

**INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA SUL-RIO-
GRANDENSE - CÂMPUS PASSO FUNDO
CURSO DE TECNOLOGIA EM SISTEMAS PARA INTERNET**

LEONARDO BRUNO DE AGUIAR DE SOUZA

**ESTUDO DO FRAMEWORK IONIC PARA O DESENVOLVIMENTO DE
APLICATIVOS**

Josué Toebe

Rafael Bertei

PASSO FUNDO

2020

LEONARDO BRUNO DE AGUIAR DE SOUZA

**ESTUDO DO FRAMEWORK IONIC PARA O DESENVOLVIMENTO DE
APLICATIVOS**

Monografia apresentada ao Curso de Tecnologia em Sistemas para Internet do Instituto Federal Sul-rio-grandense, Câmpus Passo Fundo, como requisito parcial para a obtenção do título de Tecnólogo em Sistemas para Internet.

Orientador: Josué Toebe

Co-orientador: Rafael Bertei

PASSO FUNDO

2020

LEONARDO BRUNO DE AGUIAR DE SOUZA

**ESTUDO DO FRAMEWORK IONIC PARA O DESENVOLVIMENTO DE
APLICATIVOS**

Trabalho de Conclusão de Curso aprovado em 09/12/2020 como requisito parcial para a obtenção do título de Tecnólogo em Sistemas para Internet

Banca Examinadora:

Josué Toebe

Maikon Cimoski dos Santos

Ricardo Vanni Dallasen

Coordenação do Curso

**PASSO FUNDO
2020**

RESUMO

Com o passar dos anos, os dispositivos móveis têm sido cada vez mais presentes na vida das pessoas. Tarefas que antes só podiam ser realizadas por meio de um computador, podem agora ser realizadas por meio de um smartphone, em qualquer lugar, a qualquer hora. Por esse motivo, o desenvolvimento de aplicações para esses dispositivos, seja uma aplicação web ou nativa, tem aumentado cada vez mais. Este trabalho tem como objetivo utilizar o framework Ionic e desenvolver duas versões de uma aplicação para a criação de notas e lembretes, sendo uma versão desenvolvida com o framework Javascript Angular, e a outra com o framework Javascript React. Ambas as versões da aplicação utilizaram os serviços BaaS disponibilizados pelo Firebase, como o serviço de autenticação, Authentication, e o banco de dados NoSQL Cloud Firestore. Dessa forma, foi possível identificar as características e particularidades de cada framework Javascript utilizado, e apontar qual deles se integra melhor com o Ionic.

Palavras-chave: Ionic, Angular, React.

ABSTRACT

Mobile devices have become more present in people's lives as the years went by. Tasks that were only done with the use of desktop computers now can be done using smartphones, anywhere and at any time. Because of this, the development of applications for those devices, being a web or a native one, has increased significantly. The objective of this work is to utilize the Ionic framework to develop two versions of an application for the creation and management of notes and reminders, one of them using the Javascript framework Angular, and the other one using the Javascript framework React. Both versions utilize the BaaS services offered by Firebase, like the Authentication module and the NoSQL database Cloud Firestore. As a result, it was possible to determine some characteristics and peculiarities from both frameworks utilized, and indicate which one has the best integration with the Ionic Framework.

Keywords: Ionic, Angular, React.

LISTA DE FIGURAS

Figura 1 - Interação entre services, view e components	12
Figura 2 - Diagrama de Classes	14
Figura 3 - Painel do Authentication	15
Figura 4 - Coleções do Cloud Firestore	16
Figura 5 - AuthService	17
Figura 6 - Trecho de código do Componente de Login	18
Figura 7 - Trecho de código do View de Login	19
Figura 8 - Tela de Login da versão Angular	20
Figura 9 - Trecho do NoteService	21
Figura 10 - Código da view do Componente Home Angular	22
Figura 11 - Código de estilo do Componente Home Angular	23
Figura 12 - Código da lógica do Componente Home Angular	23
Figura 13 - Tela Home da versão Angular	25
Figura 14 - Trecho do Componente React LoginButtons	26
Figura 15 - Trecho de código do Componente de Login	27
Figura 16 - Tela Home da versão React	28
Figura 17 - Trecho da consulta de notas	29
Figura 18 - Trecho do código de view do Componente Home React	30
Figura 19 - Código de estilo do Componente Home React	31
Figura 20 - Tela Home da versão React	32

LISTA DE ABREVIATURAS E SIGLAS

API – Application Programming Interface (Interface de Programação de Aplicação)

BaaS – Backend as a Service (Backend como Serviço)

CSS – Cascading Style Sheets (Folha de Estilo em Cascatas)

HTML – Hyper Text Markup Language (Linguagem de Marcação de Hipertexto)

SUMÁRIO

1 INTRODUÇÃO	8
1.1 OBJETIVOS	9
1.1.1 Objetivo Geral	9
1.1.2 Objetivos Específicos	9
1.2 JUSTIFICATIVA	9
2 REFERENCIAL TEÓRICO	10
2.1 FIREBASE	10
2.2 FRAMEWORK IONIC	10
2.3 ANGULAR	11
2.4 REACT	12
3 METODOLOGIA	13
3.1 DESCRIÇÃO DA APLICAÇÃO	13
3.2 DIAGRAMA DE CLASSES	13
4 DESENVOLVIMENTO	14
4.1 AUTENTICAÇÃO COM O FIREBASE	14
4.2 BANCO DE DADOS COM O FIREBASE	15
4.3 VERSÃO ANGULAR	16
4.3.1 Autenticação	16
4.3.2 Banco De Dados	20
4.3.3 Particularidades Da Versão	21
4.4 VERSÃO REACT	25
4.4.1 Autenticação	25
4.4.2 Banco De Dados	28
4.4.3 Particularidades Da Versão	29
4.5 DISCUSSÃO	33
5 CONSIDERAÇÕES FINAIS	34

1 INTRODUÇÃO

A utilização de dispositivos móveis como tablets e smartphones vem crescendo exponencialmente com o passar do tempo. Como apontado pela EXAME (2016), em 2016 cerca de 70% do consumo de internet seria realizado por meio de dispositivos móveis. Em 2017, esse número cresceu para 75%. Ainda segundo a EXAME (2017), cerca de 5 bilhões de pessoas ao redor do mundo possuem dispositivos móveis, como smartphones e tablets.

Consequentemente, o desenvolvimento para web e para dispositivos móveis também teve um crescimento considerável, visando atender as demandas desse crescente mercado. Segundo a CIO (2020), a linguagem Javascript mantém-se como a mais popular entre desenvolvedores no mundo todo. Por isso, tecnologias como frameworks Javascript têm se tornado cada vez mais visadas por desenvolvedores, pois possibilitam a criação de uma aplicação que será funcional tanto em computadores como dispositivos móveis, sem a necessidade de muitos ajustes.

Por meio dos pontos citados, podemos supor que daqui em diante, os dispositivos móveis continuarão fazendo parte do nosso cotidiano, tendo novas ferramentas e aplicações sendo desenvolvidas tanto para serem usadas em navegadores, como rodando de forma nativa. Tendo isso em mente, este trabalho propõe-se em desenvolver uma aplicação com o framework Ionic, em duas versões idênticas, utilizando em uma o framework Javascript Angular, e em outra, framework Javascript React, e apontar qual dos frameworks Javascript integra-se melhor com o framework Ionic. Para isso, na seção 3 são apresentados os detalhes das versões da aplicação que serão desenvolvidas, explicando suas características e funcionalidades. Na seção 4, é apresentado o desenvolvimento das versões, mostrando os diferenciais e particularidades de cada framework Javascript analisado, e como cada framework integra-se com os serviços do Firebase.

1.1 OBJETIVOS

A seguir, serão apresentados os objetivos gerais e específicos do trabalho.

1.1.1 Objetivo Geral

Este trabalho tem como objetivo geral analisar o framework Ionic, bem como as tecnologias Angular e React, que podem ser utilizadas com ele, desenvolvendo uma aplicação web/mobile para a criação de notas e lembretes.

1.1.2 Objetivos Específicos

- Desenvolver com ele um aplicativo com a plataforma Ionic, em duas versões, uma utilizando o framework web Angular, e outra utilizando o framework web React;
- Utilizar os serviços disponibilizados pela ferramenta Firebase, para auxiliar no funcionamento das duas versões da aplicação;
- Apontar as características individuais dos frameworks Javascript analisados, e apontar qual deles é a melhor opção para se utilizar em conjunto com o Ionic.

1.2 JUSTIFICATIVA

Com o crescimento da demanda por aplicativos móveis, o surgimento de tecnologias que facilitem o desenvolvimento deles tem aumentado expressivamente. Uma dessas tecnologias é o framework Ionic, que possibilita o desenvolvimento de aplicações para web ou para smartphones, com agilidade e facilidade. Nas suas primeiras versões, o Ionic só permitia a utilização do framework Javascript Angular para o desenvolvimento. Durante a escrita deste trabalho, além do citado Angular, é possível utilizar também o framework Javascript React em conjunto ao Ionic, além do framework Vue.js e também a utilização de Javascript puro, dispensando o uso de outros frameworks. Desta forma, este trabalho propõe-se a analisar a utilização dos frameworks Javascript Angular e React, em conjunto com o Ionic, e apontar

suas características e peculiaridades.

2 REFERENCIAL TEÓRICO

A seguir, são abordados alguns conceitos de maneira a dar embasamento teórico a esse projeto, tais como Firebase, Ionic, React e Angular.

2.1 FIREBASE

O Firebase é um serviço BaaS (Backend como Serviço) que oferece diversas funcionalidades que auxiliam no desenvolvimento de aplicações tanto para web, como para dispositivos móveis, disponibilizado pela Google. Contém planos de uso pagos mas pode ser utilizado de forma gratuita, com algumas restrições.

Dentre as funcionalidades que o Firebase disponibiliza para seus usuários destacam-se: (FIREBASE, 2020):

- Cloud Firestore: Um banco de dados NoSQL, que permite o armazenamento de dados na forma de coleções e documentos;
- Authentication: Um módulo que abstrai serviços de login e cadastro para o desenvolvedor, possuindo uma integração com diferentes provedores como Facebook, Twitter e Github, bem como os serviços Google, que permite criar um sistema de login e cadastro com poucas linhas de código.
- Hosting: Oferece a possibilidade de hospedagem para sites estáticos, e aplicações, permitindo com que uma aplicação seja rapidamente disponibilizada para acesso, possuindo URL personalizada.

No desenvolvimento de ambas as versões das aplicações deste trabalho, foram utilizados todos os serviços destacados.

2.2 FRAMEWORK IONIC

O framework Ionic foi criado em 2012, época em que soluções para desenvolvimento de aplicações móveis nativas ainda eram muito jovens. Tem como

objetivo tornar o desenvolvimento de aplicações híbridas mais acessível, seguindo as técnicas já empregadas no desenvolvimento de aplicações Web (IONIC, 2020). Estas aplicações, depois de desenvolvidas, podem ser executadas tanto em dispositivos Android ou IOS. Atualmente, o Ionic possui integrações com três grandes frameworks Javascript: Angular, React, e Vue (IONIC, 2020). Esses frameworks possibilitam o desenvolvimento das interfaces, utilizando HTML e CSS, e das partes lógicas, utilizando Javascript.

Um dos principais recursos do Ionic é a completa coleção de elementos de interface disponibilizados pelo framework. Dentre esses elementos destacam-se botões, caixas de texto, modais e alertas. Todos esses elementos possuem uma estilização característica do framework, mas podem ser modificados pelo desenvolvedor utilizando propriedades CSS.

Durante as primeiras três versões do Ionic, o mesmo utilizava no seu funcionamento o framework Angular como padrão. A partir da versão 4, é possível escolher entre os frameworks Javascript Angular, React e Vue, além da possibilidade de se utilizar somente Javascript puro (IONIC, 2020).

2.3 ANGULAR

É um framework Javascript de código aberto, criado e mantido pela Google e pela comunidade, para o desenvolvimento de aplicações web dinâmicas. Ele proporciona uma extensão a sintaxe HTML de um modo que se integra muito facilmente à sua estrutura (ANGULAR, 2020). Em conjunto com Ionic, é primariamente utilizado para compor a interface e gerenciar a lógica e os dados da aplicação. Provê componentes data binding, services, forms, directives, http e dependency injections (DUNKA, OYERINDE E EMMANUEL, 2017).

A arquitetura do Angular é composta por alguns conceitos fundamentais. A principal forma de compor uma aplicação Angular é utilizando componentes.

Componentes operam da seguinte maneira (ANGULAR, 2020):

- Componentes definem as *views* (visões), que basicamente são a parte da interface que o usuário irá ver e interagir, e que pode ser manipulada pela parte lógica da aplicação;
- Componentes usam *services* (serviços), que contém as partes lógicas da aplicação, mas que não precisam necessariamente manipular as

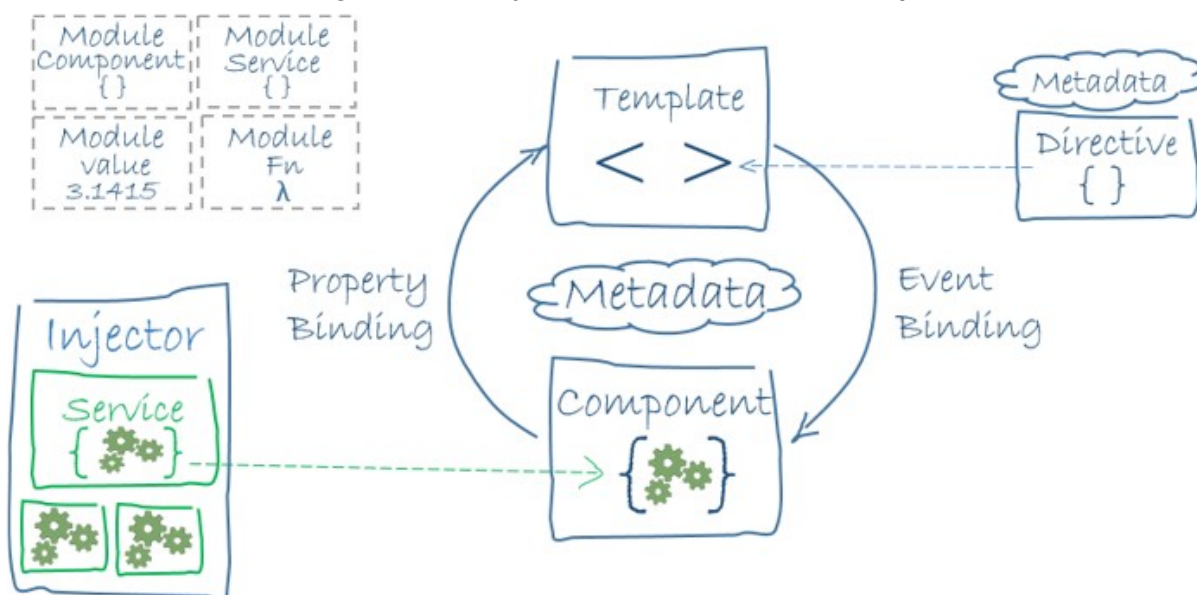
visões. Esses serviços podem ser injetados em componentes como dependências, tornando assim o código modular e reutilizável.

Cada componente Angular consiste de (ANGULAR, 2020):

- Um arquivo HTML que será a visão, contendo o código de interface daquele componente;
- Um arquivo Javascript que contém a lógica do componente, podendo ter serviços injetados nele, permitindo acesso a recursos reutilizáveis da aplicação;
- Um arquivo CSS que conterà a estilização da página.

A Figura 1 demonstra a interação entre os *services* (serviços), a *view* (visão) e os *components* (componentes).

Figura 1 - Interação entre services, view e components



Fonte: ANGULAR, 2020

2.4 REACT

É um framework Javascript de código aberto criado e mantido pelo Facebook, para a criação de aplicações web. Tem como premissa permitir uma forma simples de se construir páginas web com interfaces que terão seus componentes atualizados apenas quando os dados do componente em questão mudarem, sem recarregar a

página inteira (REACT, 2020).

Assim como o Angular, utiliza uma estrutura baseada em componentes, porém sem a restrição de necessitar que o código seja dividido em múltiplos arquivos, por meio do uso da sintaxe JSX, que possibilita o uso do HTML em conjunto ao código Javascript (SAKS, 2019).

3 METODOLOGIA

Para realizar a comparação entre os frameworks Javascript apresentados, optou-se desenvolver uma aplicação web utilizando ambos os frameworks, a ponto de se analisar suas características, similaridades e diferenças na prática. Em conjunto o framework Ionic, foram desenvolvidas duas versões idênticas da aplicação, uma utilizando cada framework Javascript citado.

3.1 DESCRIÇÃO DA APLICAÇÃO

A aplicação possui um sistema de autenticação, permitindo ao usuário a possibilidade de cadastrar-se na aplicação, ou utilizar sua conta do Google ou Facebook para realizar o acesso, dispensando a necessidade do cadastro. Para o cadastro, o usuário deverá prover informações tais como endereço de email, data de nascimento, seu nome e uma senha de no mínimo seis caracteres. Tendo realizado o cadastro, o usuário é automaticamente autenticado, e é direcionado para as demais funcionalidades da aplicação

Uma vez autenticado, o usuário tem acesso a uma página que exibe as notas ou lembretes que o mesmo possui, podendo editá-las ou deletá-las, além de criar novas.

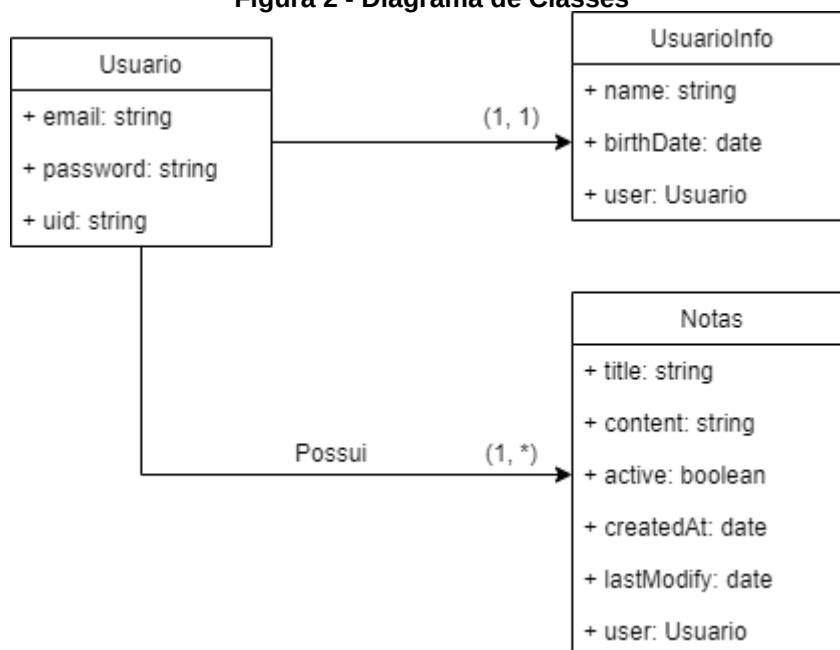
3.2 DIAGRAMA DE CLASSES

Como mostrado na Figura 2, a aplicação é composta pelas seguintes classes:

- **Usuário:** Classe que possui os dados do usuário autenticado, sendo essas informações providas pelo módulo de autenticação do Firebase;
- **UsuárioInfo:** Possui as informações do usuário, quando o mesmo realiza o cadastro pela aplicação.
- **Notas:** Possui as informações das notas que são criadas e editadas pelo

usuário.

Figura 2 - Diagrama de Classes



Fonte: Do autor, 2020

4 DESENVOLVIMENTO

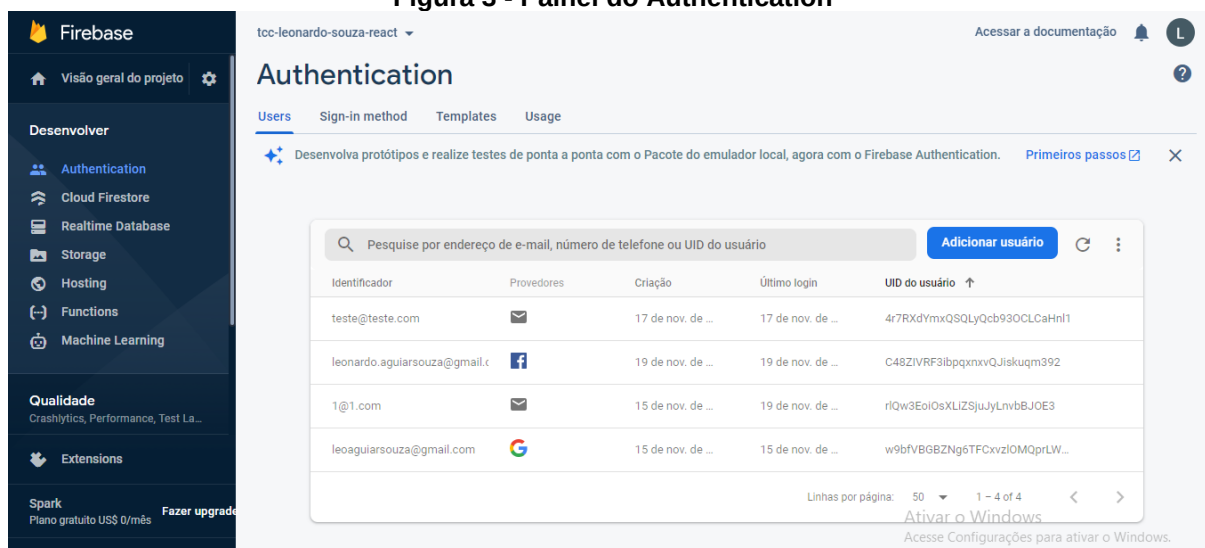
Nessa seção serão mostrados os detalhes do desenvolvimento de ambas as versões da aplicação.

Certos elementos da aplicação são utilizados da mesma maneira em ambas as versões, dentre elas o módulo de autenticação e de banco de dados oferecido pelo Firebase. A seguir, serão descritos esses dois módulos.

4.1 AUTENTICAÇÃO COM O FIREBASE

O Firebase disponibiliza um módulo de autenticação, chamado de Authentication. O Authentication disponibiliza ao desenvolvedor diversas ferramentas e recursos prontos, que gerenciam os usuários cadastrados na aplicação, mostrando seus endereços de email, e plataforma utilizada caso tenham realizado o login por uma plataforma, como o Google ou Facebook. A Figura 3 mostra o painel do Authentication disponibilizado ao desenvolvedor pelo Firebase.

Figura 3 - Painel do Authentication



Fonte: Do autor, 2020

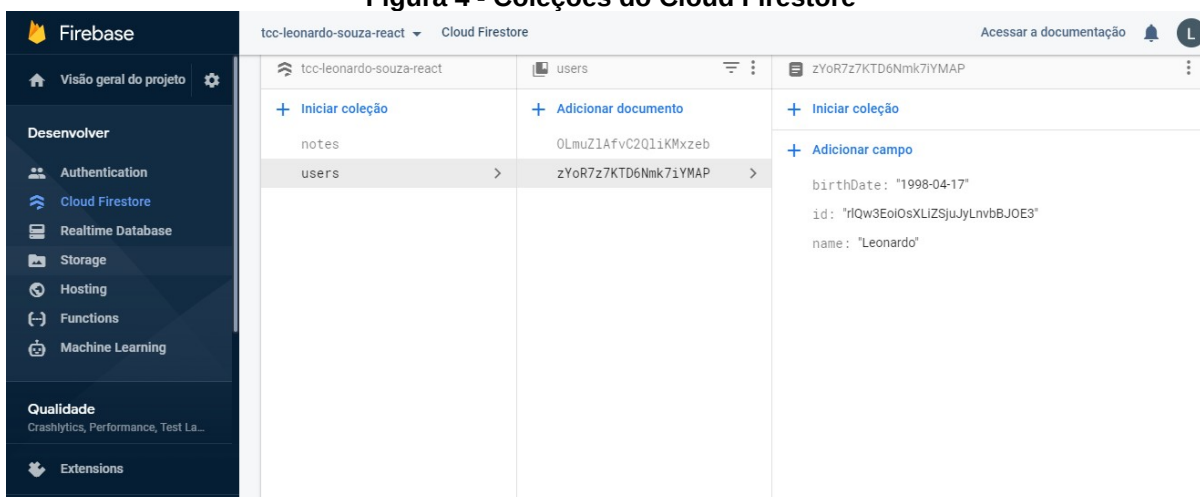
Vale ressaltar que o desenvolvedor tem acesso apenas ao endereço de email do usuário, e ao identificador único gerado pelo Firebase, não tendo acesso a demais informações pessoais privadas, como senhas, por exemplo.

4.2 BANCO DE DADOS COM O FIREBASE

Como solução de banco de dados, foi utilizado o Cloud Firestore, solução NoSQL disponibilizado pelo Firebase, que tem sua organização por meio de coleções, subcoleções e documentos. Para o funcionamento da aplicação, foram utilizadas duas coleções, como mostrado na Figura 4:

- **Users:** Coleção que armazena as informações dos usuários, possuindo identificador único do usuário gerado pelo Firebase, bem como o seu nome e data de nascimento. Essa coleção só tem novos documentos criados quando um usuário realiza o cadastro na aplicação. Caso ele realize login utilizando o Facebook, por exemplo, essas informações são disponibilizadas pelo serviço utilizado no login, dispensando assim a necessidade de se armazenar essas informações.
- **Notes:** Coleção responsável por armazenar os dados das notas. Cada documento corresponde a uma nota, e cada nota possui o identificador único do usuário dono da nota em questão.

Figura 4 - Coleções do Cloud Firestore



Fonte: Do autor, 2020

4.3 VERSÃO ANGULAR

Para utilizar os serviços disponibilizados pelo Firebase com o Angular, foi necessário utilizar a biblioteca AngularFire, biblioteca Angular oficial para acesso aos recursos do Firebase. A seguir, serão apresentados os detalhes do desenvolvimento da versão da aplicação usando o Ionic com Angular.

4.3.1 Autenticação

Para realizar a integração com o Authentication, é necessário utilizar o pacote AngularFireAuth, disponibilizado pela biblioteca AngularFire. Para ser utilizado na aplicação, em conformidade com as boas práticas de desenvolvimento Angular, foi criado um serviço chamado AuthService para gerenciar as interações de cadastro, login e logout. Como mostrado na Figura 5, o AngularFireAuth disponibiliza vários métodos para a realização das ações. Para cadastrar um novo usuário na base de autenticação do Firebase, basta chamar o método **createUserWithEmailAndPassword**, passando como parâmetros o endereço de email e a senha do usuário em questão, e o mesmo será criado na base do Firebase. Para realizar o login de usuário já cadastrado, basta utilizar-se o método **signInWithEmailAndPassword**, passando também como parâmetros o endereço de email e a senha do usuário. Para a utilização do login por meio de outras plataformas, como Google e Facebook, podem ser utilizados os métodos **signInWithPopup**, que irá abrir um popup, e então passar o tipo do provedor como

parâmetro, por exemplo, `new auth.GoogleAuthProvider()` para o login com o serviço do Google, ou `new auth.FacebookAuthProvider()` para o login com o serviço do Facebook.

Figura 5 - AuthService

```
import { Injectable } from '@angular/core';
import { AngularFireAuth } from '@angular/fire/auth';
import { Users } from '../interfaces/users';
import { auth } from 'firebase';

@Injectable({
  providedIn: 'root'
})
export class AuthService {

  constructor(private afa: AngularFireAuth) { }

  login(user: Users) {
    return this.afa.auth.signInWithEmailAndPassword(user.email, user.password);
  }

  loginGoogle() {
    return this.afa.auth.signInWithPopup(new auth.GoogleAuthProvider());
  }

  loginFacebook() {
    return this.afa.auth.signInWithPopup(new auth.FacebookAuthProvider());
  }

  register(user: Users) {
    return this.afa.auth.createUserWithEmailAndPassword(user.email, user.password);
  }

  logout() {
    return this.afa.auth.signOut();
  }

  getAuth() {
    return this.afa.auth;
  }

  getUserId() {
    return this.afa.auth.currentUser.uid;
  }

  getUserEmail() {
    return this.afa.auth.currentUser.email;
  }
}
```

Fonte: Do autor, 2020

Esse serviço é então injetado nos componentes **Register** e **Login**, que gerenciam a interface e funcionalidades das páginas de cadastro e login, respectivamente.

Na Figura 6, é mostrado um trecho do código do componente de login, que implementa o serviço AuthService, e disponibiliza as funções que serão chamadas quando os botões forem clicados na visão, como mostrado na Figura 7.

Figura 6 - Trecho de código do Componente de Login

```
async loginGoogle(){
  try {
    await this.authService.loginGoogle();
  } catch (error) {
    console.log(error);
    this.presentToast(error.message);
  }
}

async loginFacebook(){
  try {
    await this.authService.loginFacebook();
  } catch (error) {
    console.log(error);
    this.presentToast(error.message);
  }
}

async login() {
  await this.presentLoading();

  try {
    await this.authService.login(this.userLogin);
  } catch (error) {
    let message: string;
    console.log(error.code);

    this.presentToast(message);
  } finally {
    this.loading.dismiss();
  }
}
```

Fonte: Do autor, 2020

Figura 7 - Trecho de código do View de Login

```

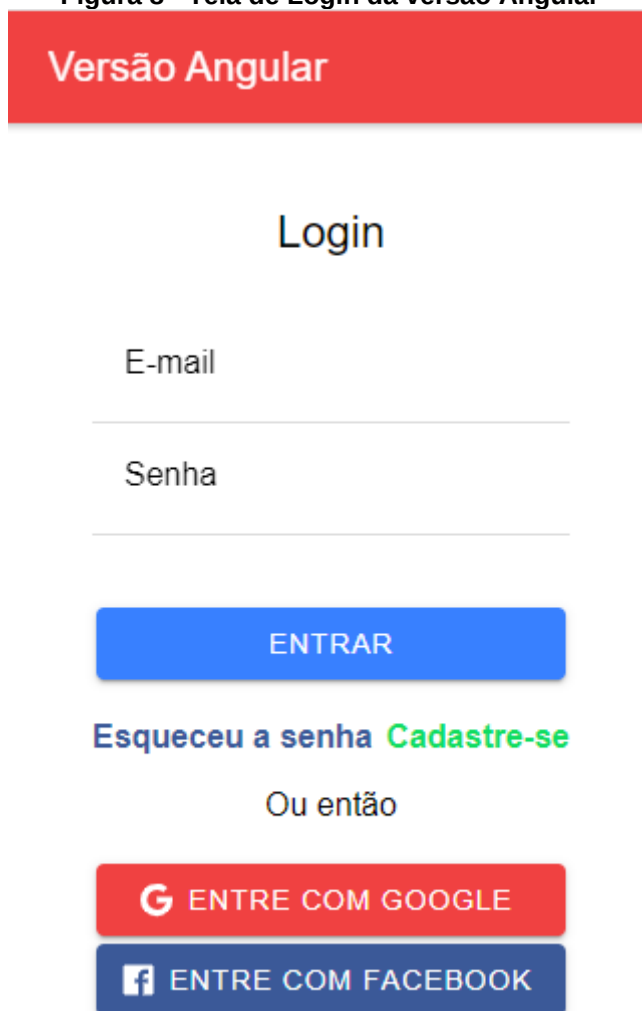
<ion-content padding>
  <ion-grid>
    <ion-row justify-content-center>
      <ion-col align-self-center size-md="6" size-lg="5" size-xs="12">
        <div text-center>
          <h3>Login</h3>
        </div>
        <div padding>
          <ion-item>
            <ion-label position="floating">E-mail</ion-label>
            <ion-input type="email" [(ngModel)]="userLogin.email"></ion-input>
          </ion-item>
          <ion-item>
            <ion-label position="floating">Senha</ion-label>
            <ion-input type="password" [(ngModel)]="userLogin.password"></ion-input>
          </ion-item>
        </div>
        <div padding>
          <ion-button expand="block" (click)="login()">Entrar</ion-button>
        </div>
        <div class="ion-padding-horizontal ion-padding-bottom links">
          <a href="">Esqueceu a senha</a>
          <a routerLink="/register" routerDirection="forward" class="highlighted">Cadastre-se</a>
        </div>
        <div text-center>
          <ion-label>Ou então</ion-label>
        </div>
        <div padding>
          <ion-button color="danger" expand="block" (click)="loginGoogle()">
            <ion-icon slot="start" name="logo-google"></ion-icon>
            Entre com Google
          </ion-button>
          <ion-button color="tertiary" expand="block" (click)="loginFacebook()">
            <ion-icon slot="start" name="logo-facebook"></ion-icon>
            Entre com Facebook
          </ion-button>
        </div>
      </ion-col>
    </ion-row>
  </ion-grid>
</ion-content>

```

Fonte: Do autor, 2020

Na Figura 8 pode ser visualizada a tela de login em funcionamento:

Figura 8 - Tela de Login da versão Angular



Versão Angular

Login


E-mail


Senha

ENTRAR

[Esqueceu a senha](#) [Cadastre-se](#)

Ou então

 ENTRE COM GOOGLE

 ENTRE COM FACEBOOK

Fonte: Do autor, 2020

4.3.2 Banco De Dados

Para realizar a integração com o Cloud Firestore, é necessário utilizar o pacote `AngularFirestore`, disponibilizado pela biblioteca `AngularFire`. Para ser utilizado em conformidade com as boas práticas de desenvolvimento Angular na aplicação, foi criado um serviço chamado `NoteService`, que gerencia as interações da aplicação com o Cloud Firestore. Como demonstrado na Figura 9, o `NoteService` disponibiliza para a aplicação uma série de métodos, que quando implementados nos componentes que mostram as notas ao usuário, interagem com o banco de dados inserindo novos registros, modificando um registro existente, excluindo um

registro específico, e buscando para listagem todas as notas pertencentes ao usuário autenticado, ou buscando uma nota específica para poder ser modificada ou excluída.

Figura 9 - Trecho do NoteService

```

getNotes(user: string): Observable<Note[]> {
  let noteCollection = this.afs.collection<Note>('notes', ref => ref.where('user', '==', user));
  let notes = noteCollection.snapshotChanges().pipe(
    map(actions => {
      return actions.map(a => {
        const data = a.payload.doc.data();
        const lastModifyDate = this.dateConverter(data.lastModify);
        const id = a.payload.doc.id;
        return { id, lastModifyDate, ...data };
      });
    });
  );
  return notes;
}

getNote(id: string): Observable<Note> {
  return this.noteCollection.doc<Note>(id).valueChanges().pipe(
    take(1),
    map(note => {
      note.id = id;
      return note;
    })
  );
}

addNote(note: Note): Promise<DocumentReference> {
  note.user = this.authService.getUserId();
  return this.noteCollection.add(note);
}

updateNote(note: Note): Promise<void> {
  return this.noteCollection.doc(note.id).update(
    {
      title: note.title,
      content: note.content,
      lastModify: note.lastModify,
      active: note.active
    }
  );
}

deleteNote(id: string): Promise<void> {
  return this.noteCollection.doc(id).delete();
}

```

Fonte: Do autor, 2020

4.3.3 Particularidades Da Versão

O componente Home foi criado para gerenciar a tela inicial que o usuário tem acesso após ser autenticado. Nela, são listados todas as notas que pertencem ao usuário em questão. Na versão Angular da aplicação, esse componente, seguindo a estrutura dos componentes Angular, é dividido em três principais arquivos:

- O **home.page.html**, que contém toda a interface da página, escrita em HTML, como mostrado no trecho de código da Figura 10;
- O **home.page.css**, que contém a estilização da página em questão, como mostrado no trecho de código da Figura 11;
- E o **home.page.ts**, que possui toda a lógica referente a esse componente, como mostrado no trecho de código da Figura 12.

Figura 10 - Código da view do Componente Home Angular

```

<ion-header>
  <ion-toolbar color="danger">
    <ion-buttons slot="start">
      <ion-menu-button autoHide="false"></ion-menu-button>
    </ion-buttons>
    <ion-title text-center>Versão Angular</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content padding>
  <ion-fab vertical="bottom" horizontal="end" slot="fixed">
    <ion-fab-button routerLink="/note" color="success">
      <ion-icon name="add"></ion-icon>
    </ion-fab-button>
  </ion-fab>

  <ion-list *ngIf="notes">
    <ion-item [class]="note.active ? 'note-active' : 'note-inactive'"
      button [routerLink]="['/note', note.id]"
      *ngFor="let note of (notes | async)">
      <ion-label>
        <h3>{{ note.title }}</h3>
        <p>Última modificação em:</p>
        <p>&mdash;&nbsp;&nbsp;&nbsp;{{ note.lastModifyDate }}</p>
      </ion-label>
    </ion-item>
  </ion-list>

  <ion-grid>
    <ion-row justify-content-center>
      <ion-col align-self-center size-md="6" size-lg="5" size-xs="12">

        </ion-col>
      </ion-row>
    </ion-grid>
  </ion-content>

```

Figura 11 - Código de estilo do Componente Home Angular

```
ion-item.note-active ion-label {
  border-left: 2px solid var(--ion-color-success);
  padding-left: 10px;
}

ion-item.note-inactive ion-label {
  border-left: 2px solid var(--ion-color-danger);
  padding-left: 10px;
}
```

Fonte: Do autor, 2020

Figura 12 - Código da lógica do Componente Home Angular

```
import { Component, OnInit } from '@angular/core';
import { Observable } from 'rxjs';
import { AuthService } from 'src/app/services/auth.service';
import { Note } from 'src/app/interfaces/note';
import { NoteService } from 'src/app/services/notes.service';
import { Events } from '@ionic/angular';
import { Users } from 'src/app/interfaces/users';

@Component({
  selector: 'app-home',
  templateUrl: './home.page.html',
  styleUrls: ['./home.page.scss'],
})
export class HomePage implements OnInit {

  public notes: Observable<Note[]>
  private user: string;
  public userObj: Observable<Users[]>;

  constructor(
    public events: Events,
    private noteService: NoteService,
    private authService: AuthService
  ) { }

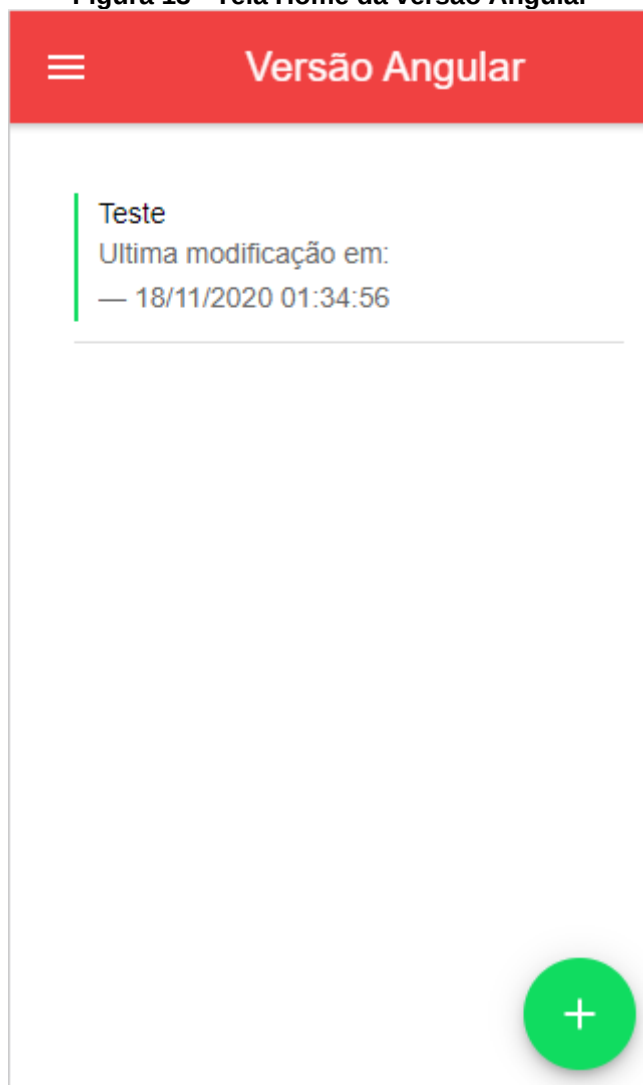
  ngOnInit() {
  }

  ionViewWillEnter() {
    this.user = this.authService.getUserId();
    this.notes = this.noteService.getNotes(this.user);
  }
}
```

Fonte: Do autor, 2020

Como mostrado na Figura 12, usando o método do Ionic **ionViewWillEnter**, que é chamado antes da página ser carregada, o identificador único do usuário autenticado é buscado utilizando o método **getUserId** definido no service AuthService, e após isso o passando esse id como parâmetro no método **getNotes**, do service NoteService, as notas são armazenadas no objeto notes. Esse objeto pode ser acessado pela visão, como mostrado na Figura 8, utilizando algumas diretrizes disponibilizadas pelo Angular, como a ***ngIf** e ***ngFor**, que permitem a renderização dinâmica dos dados na view. Nesse caso, o bloco de código de listagem das notas na visão só é carregado caso o usuário possua notas cadastradas, utilizando o ***ngIf**. Utilizando a diretriz ***ngFor**, o elemento que compõe a representação da nota é repetido quantas vezes for necessário para mostrar todas as notas pertencentes ao usuário. A Figura 13 mostra o resultado em funcionamento desta página.

Figura 13 - Tela Home da versão Angular



Fonte: Do autor, 2020

4.4 VERSÃO REACT

Para a utilização dos serviços disponibilizados pelo Firebase utilizando o Ionic com o framework Javascript React, não existe necessidade de se utilizar uma biblioteca como AngularFire. O acesso aos recursos pode ser feito diretamente pela biblioteca Javascript Firebase, disponibilizada pelo Firebase.

4.4.1 Autenticação

Para realizar a integração com o Authentication com o React, basta referenciar o objeto **auth**, disponibilizado pela biblioteca Firebase, chamando então

seus métodos, como listados anteriormente. Esses métodos, diferentemente da utilização no Angular, podem ser chamados diretamente na parte lógica do componente, já que o React normalmente não utiliza injeção de serviços nos componentes como o Angular. A Figura 14 mostra a utilização do método para autenticar usuário cadastrado na aplicação via email e senha, diretamente na lógica do componente, em conjunto com as demais ações daquele componente.

Figura 14 - Trecho do Componente React LoginButtons

```
const LoginButtons: React.FC = () => {
  let history = useHistory();

  function GoToHome() {
    history.push("/home");
  }

  async function userLoginGoogle() {
    try {
      await firebase.auth().signInWithPopup(new firebase.auth.GoogleAuthProvider()).then(() => {
        GoToHome();
      })
    } catch(error) {
      console.log(">> " + error)
    }
  }

  async function userLoginFacebook() {
    try {
      await firebase.auth().signInWithPopup(new firebase.auth.FacebookAuthProvider()).then(() => {
        GoToHome();
      })
    } catch(error) {
      console.log(">> " + error)
    }
  }
}
```

Fonte: Do autor, 2020

Esse componente pode então ser utilizado no componente que gerencia a página de login, como mostra a Figura 15.

Figura 15 - Trecho de código do Componente de Login

```

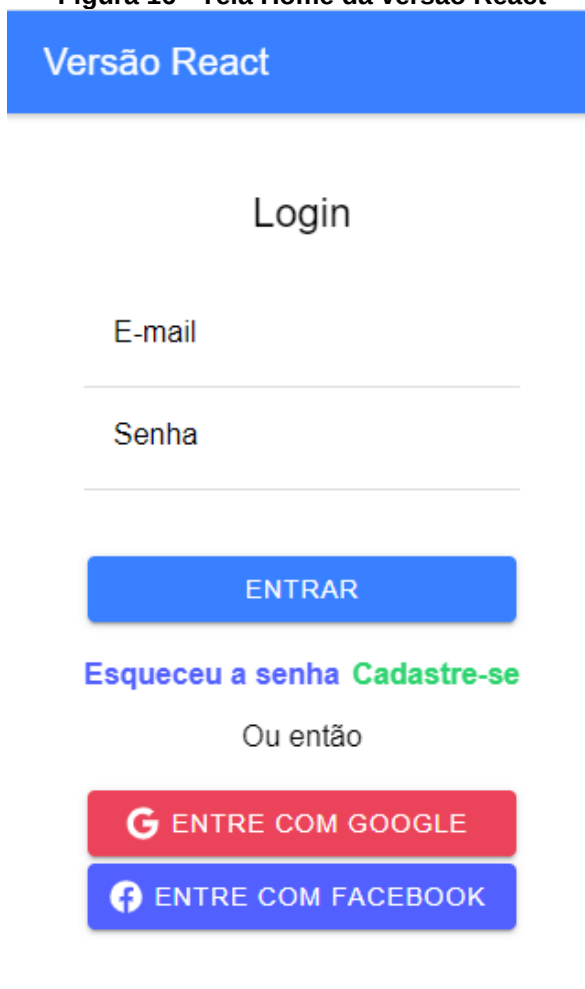
<IonContent className="ion-padding">
  <IonGrid>
    <IonRow className="ion-justify-content-center">
      <IonCol className="ion-align-self-center" size-md="6" size-lg="5" size-xs="12">
        <div className="ion-text-center">
          <h3>Login</h3>
        </div>
        <div className="ion-padding">
          <IonItem>
            <IonLabel position="floating">E-mail</IonLabel>
            <IonInput value={email} onChange={(e: any) => setEmail(e.target.value)}
              type="email" id="email" required>
          </IonInput>
        </IonItem>
        <IonItem>
            <IonLabel position="floating">Senha</IonLabel>
            <IonInput value={password} onChange={(e: any) => setPassword(e.target.value)}
              type="password" id="password" required>
          </IonInput>
        </IonItem>
      </div>
      <div className="ion-padding">
        <IonButton expand="block" onClick={userLogin}>Entrar</IonButton>
      </div>
      <div className="ion-padding-horizontal ion-padding-bottom links">
        <IonRouterLink routerLink="/login" className="link">Esqueceu a senha</IonRouterLink>
        <IonRouterLink routerLink="/register" routerDirection="forward"
          className="link highlighted">
          Cadastre-se
        </IonRouterLink>
      </div>
      <div className="ion-text-center">
        <IonLabel>Ou então</IonLabel>
      </div>
      <LoginButtons />
    </IonCol>
  </IonRow>
</IonGrid>
</IonContent>

```

Fonte: Do autor, 2020

O resultado dessa tela pode ser visto na Figura 16.

Figura 16 - Tela Home da versão React



A tela de login da versão React apresenta um cabeçalho azul com o texto "Versão React". Abaixo, o título "Login" é centralizado. Há dois campos de entrada para "E-mail" e "Senha", cada um com uma linha de separação. Um botão azul "ENTRAR" está centralizado. Abaixo dele, há links para "Esqueceu a senha" (em azul) e "Cadastre-se" (em verde). O texto "Ou então" segue. Duas opções de login social são oferecidas: "ENTRE COM GOOGLE" em um botão vermelho e "ENTRE COM FACEBOOK" em um botão azul.

Fonte: Do autor, 2020

4.4.2 Banco De Dados

Para a integração do React com o Cloud Firestore, basta referenciar o objeto **firestore**, disponibilizado pela biblioteca Firebase. Assim, com poucas linhas de código, é possível utilizar os métodos disponibilizados para fazer consultas. Como é mostrado na Figura 17, o trecho de código em questão realiza a consulta dos dados no Cloud Firestore, e trata esses dados para serem mostrados na visão.

Figura 17 - Trecho da consulta de notas

```

const fetchData = async() => {
  const loading = createLoading();
  (await loading).present();
  try {
    const response = await firebase.firestore()
      .collection("notes")
      .where("user", "==", userId)
      .get();

    response.forEach(element => {
      let path = "/note/" + element.id;
      let title = element.data()?.title;
      let active = element.data()?.active;
      let createdAt = dateConverter(element.data()?.createdAt);
      let lastModify = dateConverter(element.data()?.lastModify);

      array.push([path, title, active, createdAt, lastModify])
    })
  } catch(err) {
    console.error(err);
  } finally {
    (await loading).dismiss();
  }
};

```

Fonte: Do autor, 2020

4.4.3 Particularidades Da Versão

Assim como na versão Angular, o componente Home foi criado para gerenciar a tela inicial onde são listados as notas que pertencentes ao usuário autenticado. Diferentemente da versão Angular da aplicação, esse componente pode ser criado em apenas dois arquivos:

- O **Home.tsx**, que contém a lógica da aplicação, como demonstrado na Figura 17, bem como a definição da visão a ser renderizada, como demonstrado na Figura 18;
- O **Home.css**, que contém a estilização da página, como demonstrado na Figura 19.

Figura 18 - Trecho do código de view do Componente Home React

```

return (
  <IonPage>
    <IonHeader>
      <IonToolbar color="primary">
        <IonButtons slot="start">
          <IonMenuButton autoHide={false}></IonMenuButton>
        </IonButtons>
        <IonTitle text-center>Versão React</IonTitle>
      </IonToolbar>
    </IonHeader>

    <IonContent className="ion-padding">
      <IonFab vertical="bottom" horizontal="end" slot="fixed">
        <IonFabButton routerLink="/note" color="secondary">
          <IonIcon icon={add}></IonIcon>
        </IonFabButton>
      </IonFab>

      <IonList>
        {
          itemList.map(
            ( _item, _index ) => (
              <IonItem className={_item[2] ? "note-active" : "note-inactive"}
                key={_index} button routerLink={_item[0]}>
                <IonLabel>
                  <h3>{_item[1]}</h3>
                  <p>Ultima modificação em:</p>
                  <p>&mdash;&nbsp; {_item[4]}</p>
                </IonLabel>
              </IonItem>
            ) )
        }
      </IonList>

      <IonGrid>
        <IonRow justify-content-center>
          <IonCol align-self-center size-md="6" size-lg="5" size-xs="12">

            </IonCol>
          </IonRow>
        </IonGrid>
      </IonContent>
    </IonPage>
  );
};

```

Fonte: Do autor, 2020

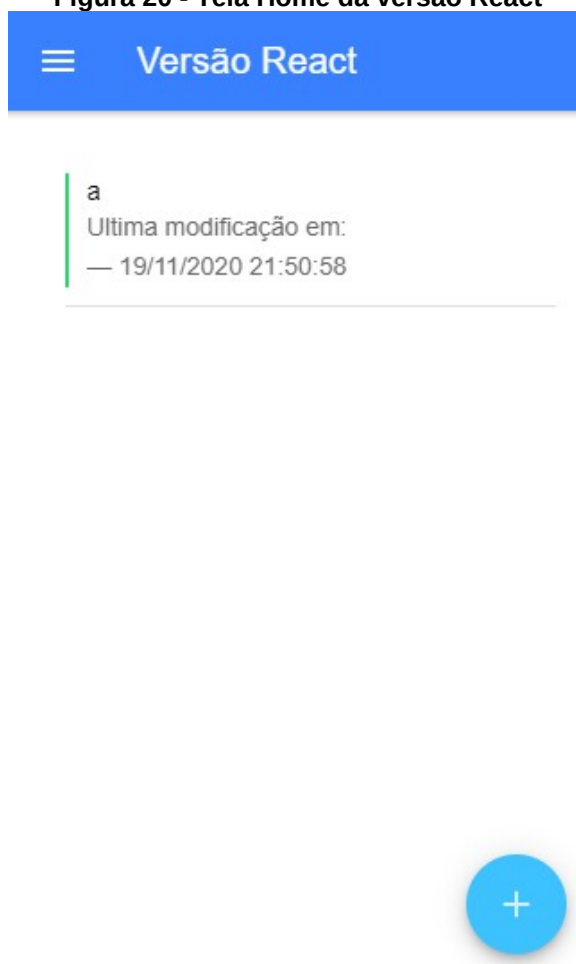
Figura 19 - Código de estilo do Componente Home React

```
ion-item.note-active ion-label {  
  border-left: 2px solid var(--ion-color-success);  
  padding-left: 10px;  
}  
  
ion-item.note-inactive ion-label {  
  border-left: 2px solid var(--ion-color-danger);  
  padding-left: 10px;  
}
```

Fonte: Do autor, 2020

Como demonstrado na Figura 17, a função **fetchData** é chamada caso o usuário esteja logado, e essa função acessa o Cloud Firestore buscando todas as notas referentes ao usuário autenticado. Em seguida essas informações são tratadas e formatadas, para então serem armazenadas em um array, que é utilizado para mostrar os dados na visão. Na renderização da visão, mostrada na Figura 18, é necessário realizar um tratamento na array que contém os dados, para listar todas as notas que usuário possui, junto com suas informações. Esse processo apesar de simples, é totalmente desnecessário na versão Angular, por conta da diretriz de renderização condicional ***ngFor**. A Figura 20 mostra o resultado final desta tela.

Figura 20 - Tela Home da versão React



Fonte: Do autor, 2020

4.5 DISCUSSÃO

Ambos os frameworks analisados funcionam de forma muito satisfatória com o Ionic. Um desenvolvedor familiarizado com a linguagem Javascript tem potencial de desenvolver uma aplicação utilizando Ionic sem muitas dificuldades. O framework possui comunidade ativa em fóruns de desenvolvedores, e a documentação disponibilizada é completa, contando com guias e tutoriais para a utilização de todos os frameworks Javascript suportados, além do uso do Javascript puro. Dentre os frameworks analisados, o Angular possui uma integração mais natural com o Ionic, por ter sido o framework padrão durante seus primeiros anos.

Na criação de um novo projeto Ionic, o mesmo já traz algumas ferramentas de linha de comando do Angular, que possibilitam a criação de módulos, serviços e outros elementos via linha de comando, o que automatiza boa parte do processo de desenvolvimento, enquanto no React essas estruturas devem ser criadas manualmente. Porém, isso não significa que o React seja uma má escolha, vide que o mesmo pode ser utilizado como porta de entrada ao desenvolvimento com React, podendo o conhecimento adquirido ser transferido para o uso do React independentemente do Ionic, por meio do framework React Native.

Então, pode-se determinar que o Angular é o framework que melhor integra-se com o Ionic, oferecendo diversas ferramentas e estruturas ao desenvolvedor logo no início do desenvolvimento da aplicação. Sua estrutura de arquivos proporciona uma organização de código eficiente, facilitando os ajustes nos componentes quando necessário.

5 CONSIDERAÇÕES FINAIS

O framework Ionic demonstrou-se eficaz, e não apresentou dificuldades na sua usabilidade durante o desenvolvimento. Foi possível obter um resultado bem próximo em cada versão da aplicação, sendo elas praticamente idênticas no visual e funcionalidades propostas. As maiores diferenças se dão na parte do código de cada uma delas, pelo fato de cada framework possuir suas particularidades, porém ambos os frameworks tendo como base o Javascript, acabam tendo uma curva de aprendizado semelhante. O uso dos recursos oferecidos pelo Firebase facilitaram o desenvolvimento do trabalho, por disponibilizarem uma estrutura de backend eficiente e simples de ser utilizada.

Para trabalhos futuros, pode-se acrescentar mais funcionalidades na aplicação proposta, e principalmente analisar também a integração do framework Javascript Vue.js com o Ionic, bem como a utilização de Javascript puro, dispensando o uso de frameworks.

6 REFERÊNCIAS

ANGULAR. *Introduction to the Angular Docs*. 2020. Disponível em: <<https://angular.io/docs>>. Acesso em: 20 out. 2020.

CIO. *JavaScript é a linguagem mais popular da atualidade e domina aplicativos web e na nuvem*. 2020. Disponível em: <<https://cio.com.br/carreira/javascript-e-a-linguagem-mais-popular-da-atualidade-e-domina-aplicativos-web-e-na-nuvem/>>. Acesso em: 27 out. 2020.

DUNKA, Bakwa; OYERINDE, Dantala Oyeyinka; EMMANUEL, Edim A. *Hybrid Mobile Application Based On Ionic Framework Technologies*. 2017. Disponível em: <https://www.researchgate.net/publication/322397904_HYBRID_MOBILE_APPLICATION_BASED_ON_IONIC_FRAMEWORK_TECHNOLOGIES> .Acesso em 20 out. 2020.

EXAME. *5 bilhões de pessoas têm smartphones*. 2017. Disponível em: <<https://exame.com/tecnologia/5-bilhoes-de-pessoas-tem-smartphones/>>. Acesso em: 10 out. 2020.

EXAME. *75% do uso de internet em 2017 será por dispositivos móveis*. 2016. Disponível em: <<https://exame.com/tecnologia/75-do-uso-de-internet-em-2017-sera-por-dispositivos-moveis/>>. Acesso em: 10 out. 2020.

IONIC. *All about Ionic*. 2020. Disponível em: <<https://ionicframework.com/about>>. Acesso em: 20 out. 2020.

REACT. *A JavaScript library for building user interfaces*. 2020. Disponível em: <<https://reactjs.org/>>. Acesso em: 22 out. 2020.

SAKS, Elar. *JavaScript frameworks: Angular vs React vs Vue*. 2019. Disponível em: <<https://www.theseus.fi/bitstream/handle/10024/261970/Thesis-Elar-Saks.pdf>>. Acesso em 20 out. 2020.