled "Explorar projeto de demonstração" and shows an "iOS+" icon. The right box is titled "Projetos do Firebase são contêineres para seus apps" and explains that apps in a project share resources like Database and Analytics. It includes icons for Android, iOS, and code, and a "Saiba mais" link. At the bottom, there is a language dropdown set to "português (Brasil)" and a footer with links for "Suporte", "Termos", and "Política de Privacidade".

Seus projetos do Firebase

Adicionar projeto

Explorar projeto de demonstração

iOS+

Projetos do Firebase são contêineres para seus apps

Apps no projeto compartilham recursos como o Database e o Analytics em tempo real

Saiba mais

Idioma português (Brasil)

Suporte · Termos · Política de Privacidade

Configuração do Firebase - Criar Projeto

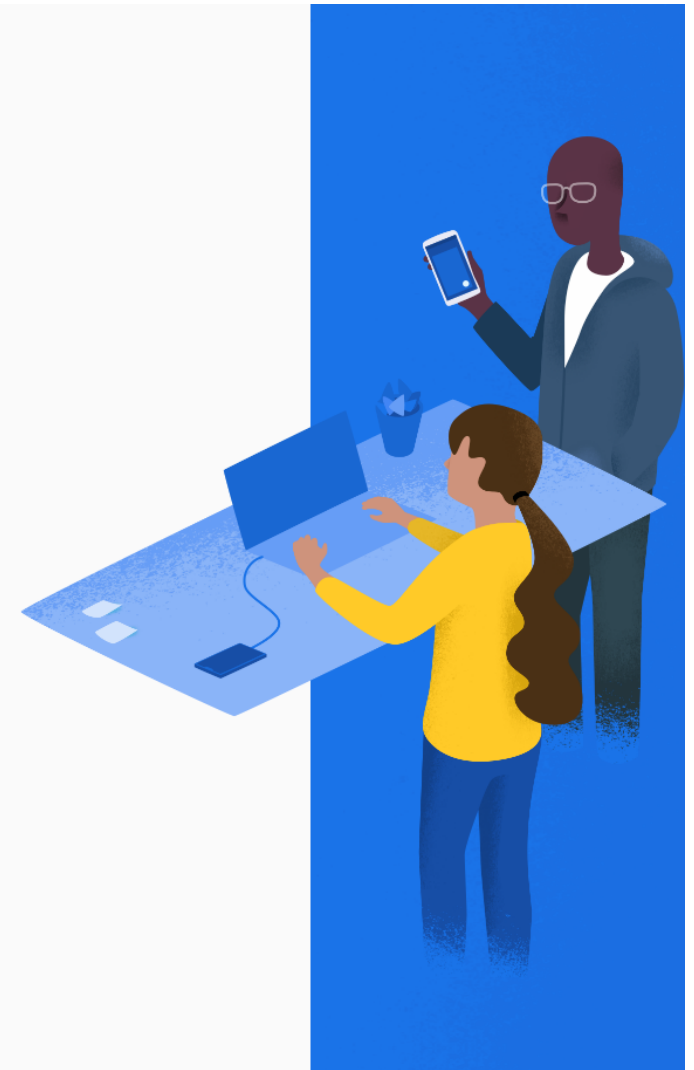
✕ Criar um projeto(Passo 1 de 3)

Vamos começar nomeando
o projeto[?]

Nome do projeto

 uerj-fire-chat

Continuar




Configuração do Firebase - Criar Projeto



✕ Criar um projeto(Passo 2 de 3)



Google Analytics para seu projeto do Firebase



O Google Analytics é uma solução de análise ilimitada e sem custos financeiros. Com ele, é possível segmentar, gerar relatórios e muito mais nos seguintes produtos: Firebase Crashlytics, Cloud Messaging, Mensagens no app, Configuração remota, Teste A/B e Cloud Functions.


O Google Analytics ativa:

 Teste A/B 

 Segmentação de usuários em produtos do Firebase 

 Usuários sem falhas 

 Gatilhos do Cloud Functions com base em eventos 

 Geração de relatórios ilimitada gratuita 

☒ Ativar o Google Analytics neste projeto
Recomendado

[Anterior](#)

[Continuar](#)



Configuração do Firebase - Criar Projeto

✕ Criar um projeto(Passo 3 de 3)

Configurar o Google Analytics

Escolha ou crie uma conta do Google Analytics ⓘ

 Default Account for Firebase ▾

Criar automaticamente uma nova propriedade nesta conta ✎

Após a criação do projeto, uma nova propriedade do Google Analytics será criada na conta selecionada e vinculada ao seu projeto do Firebase. Esse processo permitirá o fluxo de dados entre os produtos. Os dados da propriedade do Google Analytics exportados para o Firebase ficam sujeitos aos Termos de Serviço do Firebase, e os dados do Firebase importados para o Google Analytics ficam sujeitos aos Termos de Serviço do Analytics. [Saiba mais](#) 🔗

[Anterior](#)

[Criar projeto](#)



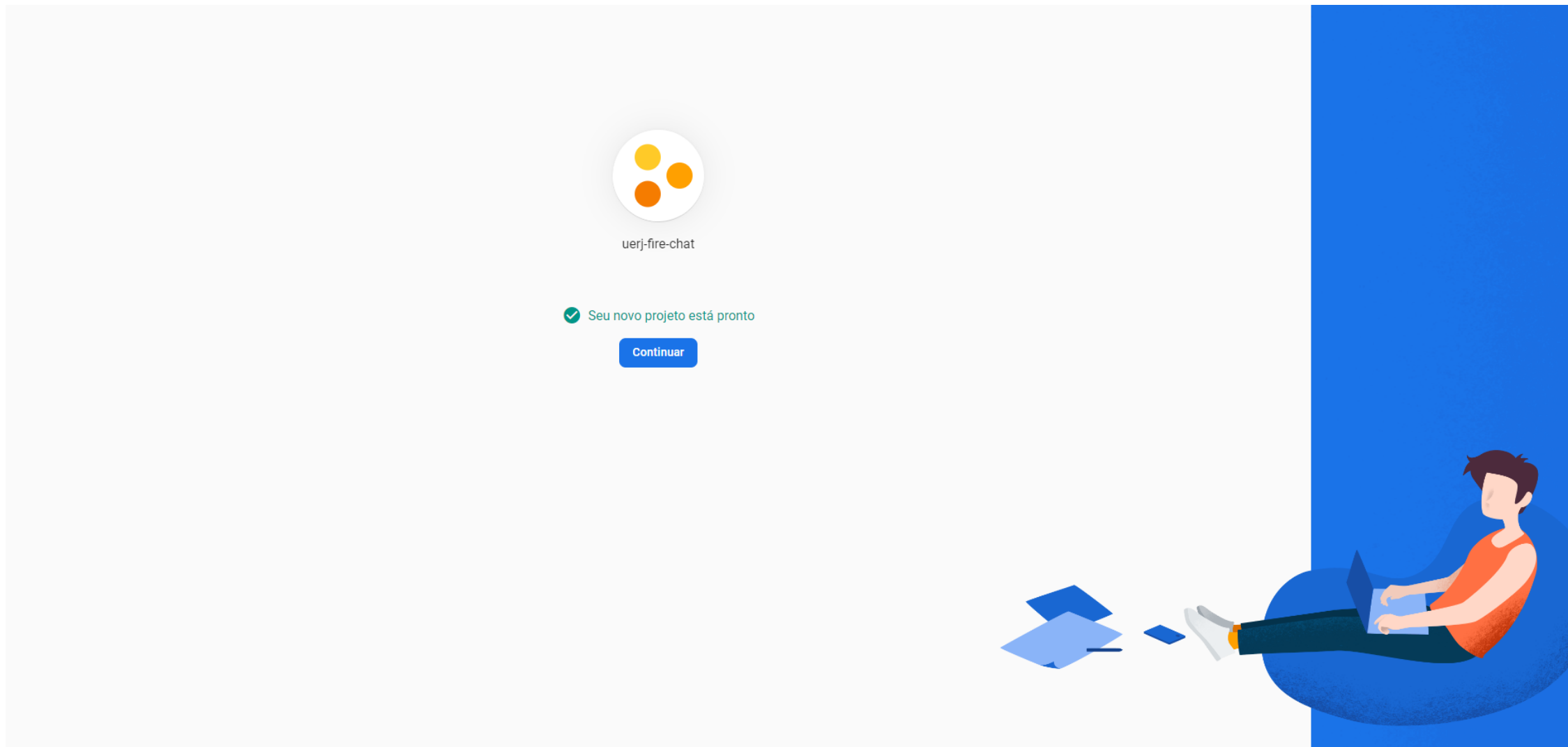
Configuração do Firebase - Criar Projeto



Criando seu projeto... Aguarde
uerj-fire-chat



Configuração do Firebase - Criar Projeto



Configuração do Firebase - Criar Projeto

The screenshot shows the Firebase console interface. On the left is a dark sidebar with the 'Firebase' logo and navigation links: 'Visão geral do projeto' (with a gear icon), 'Categorias dos produtos', a list of product categories ('Criação', 'Liberar e monitorar', 'Analytics', 'Engajamento'), 'Todos os produtos', and a 'Spark' section at the bottom stating 'Sem custos financeiros US\$ 0/mês'. The main area has a blue header with the project name 'uerj-fire-chat' and a 'Plano Spark' button. Below this, a large blue banner contains the text 'Comece adicionando o Firebase ao seu aplicativo' and icons for various platforms (iOS, Android, Web, etc.). An illustration of two people interacting with a digital interface is also present. At the bottom, a white box titled 'Armazene e sincronize dados de app em milissegundos' displays two featured services: 'Authentication' (with an ID card icon) and 'Cloud Firestore' (with a database icon).

uerj-fire-chat Plano Spark

Comece adicionando o Firebase ao seu aplicativo

Adicione um app para começar

Armazene e sincronize dados de app em milissegundos

Authentication
Autenticar e gerenciar usuários

Cloud Firestore
Atualizações em tempo real, consultas eficientes e escalonamento

Configuração do Firebase - Criar Projeto

× Adicionar o Firebase ao seu app da Web

1 Registrar app

Apelido do app ⓘ

uerj-fire-chat


☐ Configure também o **Firebase Hosting** para este app. [Saiba mais](#) ⓘ

O Hosting pode ser configurado a qualquer momento sem custos financeiros.

Registrar app

2 Adicionar o SDK do Firebase

[Acessar a documentação](#) ⓘ



Configuração do Firebase - Criar Projeto

× Adicionar o Firebase ao seu app da Web

✓ Registrar app

2 Adicionar o SDK do Firebase

☒ Usar o npm ☐ Usar a tag <script>

Se você já estiver usando o [npm](#) e um bundler de módulos, como [webpack](#) ou [Rollup](#), execute o seguinte comando para instalar o SDK mais recente ([saiba mais](#)):

```
$ npm install firebase
```

Depois inicialize o Firebase e comece a usar os SDKs dos produtos.

```
// Import the functions you need from the SDKs you need
import { initializeApp } from "firebase/app";
import { getAnalytics } from "firebase/analytics";
// TODO: Add SDKs for Firebase products that you want to use
// https://firebase.google.com/docs/web/setup#available-libraries

// Your web app's Firebase configuration
// For Firebase JS SDK v7.20.0 and later, measurementId is optional
const firebaseConfig = {
  apiKey: "AIzaSyB...",
  authDomain: "...",
  projectId: "...",
  storageBucket: "...",
  messagingSenderId: "...",
  appId: "1:123456789:web:...",
  measurementId: "G-..."
};

// Initialize Firebase
const app = initializeApp(firebaseConfig);
const analytics = getAnalytics(app);
```

[Acessar a documentação](#)



Configuração do Firebase - Autenticação

The screenshot shows the Firebase Authentication console. On the left is a dark sidebar with the 'Firebase' logo and navigation links: 'Visão geral do projeto', 'Atalhos do projeto' (with 'Authentication' highlighted), 'Categorias dos produtos' (with 'Criação', 'Liberar e monitorar', 'Analytics', and 'Engajamento' listed), and 'Todos os produtos'. The main area has a purple header with the project name 'uerj-fire-chat' and a user profile icon. Below the header, the 'Authentication' section is titled, with a description: 'Autenticar e gerenciar usuários a partir de uma variedade de provedores sem código do lado do servidor'. A 'Vamos começar' button is present. The bottom section, 'Saiba mais', contains three links: 'Como começar?', 'Como o Authentication funciona?', and 'De que forma o Authentication pode ser útil para mim?'. To the right of these links is a video player titled 'Introducing Firebase Authentication' showing various social media login screens with a red play button overlay.

Authentication

Autenticar e gerenciar usuários a partir de uma variedade de provedores sem código do lado do servidor

Vamos começar

Saiba mais

- Como começar?
Ver a documentação
- Como o Authentication funciona?
Ver a documentação
- De que forma o Authentication pode ser útil para mim?
Saiba mais

Introducing Firebase Authentication

Assistir no YouTube

Configuração do Firebase - Autenticação

The screenshot displays the Firebase Authentication console for a project named 'uerj-fire-chat'. The left sidebar contains navigation links for 'Visão geral do projeto', 'Authentication', and various product categories. The main content area is titled 'Authentication' and includes tabs for 'Users', 'Sign-in method' (selected), 'Templates', 'Usage', 'Settings', and 'Extensões'. Under 'Provedores de login', a message encourages adding a login method. Three categories of providers are shown: 'Provedores nativos' (Email/password, Smartphone, Anonymous), 'Outros provedores' (Google, Facebook, Play Games, Game Center, Apple, GitHub, Microsoft, Twitter, Yahoo), and 'Provedores personalizados' (OpenID Connect, SAML). The 'Avançado' section features 'Autenticação multifator por SMS' with a description and a link to learn more. A blue banner at the bottom promotes the Identity Platform upgrade for Spark and Blaze plans.

Provedores de login

Comece a usar o Firebase Auth ao adicionar seu primeiro método de login

Provedores nativos	Outros provedores	Provedores personalizados
E-mail/senha	Google	Facebook
Smartphone	Play Games	Game Center
Anônimo	Apple	GitHub
	Microsoft	Twitter
	Yahoo	

Avançado

Autenticação multifator por SMS

Permita que usuários incluam uma camada extra de segurança nas contas deles. Assim que essa autenticação for ativada, integrada e configurada, os usuários poderão fazer login com um processo de duas etapas por SMS. [Saiba mais](#)

★ A MFA e outros recursos avançados estão disponíveis no Identity Platform, uma solução completa de identidade do cliente do Google Cloud, criada em parceria com o Firebase. Esse upgrade está disponível nos planos Spark e Blaze.

[Fazer upgrade para ativar](#)

Configuração do Firebase - Autenticação

The screenshot shows the Firebase Authentication configuration interface. On the left is a dark sidebar with the 'Firebase' logo and navigation links: 'Visão geral do projeto', 'Atalhos do projeto', 'Authentication', 'Todas os produtos', and 'Spark'. The main content area is titled 'Authentication' and includes tabs for 'Users', 'Sign-in method' (selected), 'Templates', 'Usage', 'Settings', and 'Extensões'. Under the 'Sign-in method' tab, the 'Provedores de login' section shows the 'Google' provider is active. A text block explains that Google login is configured for Apple and Web apps, but requires SHA-1 fingerprinting for Android apps. Below this, a configuration box prompts the user to update settings at the project level, showing the 'Nome público do projeto' as 'project-34256166750' and the 'E-mail de suporte do projeto' as 'Não configurado'. At the bottom, there are expandable sections for 'Adicionar IDs de cliente à lista de permissões usando projetos externos (opcional)' and 'Configuração do SDK da Web', along with 'Cancelar' and 'Salvar' buttons.

uerj-fire-chat

Authentication

Users Sign-in method Templates Usage Settings Extensões **NOVO**

Provedores de login

Google Ativar

O Login do Google é configurado automaticamente nos seus apps da Apple e da Web conectados. Para configurar esse recurso nos seus apps Android, é necessário adicionar a [impressão digital SHA1](#) em cada app nas [Configurações do projeto](#).

Atualize abaixo a [configuração no nível do projeto](#) para continuar

Nome público do projeto

project-34256166750

E-mail de suporte do projeto

Não configurado

Selecione um endereço de e-mail

Adicionar IDs de cliente à lista de permissões usando projetos externos (opcional)

Configuração do SDK da Web

Cancelar Salvar

Configuração do Firebase - Autenticação

uerj-fire-chat ▾

Authentication

Users Sign-in method Templates Usage Settings Extensões **NOVO**

Provedores de login

Provedor	Status
Google	ativado

[Adicionar novo fornecedor](#)

Avançado

Autenticação multifator por SMS

Permita que usuários incluam uma camada extra de segurança nas contas deles. Assim que essa autenticação for ativada, integrada e configurada, os usuários poderão fazer login com um processo de duas etapas por SMS. [Saiba mais](#)

★ A MFA e outros recursos avançados estão disponíveis no Identity Platform, uma solução completa de identidade do cliente do Google Cloud, criada em parceria com o Firebase. Esse upgrade está disponível nos planos Spark e Blaze.

[Fazer upgrade para ativar](#)

Spark
Sem custos financeiros US\$ 0/mês Faz...

Instalando o SDK do Firebase

```
npm install firebase
```

Instalando o React Firebase Hooks

```
npm i react-firebase-hooks
```

Inserindo as configurações no React - .env

Criar um arquivo `.env`

```
REACT_APP_API_KEY = ""  
REACT_APP_AUTH_DOMAIN = ""  
REACT_APP_PROJECT_ID = ""  
REACT_APP_STORAGE_BUCKET = ""  
REACT_APP_MESSAGING_SENDER_ID = ""  
REACT_APP_APP_ID = ""  
REACT_APP_MEASUREMENT_ID = ""
```

Inserindo as configurações no React - .App.js

```
import { initializeApp } from "firebase/app";
import { getAuth } from "firebase/auth";
import { getFirestore } from "firebase/firestore";

const firebaseConfig = {
  apiKey: process.env.REACT_APP_API_KEY,
  authDomain: process.env.REACT_APP_AUTH_DOMAIN,
  projectId: process.env.REACT_APP_PROJECT_ID,
  storageBucket: process.env.REACT_APP_STORAGE_BUCKET,
  messagingSenderId: process.env.REACT_APP_MESSAGING_SENDER_ID,
  appId: process.env.REACT_APP_APP_ID,
  measurementId: process.env.REACT_APP_MEASUREMENT_ID,
};

const firebaseApp = initializeApp(firebaseConfig);
const auth = getAuth(firebaseApp);
const firestore = getFirestore(firebaseApp);

function App() {}
```


Inserindo o React Firebase Hooks

```
import { useAuthState } from "react-firebase-hooks/auth";  
  
function App() {  
  const [user] = useAuthState(auth);  
}
```

Preparando o botão de login (Google) - SignIn.jsx

```
import "./SignIn.css";
import { GoogleAuthProvider, signInWithPopup } from "firebase/auth";

function SignIn({ auth }) {
  const signInWithGoogle = () => {
    const provider = new GoogleAuthProvider();
    signInWithPopup(auth, provider);
  };
  return (
    <div className="sign-in">
      <button onClick={signInWithGoogle}>
        <div>
          
        </div>
        <div>Sign in with Google</div>
      </button>
    </div>
  );
}

export default SignIn;
```

Criando o SignOut.jsx

```
import './SignOut.css';

function SignOut({ auth }) {
  return (
    auth.currentUser && (
      <div className="sign-out">
        <img
          className="avatar"
          src={
            auth?.currentUser?.photoURL ||
            "https://api.dicebear.com/6.x/bottts/png"
          }
        />
        <button onClick={() => auth.signOut()}>
          
        </button>
      </div>
    )
  );
}

export default SignOut;
```

CSS - SignOut.css

```
.sign-out {  
  display: grid;  
  grid-template-columns: 60px 60px;  
  height: 60px;  
  align-items: center;  
}  
  
.avatar {  
  width: 40px;  
  height: 40px;  
  border-radius: 50%;  
}  
  
button {  
  height: 60px;  
  display: grid;  
  align-items: center;  
  justify-content: center;  
  border: none;  
  background-color: transparent;  
  cursor: pointer;  
}  
button:hover {  
  background-color: rgba(0, 0, 0, 0.06);  
}  
  
img {  
  height: 35px;  
  display: block;  
}
```

Buscando Mensagens do Firebase - HTML

```
return (  
  <>  
    <div className="chat-room">  
      <main>  
        {messages &&  
          messages.map((msg, index) => (  
            <ChatMessage key={index} message={msg} auth={auth} />  
          ))}  
      </main>  
    </div>  
  </>  
);
```

Importando Mensagens do Firebase - JS

```
import { collection, limit, orderBy, query } from "firebase/firestore";
import { useCollectionData } from "react-firebase-hooks/firestore";
import ChatMessage from "../ChatMessage";

function ChatRoom() {
  const messagesRef = collection(firestore, "messages");
  const dbQuery = query(messagesRef, orderBy("createdAt"), limit(25));
  const collectionDataOption = { idField: "id" };
  const [messages] = useCollectionData(dbQuery, collectionDataOption);
  // restante do código
}
```

Atribuição por Desestruturação

```
const [a, b] = array; // [1,2]
```

```
const { a, b } = obj;
```

```
/**
```

```
* {
```

```
*   a: 1,
```

```
*   b: 2
```

```
* }
```

```
* /
```

Importando Mensagens do Firebase - Atualizando ChatMessage

```
return (
  <>
    <div className={`message ${messageClass}`} >
      <div className="bubble">
        {messageClass === "sent" ? (
          ""
        ) : (
          <img
            className="avatar"
            alt="Foto de Avatar"
            src={photoURL || "https://api.dicebear.com/6.x/bottts/png"}
          />
        )}
      <div className="display-message">
        {messageClass === "sent" ? "" : <strong>{displayName}</strong>}
        <p>{text}</p>
        <small>{sentTime === "Invalid Date" ? "" : sentTime}</small>
      </div>
    </div>
  </div>
</>
);
```


Importando Mensagens do Firebase - Atualizando ChatMessage

```
function ChatMessage(props) {  
  const { text, uid, photoURL, createdAt, displayName } = props.message;  
  const auth = props.auth;  
  
  const sentTime = new Date(createdAt?.seconds * 1000).toLocaleTimeString([], {  
    hour: "2-digit",  
    minute: "2-digit",  
  });  
  
  const messageClass = uid === auth?.currentUser?.uid ? "sent" : "received";  
  
  // ...  
}
```

Atualização do ChatRoom

1. Precisamos da informação do usuário.
2. Precisamos da instância (autenticada) do banco de dados;

App.js

Enviando os parâmetros.

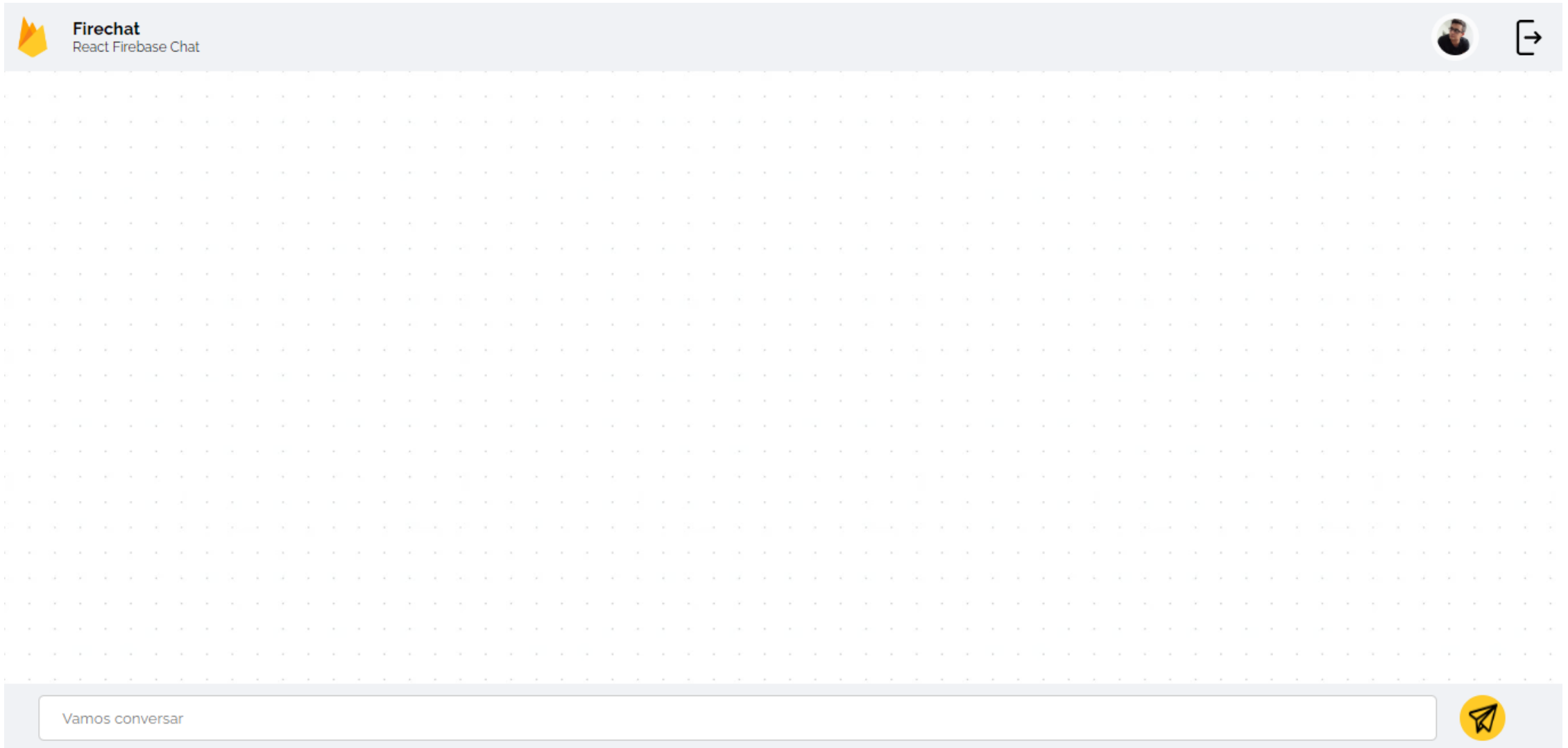
```
{
  user ? (
    <ChatRoom firestore={firestore} auth={auth} />
  ) : (
    <SignIn auth={auth} />
  );
}
```

ChatRoom.jsx

Recebendo os parâmetros.

```
function ChatRoom({ firestore, auth }) {  
  // ....  
}
```

Status Atual da Aplicação



Testando a Lista de Mensagens

The screenshot displays the Firebase console interface. On the left is a dark sidebar with the 'Firebase' logo and navigation links: 'Visão geral do projeto', 'Atalhos do projeto', 'Authentication', 'Firestore Database', 'Todas os produtos', and 'Spark'. The main content area has an orange header with 'uerj-fire-chat' and a 'Cloud Firestore' section. This section includes the text 'Atualizações em tempo real, consultas eficientes e escalonamento automático' and a 'Criar banco de dados' button. Below this is a white banner with a star icon and the text 'O Cloud Firestore é a opção certa para você?' followed by a link 'Compare os bancos de dados'. Further down is a 'Saiba mais' section with three cards: 'Como começar?' (link to documentation), 'Qual será o custo do Cloud Firestore?' (link to prices), and 'De que forma o Cloud Firestore pode ser útil para mim?' (link to 'Saiba mais'). To the right of these cards is a video player titled 'Introducing Cloud Firestore' with a play button and a 'Assistir no YouTube' button at the bottom.

Testando a Lista de Mensagens

Cloud Firestore

Criar banco de dados

1 **Regras seguras para o Cloud Firestore** 2 Defina o local do Cloud Firestore

Após definir a estrutura, é preciso criar regras para proteger seus dados. [Saiba mais](#)

☒ **Iniciar no modo de produção**
Seus dados são particulares por padrão. O acesso de leitura/gravação do cliente vai ser concedido apenas se especificado por suas regras de segurança.

☐ **Iniciar no modo de teste**
Por padrão, seus dados estão definidos para permitir uma configuração rápida. Porém, você precisa atualizar suas regras de segurança em até 30 dias para permitir o acesso de leitura/gravação do cliente em longo prazo.

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /{document=**} {
      allow read, write: if false;
    }
  }
}
```

Todas as leituras e gravações de terceiros vão ser negadas

Ativar o Cloud Firestore impede que você use o Cloud Datastore neste projeto

Cancelar **Avançar**

Testando a Lista de Mensagens

Cloud Firestore
Atualizações em tempo real, consultas eficientes e escalonamento automático

Criar banco de dados

1 Regras seguras para o Cloud Firestore — 2 **Defina o local do Cloud Firestore**

Sua configuração de local é onde os dados do Cloud Firestore vão ser armazenados.

⚠ Depois de configurar o local, não vai ser possível fazer mudanças nele. Além disso, essa definição de local vai ser onde ficará seu bucket padrão do Cloud Storage. [Saiba mais](#)

Local do Cloud Firestore
southamerica-east1 (São Paulo)

Ativar o Cloud Firestore impede que você use o Cloud Datastore neste projeto

[Cancelar](#) [Ativar](#)

Testando a Lista de Mensagens

The screenshot shows the Firebase Cloud Firestore console for a project named 'uerj-fire-chat'. The left sidebar contains the Firebase logo and navigation links: 'Visão geral do projeto', 'Authentication', 'Firestore Database', and 'Todos os produtos'. The main area is titled 'Cloud Firestore' and has tabs for 'Dados', 'Regras', 'Índices', 'Uso', and 'Extensões'. The 'Dados' tab is active, showing a 'Visualização do painel' view. The main content area displays a list of collections for the 'uerj-fire-chat' database, with a '+ Iniciar coleção' button. A large circular icon with a database symbol is centered on the screen, with the text 'Seu banco de dados está pronto, basta adicionar dados.' below it. The bottom status bar indicates the database location as 'Local do banco de dados: southamerica-east1'.

Firebase

uerj-fire-chat

Cloud Firestore

Dados Regras Índices Uso Extensões NOVO

Visualização do painel Criador de consultas

+ Iniciar coleção

Seu banco de dados está pronto, basta adicionar dados.

Local do banco de dados: southamerica-east1

Testando a Lista de Mensagens

uerj-fire-chat

Cloud Firestore

Dados **Regras** Índices Uso Extensões **NOVO**

Editar regras Monitorar regras Desenvolver e testar

Hoje • 11:21 AM

Hoje • 11:03 AM

```
1 rules_version = '2';
2 service cloud.firestore {
3   match /databases/{database}/documents {
4     match /{document=**} {
5       allow read, write: if false;
6     }
7     match /messages/{docId} {
8       allow read : if request.auth.uid != null;
9       allow create : if canCreateMessage();
10    }
11    function canCreateMessage(){
12      let isSignedIn = request.auth.uid != null;
13      let isOwner = request.auth.uid == request.resource.data.uid;
14      return isSignedIn && isOwner;
15    }
16  }
17 }
```

Laboratório de testes de regras
Faça experimentos e explore as Regras de segurança

Spark
Sem custos financeiros US\$ 0/mês Faz...

Regras de Acesso do Banco - Firestore

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /{document=**} {
      allow read, write: if false;
    }
    match /messages/{docId} {
      allow read : if request.auth.uid != null;
      allow create : if canCreateMessage();
    }
    function canCreateMessage(){
      let isSignedIn = request.auth.uid != null;
      let isOwner = request.auth.uid == request.resource.data.uid;
      return isSignedIn && isOwner;
    }
  }
}
```

React Hooks

Os hooks permitem o uso de state e outros recursos que antes só eram possíveis dentro do React através de classes.

useState

O React Hook **useState** é uma função que permite a criação e o gerenciamento do estado local de um componente. Ao utilizar esse hook, é possível definir uma variável de estado e uma função que atualiza essa variável, que são retornadas em um array como resultado. Dessa forma, é possível modificar e atualizar o estado do componente de forma eficiente, evitando problemas com a mutabilidade de variáveis em JavaScript.

```
const [state, setState] = useState(initialState);
```

Atualizando o ChatRoom.jsx

```
import { useRef, useState } from "react";

function ChatRoom({ firestore, auth }) {
  // ... código
  const [formValue, setFormValue] = useState("");

  // formValue -> Valor do Formulário
  // setFormValue -> Função para alterar o estado de formValue
}
```

Convenções do useState

- `useState` retorna um *array* com 2 índices (2 posições);
- Primeiro valor -> variável que guarda o estado (o valor);
- Segundo valor -> função que nos permite tualizar o estado (o valor). **Atenção:** Podemos nomear a função como desejar, embora seja uma conveção usar o *prefixo set*.

```
// exemplos:  
const [formValue, setFormValue] = useState("");  
const [userName, setUsername] = useState("");
```

Implementando o comportamento do "enviar"

```
return (  
  <>  
    <div className="chat-room">  
      <main>  
        {messages &&  
          messages.map((msg, index) => (  
            <ChatMessage key={index} message={msg} auth={auth} />  
          ))}  
        { /* <span ref={forceBottomScrollElement}></span> */}  
      </main>  
      <div className="form">  
        <form onSubmit={sendMessage}>  
          <input  
            value={formValue}  
            onChange={(e) => setFormValue(e.target.value)}  
            placeholder="Vamos conversar"  
            type="text"  
          />  
          <button type="submit" disabled={!formValue}>  
              
          </button>  
        </form>  
      </div>  
    </div>  
  </>  
>);
```


Comportamento do Clique (enviar)

```
const sendMessage = async (e) => {  
  const { uid, photoURL, displayName } = auth.currentUser;  
  
  const docRef = await addDoc(messagesRef, {  
    text: formValue,  
    createdAt: serverTimestamp(),  
    uid,  
    photoURL,  
    displayName,  
  });  
  
  setFormValue("");  
};
```

Atualizando o import do firestore

```
import {  
  addDoc,  
  collection,  
  limit,  
  orderBy,  
  query,  
  serverTimestamp,  
} from "firebase/firestore";
```

Problema - A página está recarregando!

```
const sendMessage = async (e) => {  
  console.log(e);  
  e.preventDefault();  
  // restante do código  
}
```