

React JS

★ AULA 07 ★

★ PROFESSOR
MAROMO ★



React JS

Integração com Backend firebase

Firebase



Agenda

- Criação uma aplicação web de lista de tarefas que permite aos usuários:
 - registrarem,
 - atualizarem e
 - excluírem suas tarefas pessoais.
- A aplicação será desenvolvida utilizando o React como framework de front-end e o Firebase como plataforma de backend para autenticação de usuários e armazenamento dos dados.



Firebase

O Firebase é uma plataforma de desenvolvimento de aplicativos móveis e web fornecida pelo Google. Ele oferece um conjunto abrangente de ferramentas e serviços para ajudar os desenvolvedores a criar, melhorar e dimensionar aplicativos com facilidade.



Principais componentes e recursos do Firebase

- **Firebase Authentication:** Um serviço de autenticação que permite adicionar autenticação de usuário aos seus aplicativos.
- **Cloud Firestore:** Um banco de dados em tempo real e escalável do Firebase
- **Firebase Realtime Database:** Outra opção de banco de dados em tempo real do Firebase, que armazena os dados do aplicativo em formato JSON.
- **Firebase Storage:** Um serviço de armazenamento em nuvem para armazenar e servir arquivos, como imagens, vídeos e outros recursos estáticos do seu aplicativo.
- **Firebase Hosting:** Um serviço de hospedagem de sites estáticos e aplicativos da web.



Firebase authentication

- O Firebase Authentication simplifica a criação de um sistema de autenticação seguro e confiável, lidando com várias tarefas complexas, como armazenamento de senhas, verificação de identidade e gerenciamento de sessões de usuário.
- Com algumas linhas de código, é possível implementar um fluxo de login, registro, recuperação de senha e outros recursos de autenticação em seu aplicativo.



Firebase cloud firestore

- O Firebase Cloud Firestore é um banco de dados NoSQL em tempo real que oferece uma estrutura flexível, sincronização em tempo real e poderosas consultas.
- Com recursos avançados, integração perfeita com outros serviços do Firebase e escalabilidade confiável, ele permite que os desenvolvedores criem aplicativos modernos e reativos, gerenciando dados de forma eficiente e proporcionando uma experiência em tempo real para os usuários.



A man with a shaved head, wearing a black t-shirt, is shown from the chest up. He has his right hand resting against his chin, looking upwards and to the left with a thoughtful expression. A large, white, rounded rectangular speech bubble originates from his mouth and points towards the text "Por que usar o Firebase".

Por que usar o
Firebase





Justificativa

- O **Firebase** é uma escolha vantajosa para o desenvolvimento de aplicativos devido aos seguintes motivos. Primeiro, o Firebase oferece uma ampla gama de serviços integrados, como autenticação de usuários, armazenamento em nuvem, banco de dados em tempo real e muito mais, eliminando a necessidade de configurar e gerenciar infraestrutura complexa. Isso acelera o desenvolvimento, permite que os desenvolvedores se concentrem na lógica do aplicativo e reduz os custos de desenvolvimento e manutenção.
- Além disso, o **Firebase fornece recursos avançados**, como sincronização em tempo real, escalabilidade confiável e segurança robusta, garantindo uma experiência de usuário rica, alta disponibilidade e proteção dos dados dos usuários.

Projeto no Firebase



- Vá para o Console do Firebase (<https://console.firebaseio.google.com/>) e crie um novo projeto.
- No painel do projeto, clique em "Adicionar app" e selecione a opção da web.
- Siga as instruções para registrar o aplicativo.
- Copie as credenciais do Firebase (apiKey, authDomain, projectId) que serão usadas posteriormente.

Projeto no Firebase



X Criar um projeto(Passo 1 de 3)

Vamos começar nomeando o projeto [?]

Nome do projeto

sample07

sample07-feea3 Selecionar recurso pai

Continuar

Projeto no Firebase



X Criar um projeto(Passo 2 de 2)

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:

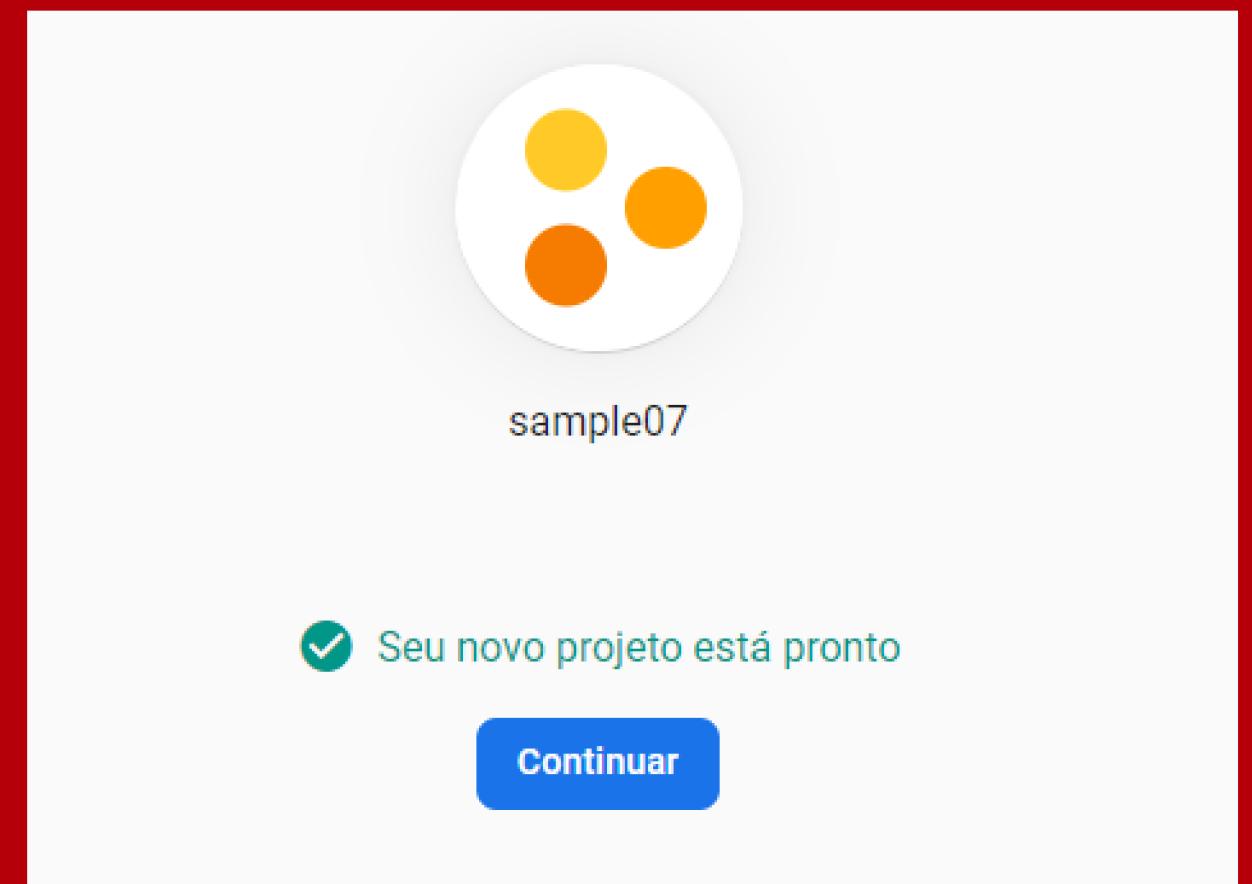
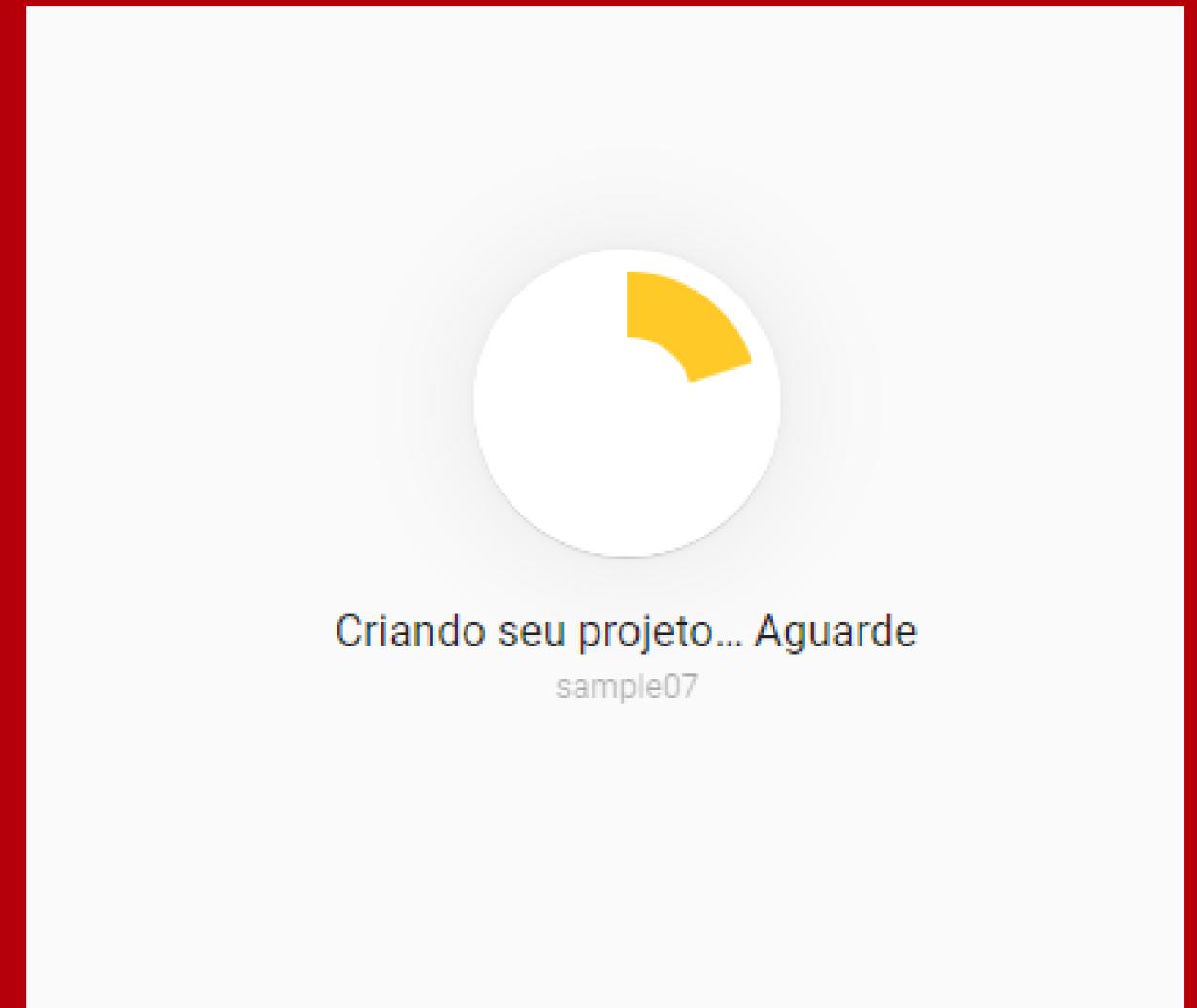
- [Teste A/B](#) ⓘ
- [Segmentação de usuários em produtos](#) ⓘ
do Firebase
- [Usuários sem falhas](#) ⓘ
- [Gatilhos do Cloud Functions com base](#) ⓘ
em eventos
- [Geração de relatórios ilimitada gratuita](#) ⓘ

Ativar o Google Analytics neste projeto
Recomendado

[Anterior](#)

[Criar projeto](#)

Projeto no Firebase



Projeto no Firebase



Projeto no Firebase



× Adicionar o Firebase ao seu app da Web

1 Registrar app

Apelido do app ⓘ
sample07

Configure também [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

Projeto no Firebase

Reserve este trecho, copiando



Depois initialize 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";
// 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
const firebaseConfig = {
  apiKey: "AIzaSyDyJLWzXGKUOOGHs",
  authDomain: "sample07-feeaa.firebaseioapp.com",
  projectId: "sample07-feeaa3",
  storageBucket: "sample07-feeaa3.appspot.com",
  messagingSenderId: "198929100013",
  appId: "1:198929100013:web:f7ade66bf36a62eec5b0e4"
};

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



Para o laboratório serão utilizados:

- **Create React App**: ferramenta que facilita a configuração inicial de um projeto React.
- **firebase**: O pacote Firebase é necessário para utilizar os serviços do Firebase, como autenticação de usuários, armazenamento em nuvem e banco de dados em tempo real.
- **bootstrap**: O Bootstrap é uma biblioteca de componentes e estilos CSS que facilita a criação de interfaces responsivas e visualmente agradáveis.
- **react-router-dom**: O react-router-dom é uma biblioteca que permite a criação de rotas e o controle de navegação em um aplicativo React



Laboratório



Desenvolvimento de uma aplicação SPA usando o React JS. Nome da aplicação **sample07.**

```
npx create-react-app sample07
```

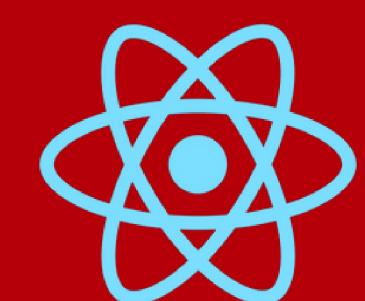
```
cd sample07
```

```
npm install firebase
```

```
npm install bootstrap
```

```
npm install react-router-dom
```

```
code .
```



React JS





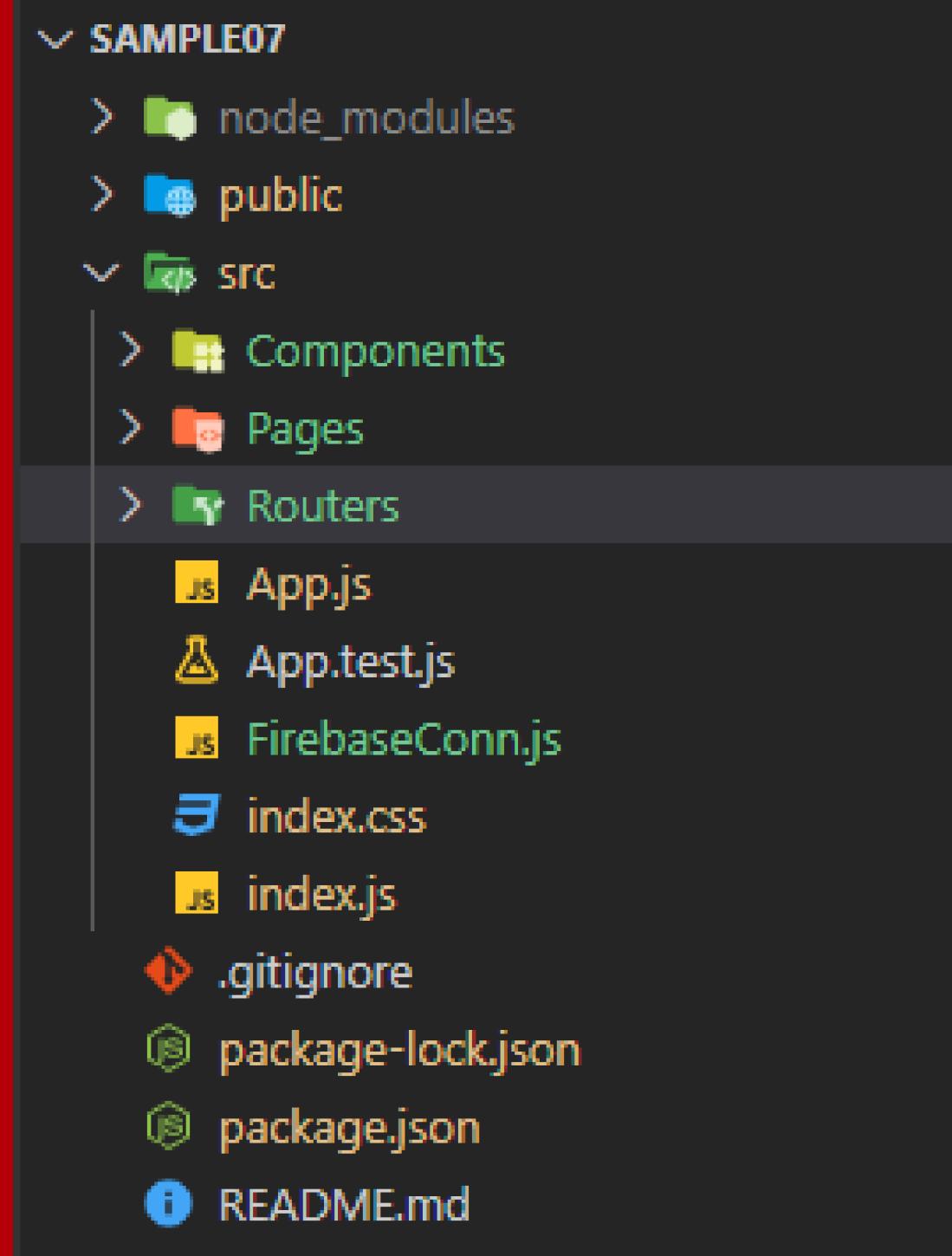
Passo 2: Estrutura inicial

Crie três diretórios (folders) chamados:

- a) Components
- b) Pages
- c) Routers

Crie três arquivos:

- a) FirebaseConn.js
- b) index.css
- c) index.js



```
SAMPLE07
├── node_modules
├── public
└── src
    ├── Components
    ├── Pages
    └── Routers
    ├── App.js
    ├── App.test.js
    ├── FirebaseConn.js
    ├── index.css
    ├── index.js
    └── README.md
```

Passo 3

No arquivo FirebaseConn.js cole o texto reservado de configurações, como mostra a figura:

Deixe igual mas com suas configurações do firebase

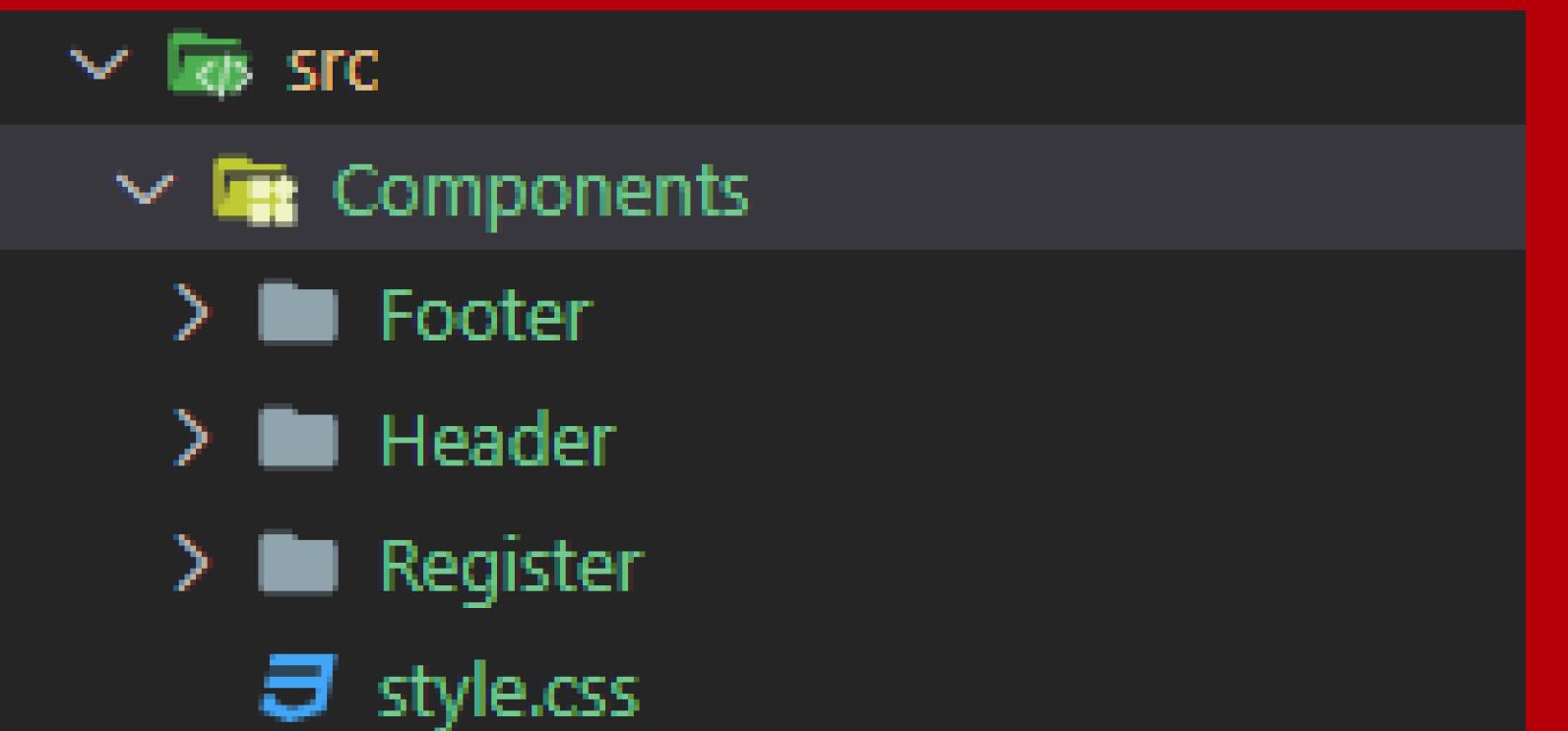


```
src > js FirebaseConn.js > ...
1 import { initializeApp } from "firebase/app"
2 import { getFirestore } from "firebase/firestore"
3 import { getAuth } from "firebase/auth"
4
5 // Your web app's Firebase configuration
6 const firebaseConfig = {
7   apiKey: "XXXXXXXXXXXXXXXXXXXXXX",
8   authDomain: "sample07-e4a52.firebaseioapp.com",
9   projectId: "sample07-e4a52",
10  storageBucket: "sample07-e4a52.appspot.com",
11  messagingSenderId: "385365509462",
12  appId: "1:385365509462:web:1ca60cd46139c6d80802f7"
13 };
14
15 // Initialize Firebase
16 const firebaseApp = initializeApp(firebaseConfig);
17
18 const db = getFirestore(firebaseApp);
19
20 const auth = getAuth(firebaseApp);
21
22 export { db, auth };
```

Passo 4

Crie os componentes abaixo no diretório Components:

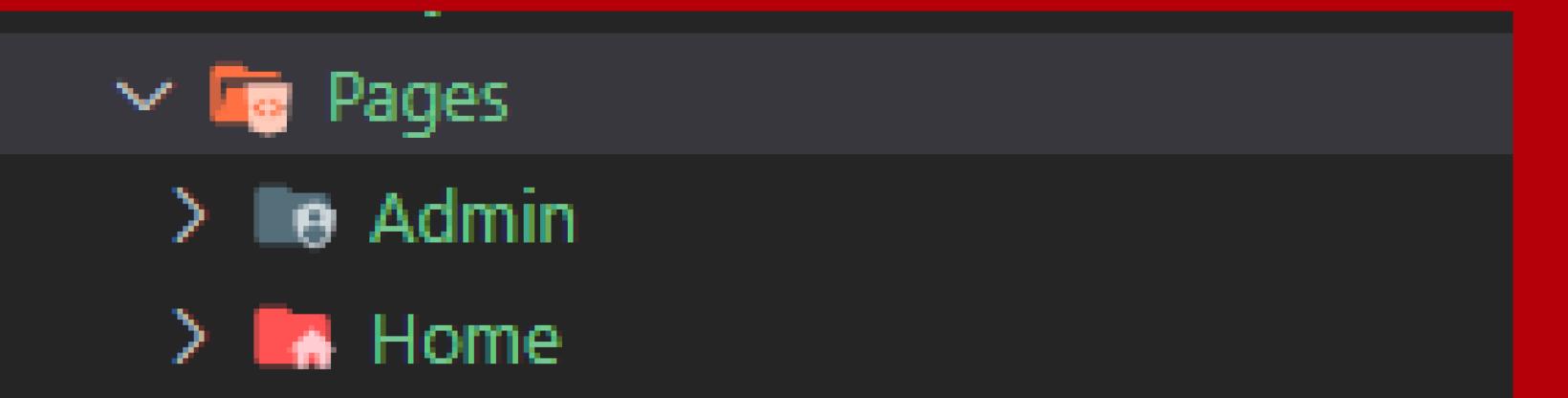
- a) Diretório Footer
- b) Diretório Header
- c) Diretório Register
- d) Arquivo style.css



Passo 5

Crie os sub-diretórios abaixo no diretório Pages:

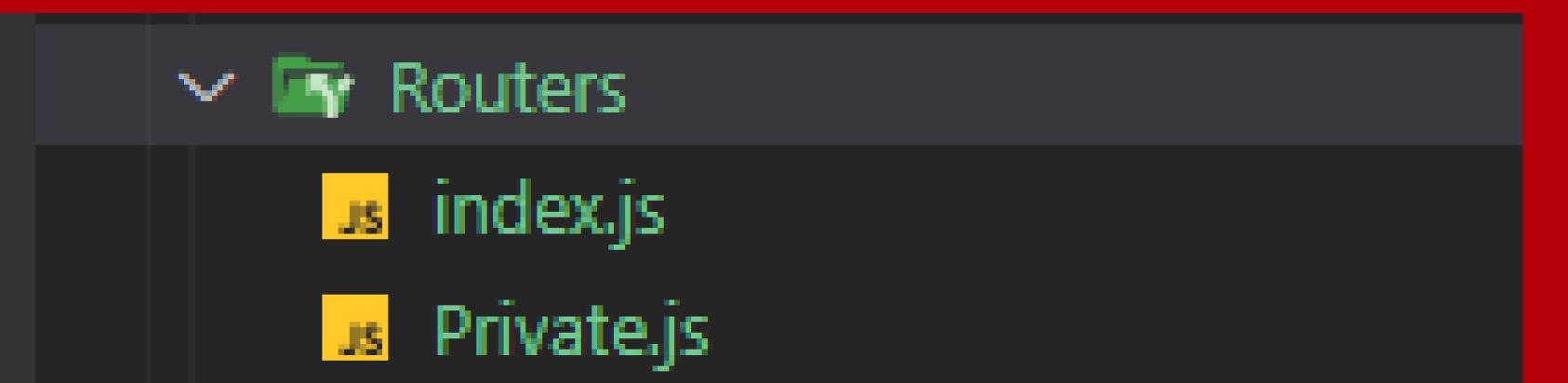
- a) Diretório Admin
- b) Diretório Home



Passo 6

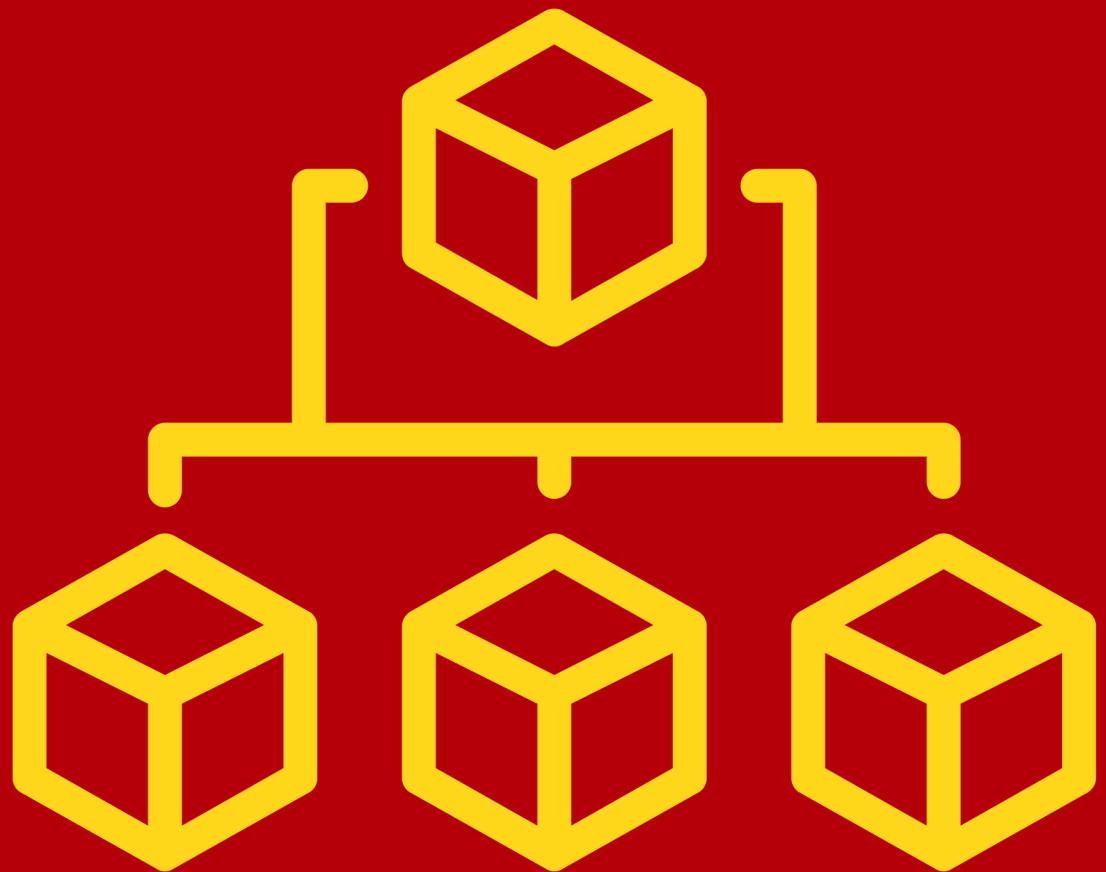
Crie os arquivos abaixo no diretório Routers:

- a) arquivo index.js
- b) arquivo Private.js





Passo 7



Começando pelos componentes

- Header -> cabeçalho com estilização do nav do bootstrap.
- Footer -> rodapé com estilização do bootstrap.
- Register -> componente para registrar novo usuário.
- style.css -> complemento de css3 personalizado.



Arquivo: style.css

```
.bg-pastel-nav {  
    background-color: #cccad4b0;  
}  
  
.bg-pastel-footer {  
    background-color: #cccad4b0;  
}
```

Essa cor representa um tom pastel em tons de cinza e violeta claro.
A transparência permite que elementos subjacentes sejam
parcialmente visíveis.

```
.btn-logout {  
    position: absolute;  
    font-size: 15px;  
    font-weight: bold;  
    font-family: sans-serif;  
    color: #fff;  
    background-color: #be4c4c;  
    bottom: 15%;  
    right: 4%;  
    height: 60px;  
    width: 60px;  
    border-radius: 30px;  
    border: 0;  
    transition: all 0.5s;  
}  
  
.btn-logout:hover {  
    background-color: red;  
    color: #fff;  
}
```

o código define um estilo CSS para um botão de logout com uma classe .btn-logout. Ele tem uma posição absoluta, um fundo vermelho, um tamanho fixo, bordas arredondadas e uma transição suave. Quando o botão é hover, a cor de fundo muda para vermelho e o texto permanece branco. Esse estilo pode ser aplicado a um elemento HTML usando a classe .btn-logout.



Componente Register -> index.js

O próximo código representa um componente React que renderiza um formulário de cadastro de usuário, utiliza o Firebase para criar a conta de usuário e usa o React Router para navegar entre páginas.

Lista de Tarefas

Cadastre-se

Vamos criar sua conta

Seu e-mail:

Sua senha:

Cadastrar

[Já possui uma conta ? Faça o login](#)



Componente Register -> index.js

```
import { useState } from "react";
import { Link, useNavigate } from "react-router-dom";
import { auth } from "../../FirebaseConn";
import { createUserWithEmailAndPassword } from "firebase/auth";
import Header from "../Header";
import Footer from "../Footer";
export default function Register() {
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const navigate = useNavigate();
  async function handleRegister(e) {
    e.preventDefault();
    if (email !== "" && password !== "") {
      await createUserWithEmailAndPassword(auth, email, password)
        .then(() => {
          navigate('/admin', { replace: true });
        })
        .catch((e) => {
          console.log("Aconteceu o erro: " + e.message);
        })
    } else {
      alert('Preencha todos os campos');
    }
  }
}
```



Componente Register -> index.js

```
return (
  <div>
    <Header />
    <span className="mt-5">&nbsp;</span>
    <div className="container">
      <div className="row mt-5">
        <div className="col-2"></div>
        <div className="col-8"><h4 className="text-center">Cadastre-se</h4>
          <span className="text-center"><p>Vamos criar sua conta</p></span>
          <form className="form-control" onSubmit={handleRegister}>
            Seu e-mail:
            <input
              className="form-control mb-3" type="text" placeholder="Digite seu e-mail..." value={email}
              onChange={(e) => setEmail(e.target.value)}
            />
            Sua senha:
            <input
              className="form-control mb-3" autoComplete={"false"} type="password" placeholder="*****"
              value={password} onChange={(e) => setPassword(e.target.value)}
            />
            <button className="btn btn-primary" type="submit">Cadastrar</button>
          </form></div>
        <div className="col-2"></div>
      </div>
    </div>
    <div className="container">
      <div className="row">
        <div className="col-2"></div>
        <div className="col-8">
          <p className="text-center"><Link to="/">Já possui uma conta ? Faça o login</Link></p>
        </div>
        <div className="col-2"></div>
      </div>
    </div>
    <Footer />
  </div>
);
}
```



Header -> index.js

```
import 'bootstrap/dist/css/bootstrap.min.css';
import './style.css';
export default function Header() {
  return (
    <nav className="navbar navbar-expand-lg navbar-light bg-pastel-nav fixed-top">
      <div className="container">
        <p className="navbar-brand text-center w-100">Lista de Tarefas</p>
      </div>
    </nav>
  );
}
```

Em resumo, o código cria um componente de cabeçalho (Header) que representa uma barra de navegação fixa com um título centralizado, estilizado com cores claras.



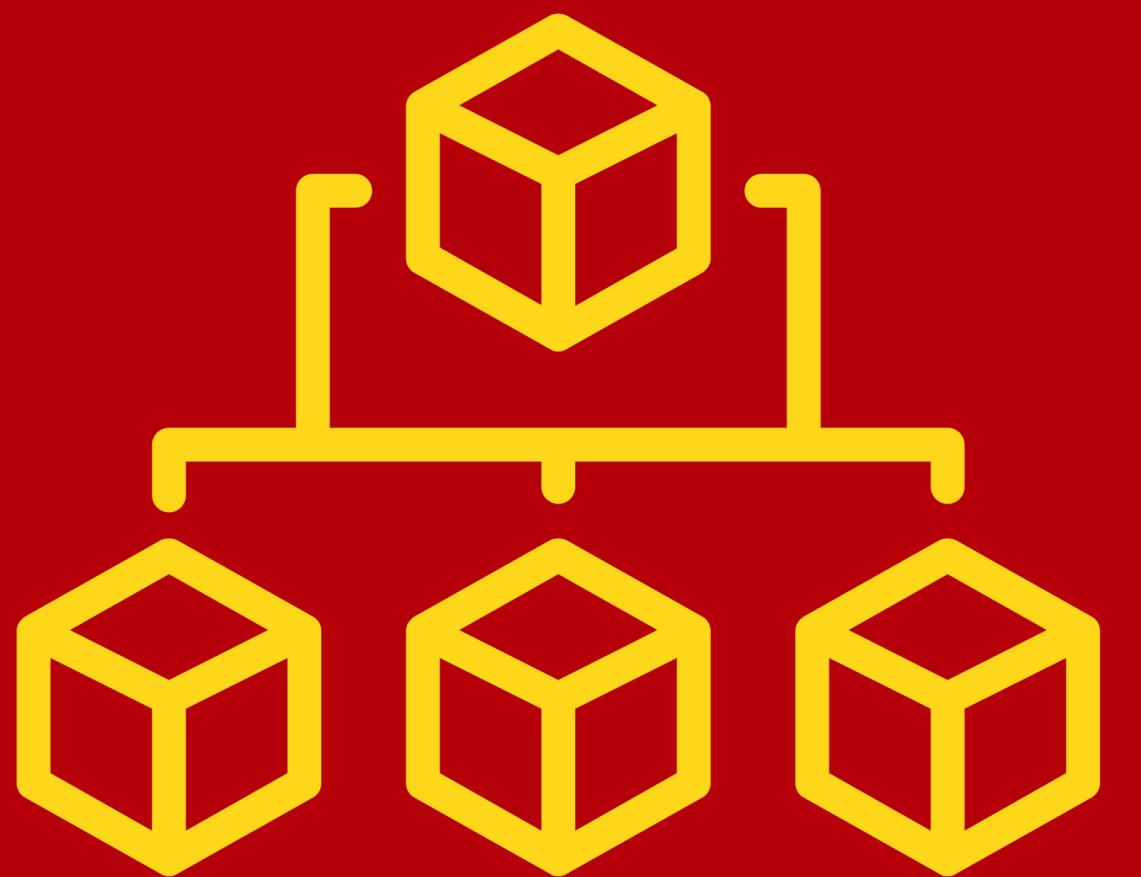
Footer -> index.js

```
import 'bootstrap/dist/css/bootstrap.min.css';
import './style.css';
export default function Footer() {
  return (
    <nav className="navbar navbar-expand-lg navbar-light bg-pastel-footer fixed-bottom">
      <div className="container">
        <p className="navbar-brand text-center w-100">Maromo - Exemplo (To Do)</p>
      </div>
    </nav>
  );
}
```

Em resumo, o código cria um componente (footer) que representa uma barra fixa (rodapé) com um título centralizado, estilizado com cores claras.



Passo 8



Agora o roteamento

- index.js ->
- Private.js->



RoutersApp - roteamento

O componente RoutesApp, a seguir, é definido como uma função e exportado como padrão.

Dentro da função RoutesApp, o componente Routes é utilizado para envolver as rotas da aplicação.

Cada rota é definida usando o componente Route:

- A primeira rota tem o caminho ("/") e o componente Home associado a ela.
- A segunda rota tem o caminho ("/register") e o componente Register associado a ela.
- A terceira rota tem o caminho ("/admin") e o componente Private envolvendo o componente Admin.

O componente Private é usado para proteger a rota "/admin", permitindo o acesso somente se o usuário estiver autenticado.



RoutersApp - roteamento

```
import { Routes, Route } from "react-router-dom";
import Home from "../Pages/Home";
import Register from "../Components/Register";
import Admin from "../Pages/Admin";
import Private from "./Private";

export default function RoutesApp() {
  return (
    <Routes>
      <Route path="/" element={<Home />} />
      <Route path="/register" element={<Register />} />
      <Route path="/admin" element={<Private> <Admin/> </Private>} />
    </Routes>
  );
}
```



Private - componente

O código a seguir representa um componente React chamado Private que protege rotas privadas, verificando se o usuário está autenticado usando o Firebase. Se o usuário estiver autenticado, o componente renderiza o conteúdo das rotas protegidas. Caso contrário, o componente redireciona o usuário para a página inicial.



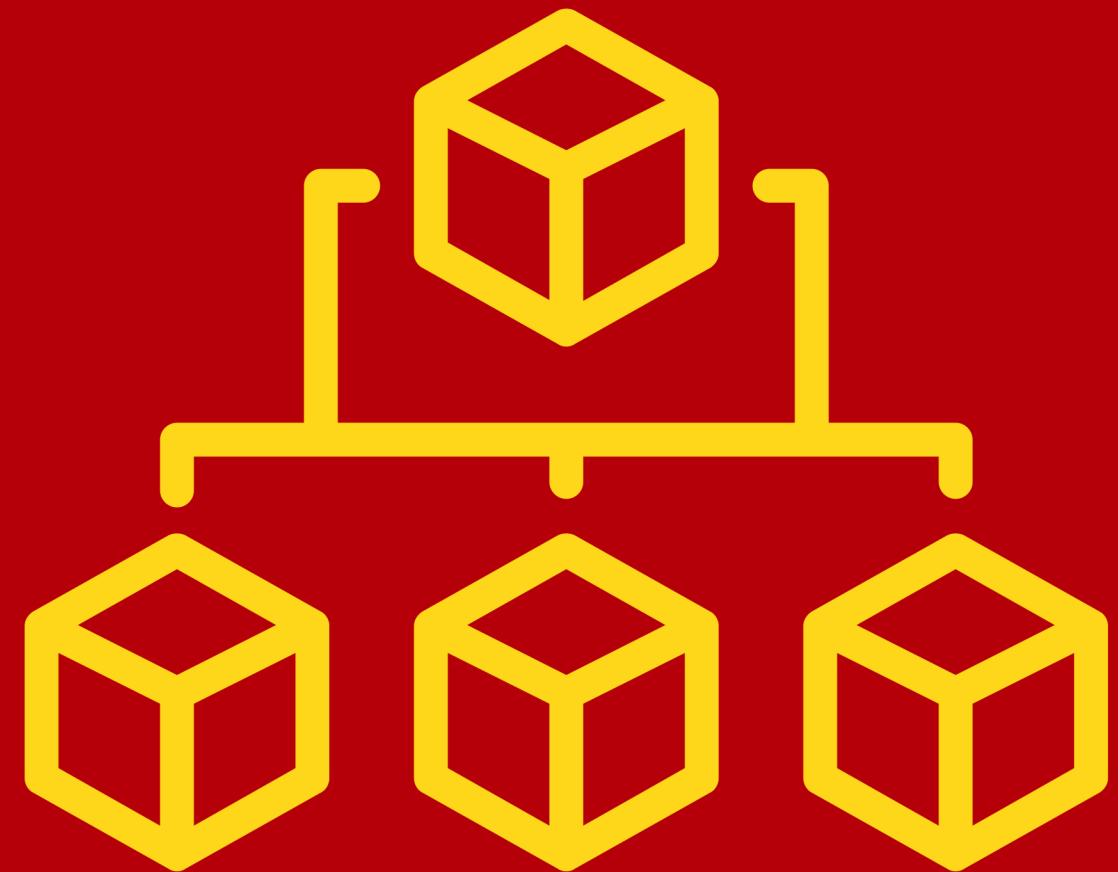
Private - componente

```
import { useState, useEffect } from "react";
import { auth } from "../FirebaseConn";
import { onAuthStateChanged } from "firebase/auth";
import { Navigate } from "react-router-dom";

export default function Private({ children }) {
  const [loading, setLoading] = useState(true);
  const [signed, setSigned] = useState(false);
  useEffect(() => {
    async function checkLogin() {
      onAuthStateChanged(auth, (user) => {
        //verifica se tem usuário logado
        if (user) {
          const userData = {
            uid: user.uid,
            email: user.email,
          };
          localStorage.setItem("@detailsUser", JSON.stringify(userData));
          setLoading(false);
          setSigned(true);
        } else {
          //se nao possui usuário logado
          setLoading(false);
          setSigned(false);
        }
      })
    }
    checkLogin();
  }, []);
  if (loading) { return (<div></div>); }
  if (!signed) { return <Navigate to="/" /> }
  return children;
}
```



Passo 9



Agora as páginas:

Admin -> index.js - a página para administrar as tarefas

Home -> index.js - a página inicial da aplicação

Home - página

```
import { useState } from "react";
import { Link } from "react-router-dom";
import { auth } from "../../FirebaseConn";
import { signInWithEmailAndPassword } from "firebase/auth";
import { useNavigate } from "react-router-dom";
import 'bootstrap/dist/css/bootstrap.min.css';
import Header from "../../Components/Header";
import Footer from "../../Components/Footer";

export default function Home() {
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const navigate = useNavigate();
  async function handleLogin(e) {
    e.preventDefault();
    if (email !== "" && password !== "") {
      await signInWithEmailAndPassword(auth, email, password)
        .then(() => {
          //navegar para /admin
          navigate('/admin', { replace: true })
        })
        .catch((e) => {
          console.log("Erro ao fazer o login: " + e.message);
        })
    } else {
      alert('Preencha todos os campos');
    }
  }
}
```

O código representa a página inicial da aplicação, onde os usuários podem fazer login usando e-mail e senha. Após o login bem-sucedido, eles são redirecionados para a página de administração. O componente também inclui um link para a página de registro e utiliza os componentes Header e Footer para a estrutura da página.



Home - página

```
return (
  <div>
    <Header />
    <div className="container mt-5">
      <div className="row">
        <div className="col"><p className="text-center mt-5">Gerencie sua agenda de forma fácil</p></div>
      </div>
      <div className="row">
        <div className="col-3"></div>
        <div className="col-6">
          <form className="form-control" onSubmit={handleLogin}>
            <label>E-mail</label>
            <input
              className="form-control mb-3" type="text" placeholder="Digite seu e-mail..." value={email}
              onChange={(e) => setEmail(e.target.value)}>
            />
            <label>Senha</label>
            <input
              className="form-control mb-3" autoComplete="false" type="password" placeholder="*****"
              value={password} onChange={(e) => setPassword(e.target.value)}>
            />
            <button className="btn btn-primary form-control mb-5" type="submit">Acessar</button>
          </form>
        </div>
      </div>
      <div className="row">
        <div className="col-3"></div>
        <div className="col-6"><p className="text-center">Não possui conta ? <Link className="" to="/register">Registre-se</Link></p></div>
        <div className="col-3"></div>
      </div>
    </div>
    <Footer />
  </div>
);
}
```

O código a seguir representa a página de administração de tarefas, onde os usuários autenticados podem registrar, editar, excluir e visualizar suas tarefas. O componente utiliza o Firebase Firestore para armazenar os dados das tarefas e os componentes Header e Footer para a estrutura da página.

Admin - página

```
import { useState, useEffect } from "react";
import { auth, db } from "../../FirebaseConn";
import { signOut } from "firebase/auth";
import '../../Components/style.css';
import {
  addDoc, collection, onSnapshot,
  query, orderBy, where,
  doc, deleteDoc, updateDoc
} from "firebase/firestore";
import Header from "../../Components/Header";
import Footer from "../../Components/Footer";

export default function Admin() {
  const [tarefaInput, setTarefaInput] = useState("");
  const [user, setUser] = useState({});
  const [tarefas, setTarefas] = useState([]);
  const [edit, setEdit] = useState({});

  useEffect(() => {
    async function loadTarefas() {
      const userDetail = localStorage.getItem("@detailsUser");
      setUser(JSON.parse(userDetail));
    }
  }, []);

  return (
    <div>
      <h1>Tarefas</h1>
      <input type="text" value={tarefaInput} onChange={(e) => setTarefaInput(e.target.value)} />
      <button onClick={() => handleAddTarefa()}>Adicionar</button>
      <table border="1">
        <thead>
          <tr>
            <th>Descrição</th>
            <th>Data de Criação</th>
            <th>Ações</th>
          </tr>
        </thead>
        <tbody>
          {tarefas.map((tarefa) => (
            <tr>
              <td>{tarefa.tarefa}</td>
              <td>{formatDate(tarefa.createdAt)}</td>
              <td>
                <button onClick={() => handleEditTarefa(tarefa)}>Editar</button>
                <button onClick={() => handleDeleteTarefa(tarefa)}>Excluir</button>
              </td>
            </tr>
          ))}
        </tbody>
      </table>
    </div>
  );
}

function formatDate(date) {
  const options = { year: "numeric", month: "long", day: "numeric" };
  return date.toLocaleDateString("pt-BR", options);
}
```

```
if (userDetail) {
  const data = JSON.parse(userDetail);
  const tarefaRef = collection(db, "tarefas");
  const consulta = query(tarefaRef, orderBy("created", "desc"),
    where("userUid", "==", data?.uid));
  onSnapshot(consulta, (snapshot) => {
    let lista = [];
    snapshot.forEach((doc) => {
      lista.push({
        id: doc.id,
        tarefa: doc.data().tarefa,
        userUid: doc.data().userUid,
      })
    })
    setTarefas(lista);
  })
}

loadTarefas();
}, []);
```



Admin - página

```
async function handleRegister(e) {
  e.preventDefault();
  if (tarefaInput === "") {
    alert('Digite sua tarefa');
    return;
  }
  if (edit?.id) {
    handleUpdateTarefa();
    return;
  }
  await addDoc(collection(db, "tarefas"), {
    tarefa: tarefaInput,
    created: new Date(),
    userUid: user?.uid
  }).then(() => {
    console.log("Tarefa registrada");
    setTarefaInput("");
  }).catch((e) => {
    console.log("Erro ao registrar tarefa" + e.message);
  });
}

async function handleLogout() {
  await signOut(auth);
}

async function taskDelete(id) {
  const docRef = doc(db, "tarefas", id);
  await deleteDoc(docRef);
}

function taskEdit(item) {
  setTarefaInput(item.tarefa);
  setEdit(item);
}

async function handleUpdateTarefa() {
  const docRef = doc(db, "tarefas", edit?.id);
  await updateDoc(docRef, {
    tarefa: tarefaInput
  }).then(() => {
    console.log("tarefa atualizada");
    setTarefaInput("");
    setEdit({});
  }).catch((e) => {
    console.log("erro: " + e.message);
    setTarefaInput("");
    setEdit({});
  });
}
```



Admin - página

```
return (
  <div>
    <Header />
    <div className="mt-5"><span className="mt-5">&nbsp;</span></div>
    <div className="container">
      <div className="row mt-5">
        <div className="col-3"></div>
        <div className="col-6">
          <form className="form-control" onSubmit={handleRegister}>
            <label className="mt-3 mb-2">Sua tarefa</label>
            <textarea
              className="form-control mb-3"
              placeholder="Digite sua tarefa..."
              value={tarefalnput}
              onChange={(e) => setTarefalnput(e.target.value)}
            />
            {Object.keys(edit).length > 0 ?
              <button className="form-control btn btn-primary"
                type="submit">Atualizar Tarefa</button>
            :
              <button className="form-control btn btn-success"
                type="submit">Registrar Tarefa</button>
            }
          </form>
        </div>
        <div className="col-3"></div>
      </div>
```

```
<div className="row mt-5">
  <div className="col-3"></div>
  <div className="col-6"><div className="form-control">
    {tarefas.map((item) => (
      <article key={item.id} className="form-control">
        <p>{item.tarefa}</p>
        <div>
          <button onClick={() => taskEdit(item)} className="btn btn-secondary btn-sm" >Editar</button>&nbsp;
          <button onClick={() => taskDelete(item.id)} className="btn btn-danger btn-sm" >Concluir</button>
        </div>
      </article>
    )));
  </div></div>
  <div className="col-3"></div>
</div>
<button className="btn-logout" onClick={handleLogout}>Sair</button>
<Footer />
</div>
);
}
```



Passo 10

Arquivos finais a serem codificados na raiz do projeto:

- index.js
- index.css
- App.js

index.css

```
*{  
    margin: 0;  
    padding: 0;  
    box-sizing: border-box;  
}  
  
button {  
    cursor: pointer;  
}
```

index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import App from './App';
import './index.css';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
```

App.js

```
import { BrowserRouter } from "react-router-dom";
import RoutesApp from "./Routers";

export default function App() {
  return (
    <BrowserRouter>
      <RoutesApp />
    </BrowserRouter>
  );
}
```

Listar de Tarefas

Gerencie sua agenda de forma fácil

E-mail
Digite seu e-mail...

Senha

Acessar

Não possui conta ? [Registre-se](#)

Maromo Desenvolvimento ®

Listar de Tarefas

Cadastre-se

Vamos criar sua conta

Seu e-mail:

Sua senha:

[Cadastrar](#)

[Já possui uma conta ? Faça o login](#)

Listar de Tarefas

Sua tarefa

Digite sua tarefa...

Registrar Tarefa

Comprar nova bolinha

[Editar](#) [Concluir](#)

Consertar a antena

[Editar](#) [Concluir](#)

Buscar a marmita nova

[Editar](#) [Concluir](#)

Sair

Lista de Tarefas

Sua tarefa

Comprar nova bolinha

Atualizar Tarefa

Comprar nova bolinha

Editar

Concluir



Resumo

- O aplicativo criado possui duas páginas principais: a página inicial, onde os usuários podem fazer login utilizando seu e-mail e senha, e a página de administração, onde os usuários autenticados podem gerenciar suas tarefas.
- **Na página inicial**, os usuários podem inserir suas credenciais de login e fazer login no aplicativo. Se as credenciais estiverem corretas, eles serão redirecionados para a página de administração. Caso contrário, receberão uma mensagem de erro.
- **Na página de administração**, os usuários autenticados podem visualizar suas tarefas registradas anteriormente. Eles têm a opção de adicionar uma nova tarefa, editar tarefas existentes e excluí-las. As tarefas são exibidas em uma lista, onde cada item contém o texto da tarefa e botões para editar e excluir.
- A aplicação utiliza o **Firebase Firestore** para armazenar as tarefas dos usuários. Os dados das tarefas são recuperados e atualizados em tempo real, garantindo uma experiência de usuário atualizada.
- Além disso, o aplicativo possui um design responsivo, adaptando-se a diferentes tamanhos de tela, e utiliza estilos personalizados e a biblioteca Bootstrap para fornecer uma interface visual agradável e intuitiva.

"A programação é a arte de criar algo do nada, usando apenas a imaginação e o poder do código."

