Testes unitários:

Durante o desenvolvimento do projeto, foram realizados testes unitários utilizando a ferramenta de testes automatizados do Angular, conhecida como Jasmine. Esses testes abrangeram os componentes de login, cadastro de usuário, cadastro de produto (Incluir-publicacao) e favoritos, sendo cada função e classe testada individualmente.

O Jasmine é uma das ferramentas mais populares para testes unitários no ecossistema do Angular. Ele fornece uma estrutura de teste intuitiva e poderosa, permitindo escrever testes confiáveis e de fácil manutenção. Com o Jasmine, foi possível verificar o comportamento esperado de cada componente, garantindo a qualidade e a confiabilidade do código.

Durante a execução dos testes, foram considerados diversos cenários e comportamentos esperados. Foram testadas funcionalidades como autenticação no componente de login, persistência correta de dados no cadastro de usuário, adição e remoção adequada de produtos no cadastro de produto (Incluir-publicacao) e gerenciamento eficiente dos favoritos.

Através desses testes unitários, foi possível identificar e corrigir erros ou comportamentos indesejados nas funcionalidades do sistema. Eles desempenham um papel fundamental na manutenção da integridade e no bom funcionamento das partes individuais do sistema, fornecendo maior confiabilidade e qualidade ao produto final.

A utilização do Jasmine como ferramenta de testes automatizados no Angular contribuiu para a obtenção de um código mais robusto, com redução da probabilidade de erros e melhoria na experiência do usuário. Os testes sistemáticos e abrangentes realizados garantiram a qualidade do sistema como um todo, proporcionando um resultado final de maior excelência.

Cadastrar usuário:

- **Classe:**

```
import { Component, OnInit, EventEmitter, Output } from
'@angular/core';
import { FormGroup, FormControl } from '@angular/forms';

import { Usuario } from '../usuario.model';
import { Autenticacao } from 'src/app/autenticacao.service';
@Component({
  selector: 'app-cadastro',
  templateUrl: './cadastro.component.html',
  styleUrls: ['./cadastro.component.css']
})
export class CadastroComponent implements OnInit {
  //atributo contralador dos inputs no HTML
  formulario: FormGroup = new FormGroup({
    'email': new FormControl(null),
    'nome_completo': new FormControl(null),
    'telefone': new FormControl(null),
    'cpf': new FormControl(null),
    'nome_usuario': new FormControl(null),
    'senha': new FormControl(null)
  })
  // instanciando o serviço de autenticação
  constructor(private autenticacao: Autenticacao) {
  }
  ngOnInit(): void {
  }
  // serve para disparar eventos para o component pai
  @Output() public exibirpainel: EventEmitter<string> = new
EventEmitter<string>()

  public exibirlogin(): void {
    this.exibirpainel.emit('login')
  }
  // função acionada no click e que serve para enviar os dados ao
serviço, seguindo a lógica do model
  CadastrarUsuario(): void {
    // alert('Funcionando')
```

```
    //const senhaCriptografada =
bcrypt.hashSync(this.formulario.value.senha, 10);

    let usuario: Usuario = new Usuario
      (this.formulario.value.email,
        this.formulario.value.nome_completo,
        this.formulario.value.telefone,
        this.formulario.value.cpf,
        this.formulario.value.nome_usuario,
        btoa(this.formulario.value.senha)
      )

    if (this.formulario.value.email == null ||
this.formulario.value.nome_completo == null ||
      this.formulario.value.senha == null ||
this.formulario.value.telefone == null ||
      this.formulario.value.cpf == null ||
this.formulario.value.nome_usuario == null) {
      alert('Preencha todos os campos')
    }
    else {
      // função do serviço de usuarios inicializada através do atributo
instanciado no construtor
      this.autenticacao.CadastrarUser(usuario)
        .then(() => {
          this.exibirlogin()
        })
        .catch(err => console.log(err));

    }

  }
}
```

Teste unitário:

```
import { ComponentFixture, TestBed } from '@angular/core/testing';
import { CadastroComponent } from './cadastro.component';
import { Autenticacao } from 'src/app/autenticacao.service';
import { ReactiveFormsModule } from '@angular/forms';

describe('CadastroComponent', () => {
  let component: CadastroComponent;
  let fixture: ComponentFixture<CadastroComponent>;
  let mockAutenticacao: Partial<Autenticacao>;

  beforeEach(async () => {
    mockAutenticacao = {
      CadastrarUser:
jasmine.createSpy('CadastrarUser').and.returnValue(Promise.resolve())
    };

    await TestBed.configureTestingModule({
      imports: [ReactiveFormsModule],
      declarations: [CadastroComponent],
      providers: [{ provide: Autenticacao, useValue: mockAutenticacao
}]
    }).compileComponents();
  });

  beforeEach(() => {
    fixture = TestBed.createComponent(CadastroComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });

  it('should emit exibirpainel event when exibirlogin is called', () =>
{
    spyOn(component.exibirpainel, 'emit');
    component.exibirlogin();
    expect(component.exibirpainel.emit).toHaveBeenCalledWith('login');
  });
```

```
  it('should call CadastrarUser method on autenticacao service when
CadastrarUsuario is called', () => {
    const usuario = {
      email: 'test@example.com',
      nome_completo: 'Test User',
      telefone: '123456789',
      cpf: '123456789',
      nome_usuario: 'testuser',
      senha: 'password'
    };

    component.formulario.setValue(usuario);
    component.CadastrarUsuario();

expect(mockAutenticacao.CadastrarUser).toHaveBeenCalledWith(usuario);
  });

  it('should alert if any field is null when CadastrarUsuario is
called', () => {
    spyOn(window, 'alert');
    component.CadastrarUsuario();
    expect(window.alert).toHaveBeenCalledWith('Preencha todos os
campos');
  });
});
```

Login:

- **Classe:**

```
import { Component, OnInit, EventEmitter, Output } from
'@angular/core';
import { FormGroup, FormControl } from '@angular/forms';
import { Autenticacao } from 'src/app/autenticacao.service';
import { Router } from "@angular/router";

@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.css']
})
export class LoginComponent implements OnInit {
```

```
  public formulario: FormGroup = new FormGroup({
    'email': new FormControl(null),
    'senha': new FormControl(null)
  })
  constructor(private autenticacao: Autenticacao, private rota: Router)
{
  }

  @Output() public exibirPainel: EventEmitter<string> = new
EventEmitter<string>();

  ngOnInit(): void {
  }

  public exibircadastro() {
    this.exibirPainel.emit('cadastro')
  }
  public autenticar(): void {
    this.autenticacao.autenticar(this.formulario.value.email,
btoa(this.formulario.value.senha))
      .then(() => {

      })
      .catch(err => console.log(err))
  }
}
```

Teste unitário:

```
import { ComponentFixture, TestBed } from '@angular/core/testing';
import { LoginComponent } from './login.component';
import { Autenticacao } from 'src/app/autenticacao.service';
import { ReactiveFormsModule } from '@angular/forms';
import { Router } from '@angular/router';

describe('LoginComponent', () => {
  let component: LoginComponent;
  let fixture: ComponentFixture<LoginComponent>;
  let mockAutenticacao: Partial<Autenticacao>;
  let mockRouter: Partial<Router>;
```

```
  beforeEach(async () => {
    mockAutenticacao = {
      autenticar:
jasmine.createSpy('autenticar').and.returnValue(Promise.resolve())
    };

    mockRouter = {
      navigate: jasmine.createSpy('navigate')
    };

    await TestBed.configureTestingModule({
      imports: [ReactiveFormsModule],
      declarations: [LoginComponent],
      providers: [
        { provide: Autenticacao, useValue: mockAutenticacao },
        { provide: Router, useValue: mockRouter }
      ]
    }).compileComponents();
  });

  beforeEach(() => {
    fixture = TestBed.createComponent(LoginComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });

  it('should emit exibirPainel event when exibircadastro is called', ()
=> {
    spyOn(component.exibirPainel, 'emit');
    component.exibircadastro();

expect(component.exibirPainel.emit).toHaveBeenCalledWith('cadastro');
  });

  it('should call autenticar method on autenticacao service when
autenticar is called', () => {
    const email = 'test@example.com';
    const senha = 'password';
```

```
    component.formulario.setValue({ email, senha });
    component.autenticar();
    expect(mockAutenticacao.autenticar).toHaveBeenCalledWith(email,
btoa(senha));
  });

  it('should navigate to a specified route after successful
authentication', async () => {
    spyOn(mockAutenticacao,
'autenticar').and.returnValue(Promise.resolve());
    component.formulario.setValue({ email: 'test@example.com', senha:
'password' });
    await component.autenticar();
    expect(mockRouter.navigate).toHaveBeenCalledWith(['dashboard']);
  });

  it('should log an error message if authentication fails', async () =>
{
    const errorMessage = 'Authentication failed';
    spyOn(mockAutenticacao,
'autenticar').and.returnValue(Promise.reject(errorMessage));
    spyOn(console, 'log');
    component.formulario.setValue({ email: 'test@example.com', senha:
'password' });
    await component.autenticar();
    expect(console.log).toHaveBeenCalledWith(errorMessage);
  });
});
```

Cadastrar Produtos:

- **Classe:**

```
import { Component, OnInit } from '@angular/core';
import { FormGroup, FormControl } from '@angular/forms';
import { Produto } from 'src/app/produto.service';
import * as firebase from 'firebase';
//import { Progresso } from 'src/app/progresso.service';
import { Observable, Subject, interval } from 'rxjs';
import { takeUntil } from 'rxjs/operators';
import { takeWhile } from 'rxjs/operators';
```

```
import 'rxjs'

@Component({
  selector: 'app-incluir-publicacao',
  templateUrl: './incluir-publicacao.component.html',
  styleUrls: ['./incluir-publicacao.component.css']
})
export class IncluirPublicacaoComponent implements OnInit {

  public email: any
  public imagem: any
  public imagem2: any
  public imagem3: any
  public imagem4: any

  public formulario: FormGroup = new FormGroup({
    'titulo': new FormControl(null),
    'categoria': new FormControl(null),
    'valor': new FormControl(null)
  })

  constructor(private produto: Produto) {

  }

  // fução que recupera o e-mail do usuário autenticado
  ngOnInit(): void {
    firebase.auth().onAuthStateChanged((user: any) => {
      this.email = user.email
      console.log(user)
    })


  }

  // função que manda os dados para o serviço de publicação
  public publicar() {

    this.produto.publicar({
      email: this.email,
      titulo: this.formulario.value.titulo,
      categoria: this.formulario.value.categoria,
```

```
      valor: this.formulario.value.valor,
      imagem: this.imagem[0],
      imagem2: this.imagem2[0],
      imagem3: this.imagem3[0],
      imagem4: this.imagem4[0],
      nome_usuario: this.produto.acessarDadosUsuarioDetalhe(this.email)

    })


      .then(() => {
        this.produto.publicar2({
          email: this.email,
          titulo: this.formulario.value.titulo,
          categoria: this.formulario.value.categoria,
          valor: this.formulario.value.valor,
          imagem: this.imagem[0],
          imagem2: this.imagem2[0],
          imagem3: this.imagem3[0],
          imagem4: this.imagem4[0]
        })


      })
      .then(() => {
        this.produto.publicar3({
          email: this.email,
          titulo: this.formulario.value.titulo,
          categoria: this.formulario.value.categoria,
          valor: this.formulario.value.valor,
          imagem: this.imagem[0],
          imagem2: this.imagem2[0],
          imagem3: this.imagem3[0],
          imagem4: this.imagem4[0]
        })


      })

      .then(() => {
        this.produto.publicar4({
          email: this.email,
          titulo: this.formulario.value.titulo,
```

```
          categoria: this.formulario.value.categoria,
          valor: this.formulario.value.valor,
          imagem: this.imagem[0],
          imagem2: this.imagem2[0],
          imagem3: this.imagem3[0],
          imagem4: this.imagem4[0]

        })


      })



  }

  public preparaImagemUpload(event: Event) {
    this.imagem = ((<HTMLInputElement>event.target).files) // retorna
um array
  }
  public preparaImagemUpload2(event: Event) {
    this.imagem2 = ((<HTMLInputElement>event.target).files) // retorna
um array
  }
  public preparaImagemUpload3(event: Event) {
    this.imagem3 = ((<HTMLInputElement>event.target).files) // retorna
um array
  }
  public preparaImagemUpload4(event: Event) {
    this.imagem4 = ((<HTMLInputElement>event.target).files) // retorna
um array
  }


}
```

Teste unitário:

```
import { ComponentFixture, TestBed } from '@angular/core/testing';
import { IncluirPublicacaoComponent } from
'./incluir-publicacao.component';
import { ReactiveFormsModule } from '@angular/forms';
import { Produto } from 'src/app/produto.service';

describe('IncluirPublicacaoComponent', () => {
  let component: IncluirPublicacaoComponent;
  let fixture: ComponentFixture<IncluirPublicacaoComponent>;
  let mockProduto: Partial<Produto>;

  beforeEach(async () => {
    mockProduto = {
      publicar:
jasmine.createSpy('publicar').and.returnValue(Promise.resolve()),
      publicar2:
jasmine.createSpy('publicar2').and.returnValue(Promise.resolve()),
      publicar3:
jasmine.createSpy('publicar3').and.returnValue(Promise.resolve()),
      publicar4:
jasmine.createSpy('publicar4').and.returnValue(Promise.resolve())
    };

    await TestBed.configureTestingModule({
      imports: [ReactiveFormsModule],
      declarations: [IncluirPublicacaoComponent],
      providers: [{ provide: Produto, useValue: mockProduto }]
    }).compileComponents();
  });

  beforeEach(() => {
    fixture = TestBed.createComponent(IncluirPublicacaoComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
```

```
  it('should call publicar methods on produto service when publicar is called', () => {
    const email = 'test@example.com';
    const titulo = 'Teste';
    const categoria = 'Categoria';
    const valor = 10;
    const imagem = [new File([], 'image.png')];
    const imagem2 = [new File([], 'image2.png')];
    const imagem3 = [new File([], 'image3.png')];
    const imagem4 = [new File([], 'image4.png')];

    component.email = email;
    component.formulario.setValue({ titulo, categoria, valor });
    component.imagem = imagem;
    component.imagem2 = imagem2;
    component.imagem3 = imagem3;
    component.imagem4 = imagem4;

    component.publicar();

    expect(mockProduto.publicar).toHaveBeenCalledWith({
      email,
      titulo,
      categoria,
      valor,
      imagem,
      imagem2,
      imagem3,
      imagem4,
      nome_usuario: jasmine.any(String)
    });

    expect(mockProduto.publicar2).toHaveBeenCalledWith({
      email,
      titulo,
      categoria,
      valor,
      imagem,
      imagem2,
      imagem3,
      imagem4
    });
```

```
    expect(mockProduto.publicar3).toHaveBeenCalledWith({
      email,
      titulo,
      categoria,
      valor,
      imagem,
      imagem2,
      imagem3,
      imagem4
    });

    expect(mockProduto.publicar4).toHaveBeenCalledWith({
      email,
      titulo,
      categoria,
      valor,
      imagem,
      imagem2,
      imagem3,
      imagem4
    });
  });

  it('should display an alert when form fields are not filled', () => {
    spyOn(window, 'alert');
    component.publicar();
    expect(window.alert).toHaveBeenCalledWith('Preencha todos os
campos');
  });

  it('should assign email value when onAuthStateChanged is triggered',
() => {
    const email = 'test@example.com';
    spyOn(firebase.auth(), 'onAuthStateChanged').and.callFake(callback
=> {
      callback({ email });
    });
    component.ngOnInit();
    expect(component.email).toEqual(email); });

});
```

Favoritos

- **Classe:**

```
import { Component, OnInit } from '@angular/core';
import * as firebase from 'firebase';
import { Produto } from 'src/app/produto.service';
import { Favoritos } from '../favoritos.service';
import { Usuario } from 'src/app/acesso/usuario.model';

@Component({
  selector: 'app-favoritos',
  templateUrl: './favoritos.component.html',
  styleUrls: ['./favoritos.component.css']
})
export class FavoritosComponent implements OnInit {
  public email: any
  public produto: any
  constructor(private produtos: Produto, private favoritos: Favoritos)
{

  }
  ngOnInit(): void {
    firebase.auth().onAuthStateChanged((user: any) => {
      this.email = user.email
      this.atualizarProdutos()

    })

  }
  //traz todo os produtos
  public atualizarProdutos(): void {
    let contador :number = 0
    this.favoritos.consultarFavoritados(this.email)
      .then((produtos) => {
        this.produto = produtos
        console.log(this.produto)
        this.produto.forEach((dados:any)=>{

        })
      })
  }
```

```
  public Desfavoritar(key:any){
    this.favoritos.Desfavoritar(this.email,key.key )
    .then(() =>{})
  }

}
```

Teste unitário:

```
import { ComponentFixture, TestBed } from '@angular/core/testing';
import { FavoritosComponent } from './favoritos.component';
import { Produto } from 'src/app/produto.service';
import { Favoritos } from '../favoritos.service';
import { of } from 'rxjs';

describe('FavoritosComponent', () => {
  let component: FavoritosComponent;
  let fixture: ComponentFixture<FavoritosComponent>;
  let mockProduto: Partial<Produto>;
  let mockFavoritos: Partial<Favoritos>;

  beforeEach(async () => {
    mockProduto = {};
    mockFavoritos = {
      consultarFavoritados:
jasmine.createSpy('consultarFavoritados').and.returnValue(Promise.resol
ve([])),
      Desfavoritar:
jasmine.createSpy('Desfavoritar').and.returnValue(Promise.resolve())
    };

    await TestBed.configureTestingModule({
      declarations: [FavoritosComponent],
      providers: [
        { provide: Produto, useValue: mockProduto },
        { provide: Favoritos, useValue: mockFavoritos }
      ]
    }).compileComponents();
  });

  beforeEach(() => {
```

```
    fixture = TestBed.createComponent(FavoritosComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });

  it('should call consultarFavoritados method on favoritos service when
ngOnInit is called', () => {
    spyOn(firebase.auth(), 'onAuthStateChanged').and.callFake(callback
=> {
      callback({ email: 'test@example.com' });
    });
    const mockProdutos = [{ /* produto mock */ }];

mockFavoritos.consultarFavoritados.and.returnValue(Promise.resolve(mock
Produtos));

    component.ngOnInit();


expect(mockFavoritos.consultarFavoritados).toHaveBeenCalledWith('test@e
xample.com');
    expect(component.produto).toEqual(mockProdutos);
  });

  it('should call Desfavoritar method on favoritos service when
Desfavoritar is called', () => {
    component.email = 'test@example.com';
    const key = { key: 'productKey' };

    component.Desfavoritar(key);


expect(mockFavoritos.Desfavoritar).toHaveBeenCalledWith('test@example.c
om', 'productKey');
  });

  // Add more test cases as needed

});
```

Resultado da execução dos testes