



Universidade Federal de Roraima
Departamento de Ciência da Computação
Arquitetura e tecnologias de sistemas web
Projeto Final

ALUNO(A): Josemar Rocha da Silva.

Matrícula: 2016030185.

ALUNO(A): Markus Kaul Monteiro Gerrits.

Matrícula: 2017024370.

Relatório

Os arquivos do projeto final podem ser acessados no repositório do github:
<https://github.com/JosemarRocha/ARQWEB_projetoFinal_Josemar_Markus>

Com base no que foi implementado até o final da vídeo aula 15(Aula React 15 - Deletando Blog e Page NotFound), o projeto final foi feito então com base na descrição dada pelo professor.

Primeiramente damos início à criação do sistema de autenticação criando as páginas de login e de registro:

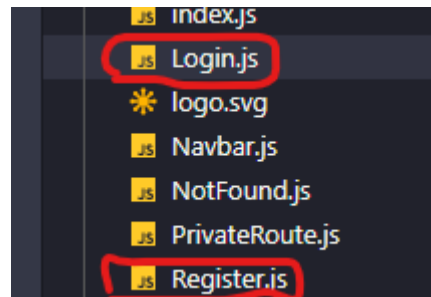


Figura 1. Páginas de login e registro.

Foi então criado um projeto no Firebase para que possa ser feita autenticação via email e senha:

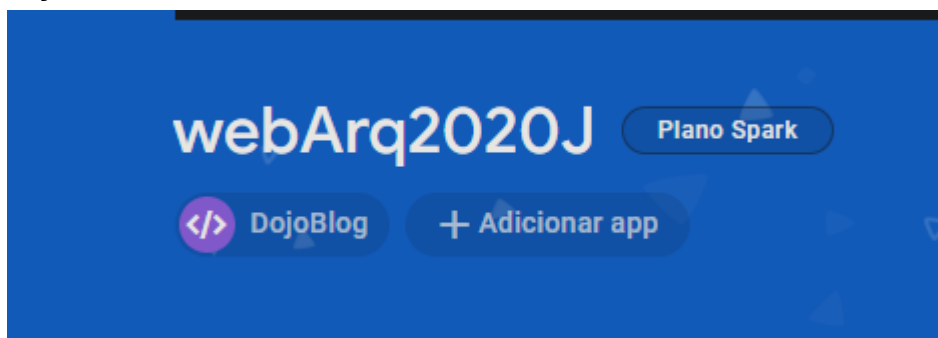


Figura 2. Projeto no firebase.

Com o projeto do firebase criado, podemos pegar a chave de acesso e inseri-la em nosso projeto:

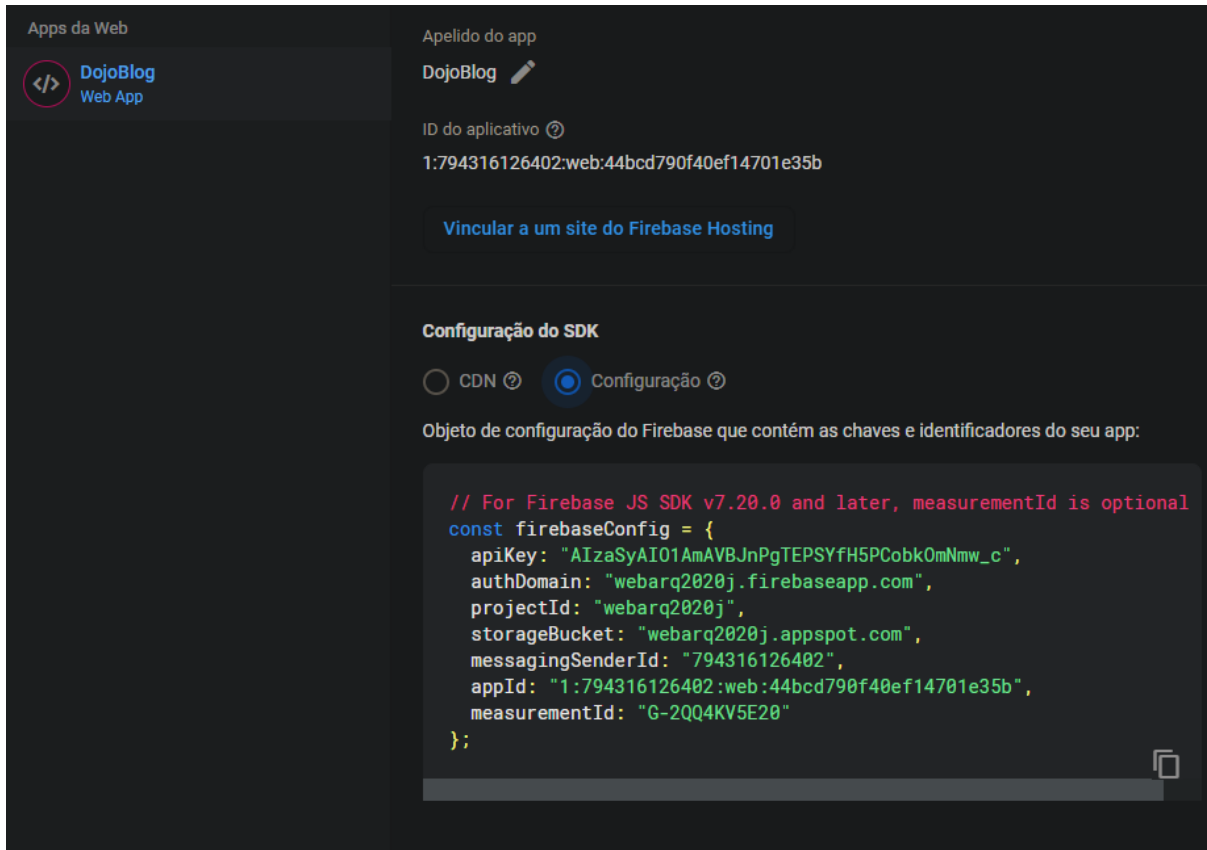


Figura 3. Chave de e identificadores do firebase.

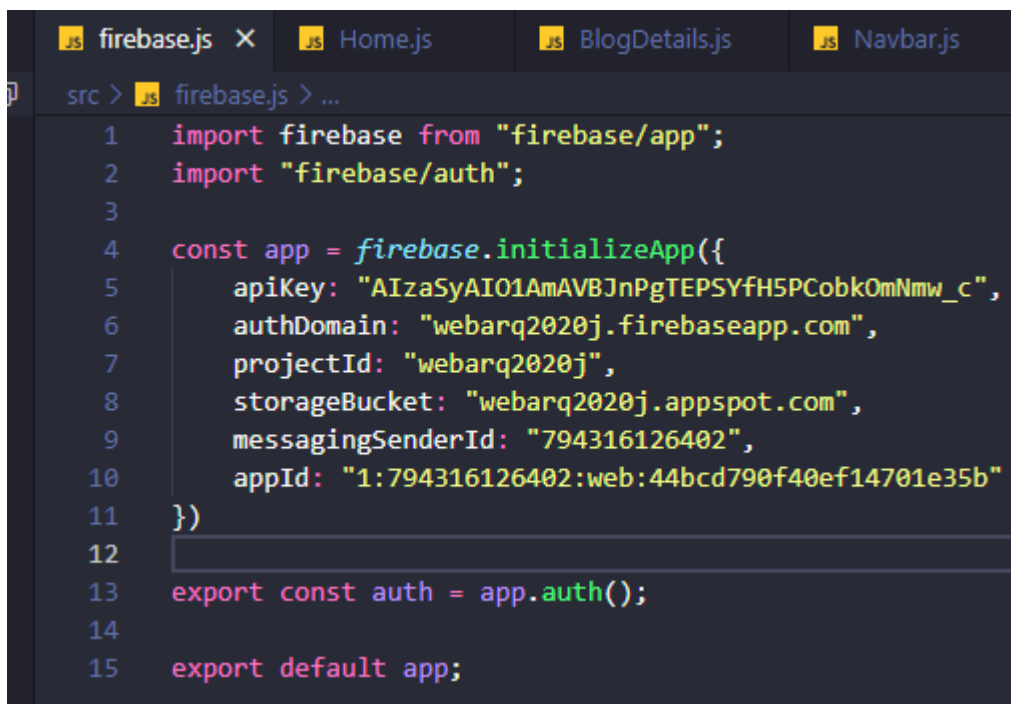


Figura 4. Chaves já no programa.

O nosso projeto então é capaz de utilizar e se conectar ao firebase, nas páginas de login e de registro seremos capazes de utilizar email e senha para autenticação do usuário:


Provedores de login	
Provedor	Status
 E-mail/senha	Ativada

Figura 5. Autenticação via e-mail/senha ativada no firebase.

```
return (  
  <div className="content">  
    <div className="create">  
      <h2>Entrar no Dojo Blog</h2>  
      <form onSubmit={handleSubmit}>  
        <label>E-mail:</label>  
        <input  
          type="text"  
          required  
          value={name}  
          onChange={(e) => setName(e.target.value)}  
        />  
        <label>Password:</label>  
        <input  
          required  
          value={password}  
          onChange={(e) => setPassword(e.target.value)}  
        ></input>  
        { !isPending && <button>Entrar</button> }  
        { isPending && <button disabled>Cagando...</button> }  
        { /*currentUser.email*/ }  
      </form>  
    </div>  
    <div className="signuplink">  
      <p>Não possui uma conta? <Link to="/register">Registre-se </Link></p>  
    </div>  
  </div>  
)
```

Figura 6. Página de login fazendo a verificação por email/senha.

Entrar no Dojo Blog

E-mail:

Password:

Entrar

Não possui uma conta? [Registre-se](#)

Figura 7. Página de login.

Sign Up to Dojo Blog

E-mail:

Password:

Sign Up

Don't have an account? [Log In](#)

Figura 8. Página de registro.

Foi criada a AuthContext.js para ser utilizada em páginas como a de login e registro, para utilizar funções de autenticação e verificação de usuários já presentes ou não no “banco de dados”/json:

```

10 export function AuthProvider({children}){
11     const [currentUser, setCurrentUser] = useState();
12     const [loading, setLoading] = useState(true);
13
14     function register(email, password){
15         return auth.createUserWithEmailAndPassword(email, password);
16     }
17
18     function login(email, password){
19         return auth.signInWithEmailAndPassword(email, password);
20     }
21
22     function logout(){
23         return auth.signOut();
24     }
25
26     useEffect(() => {
27         const unsubscribe = auth.onAuthStateChanged(user =>{
28             setCurrentUser(user);
29             setLoading(false);
30         });
31
32         return unsubscribe;
33     }, [])
34
35
36     const value = {
37         currentUser,
38         login,
39         register,
40         logout
41     }
42
43     return (

```

Figura 9. Parte da AuthContext.js mostrando como seria usada a autenticação nas outras páginas, esta importa a “auth” da firebase.js visto na Figura 4.

Foi feita a implementação de adição de imagens na criação de um blog:

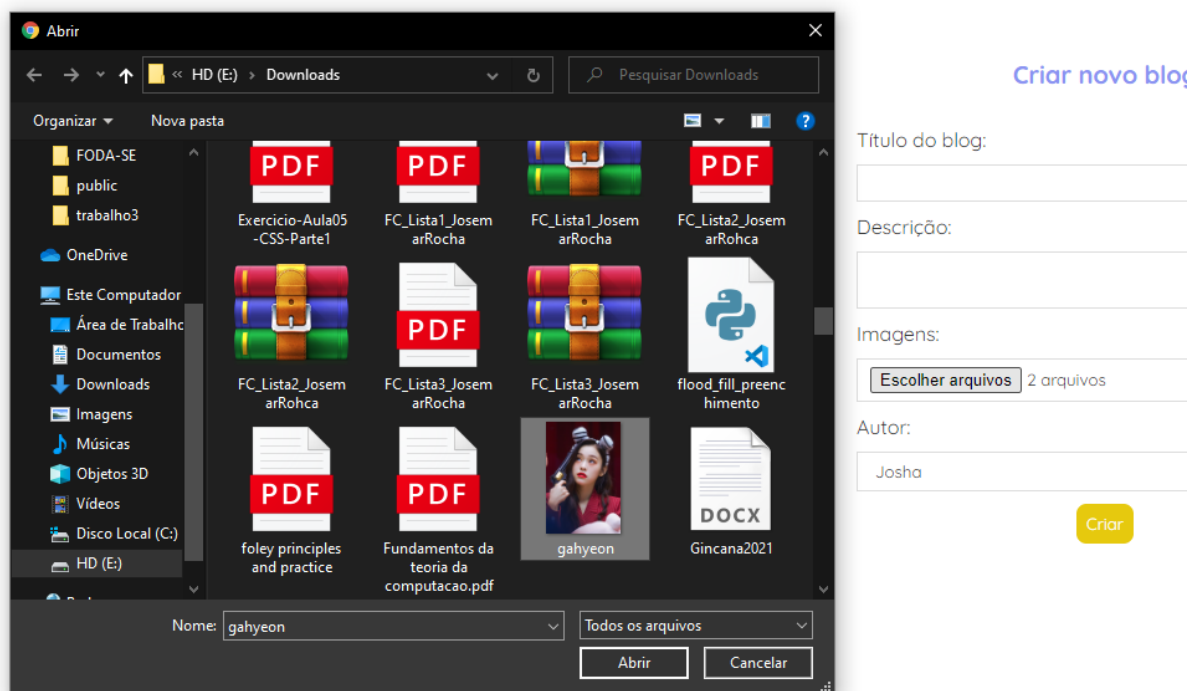


Figura 10. Adição de uma ou múltiplas imagens na criação de um blog.

Foi criado dois botões na página home, um para que seja feito o Log Out do usuário, para que ele saia do sistema e outro para verificação se ele é um admin ou não, caso seja pressionado o botão de “log out” ele será redirecionado para a tela de log in, caso seja pressionado o botão de “verifica” um alerta aparecerá na tela dizendo se aquele usuário é um admin ou não.

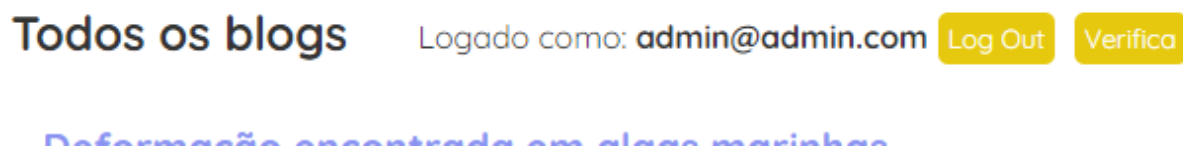
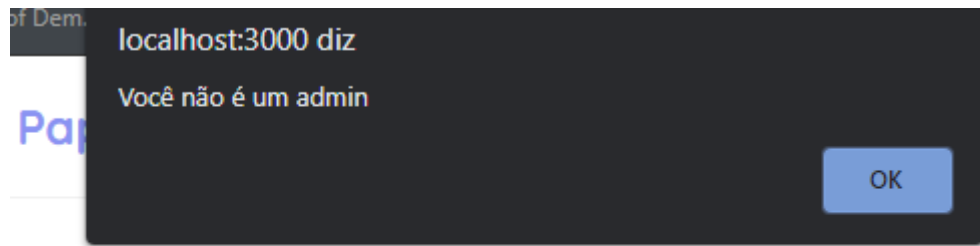


Figura 9. Botões de Log Out e Verificação



Figura 11. Alerta informando que o usuário é admin.



os os blogs

Logado como: joshua@gmail.com [Log Out](#)

Figura 12. Alerta informando que o usuário não é admin.

```
<div className="logoutDiv">
  <label>Logado como: <strong>{currentUser.email}</strong></label>
  <button className="datBtn" onClick={handleLogout}>Log Out</button>
  <button onClick={verifica}>Verifica</button>
</div>
```

Figura 13. Trecho de código da parte informando quem está logado e os respectivos botões vistos acima.

Foi implementada a deleção de blogs que só pode ser feita por admins, caso um usuário não seja admin, um alerta aparece informando isso.

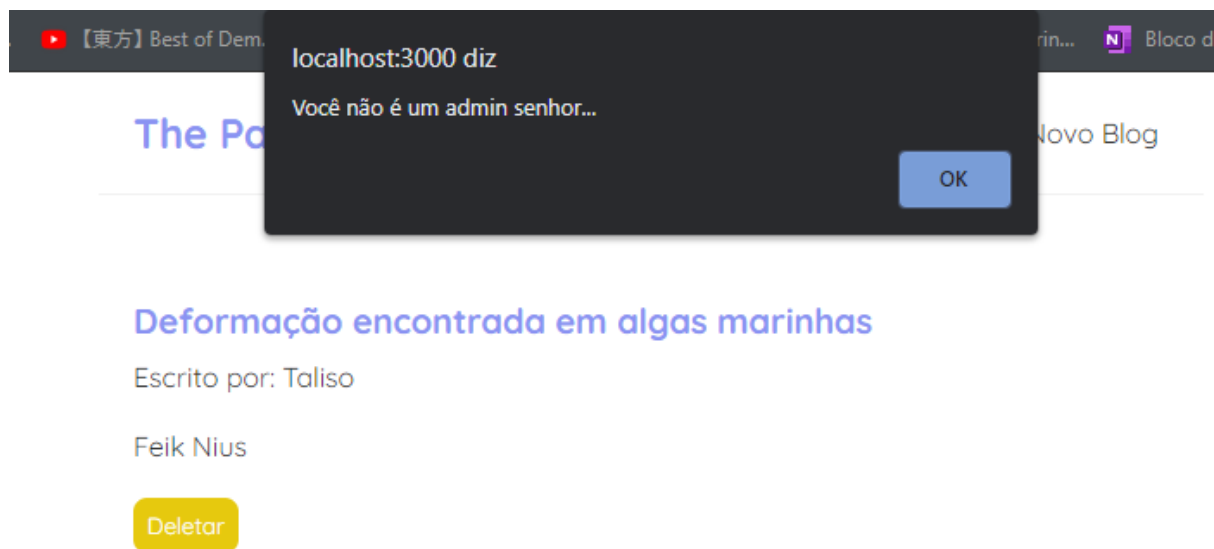


Figura 14. Usuário não admin tentando deletar blog.

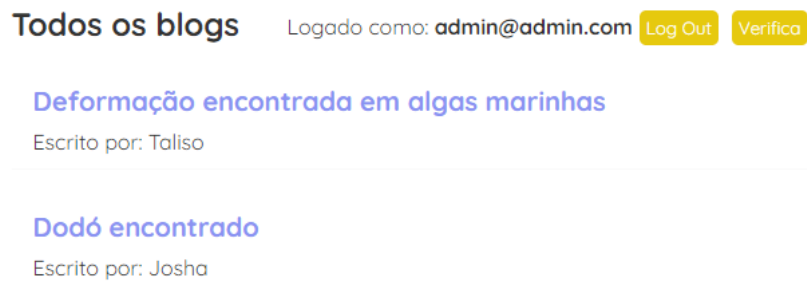


Figura 15. Lista de blogs e Login como admin.

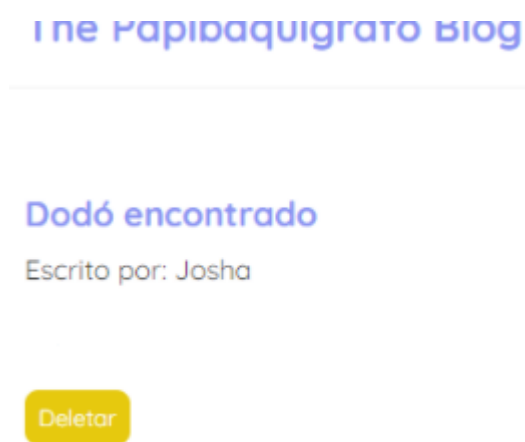


Figura 16. Admin deletando blog.



Figura 17. Lista de blogs após deletar como admin.