

# ANGULAR 17 COM FIREBASE

## ÍNDICE

[Instalação dos programas](#)

[Componentes em angular](#)

[Inserindo imagens e css](#)

[Ciclo de vida de um componente angular](#)

[Criação do componente login](#)

[Criando projeto no firebase](#)

[Consumindo o firebase](#)

[Model e passagem de dados](#)

[Enviando dados para o firebase](#)

[Exclusão com doubleclick](#)

[Lazyloading com difer](#)

\* angular.dev - site da documentação que é uma IDE.

\* Formas de rodar um projeto: ng serve / npm run start

\* Firebase: banco de dados por interface gráfica, e pode ser usado por diversas tecnologias, como: Android, IOS, Javascript, [Node.js](#), Java, PHP, Unit, C++.

## INSTALANDO OS PROGRAMAS

1) nvm: <https://github.com/coreybutler/nvm-windows/releases>

2) Configure versão 20.11.0 do node com o comando: nvm install 20.11.0

3) Baixe o angular cli: npm install -g @angular/cli@17

4) Baixe e instale o vscode.

5) Crie o projeto com o comando: ng new crud-usuarios --no-standalone ⇒ **--no-standalone** é para criar o projeto com módulos.

\*\* o comando "code ." abre o vscode a partir de qq pasta.

OBS: Se quiser usar o jest ao invés do jasmine para testes:

5.1) instale o jest: `npm install --save-dev jest`

5.2) No diretório raiz do seu projeto, crie um novo arquivo chamado jest.config.js

5.3) Dentro do arquivo jest.config.js, você pode adicionar as configurações que deseja para o seu projeto de testes. Aqui está um exemplo de um arquivo de configuração básico:

```
/** @type {import('jest').JestConfigWithTsSupport} */
module.exports = {
  preset: 'ts-jest', // Para projetos com TypeScript
  testEnvironment: 'node', // Ambiente de teste (por exemplo, 'node' ou 'jsdom')
  // ... outras configurações
};
```

Explicando algumas opções de configuração:

**preset: 'ts-jest':**

Se o seu projeto usar TypeScript, esta configuração habilita o preset do Jest para TypeScript.

`testEnvironment: 'node':`

Define o ambiente de teste. 'node' para ambientes Node.js e 'jsdom' para ambientes de navegador.

Exemplo de um teste básico:

```
// exemplo.js
```

```
function soma(a, b) {  
  return a + b;  
}
```

```
// exemplo.spec.js
```

```
test('A função soma deve retornar a soma de dois números', () => {  
  expect(soma(2, 3)).toBe(5);  
});
```

5.4) Executando os testes: `npx jest`

Fim da explicação de testes com jest.

\*\* No arquivo `tsconfig.json`, na propriedade `compilerOptions: strictPropertyInitialization: false` ⇒ serve para desabilitar a obrigação de ter que inicializar uma variável ao criá-la.

6) Instalando o bootstrap: `npm install bootstrap@5.2.3`

\* `angular.json` ⇒ Há configurações para qual tema usar para ambiente de produção e de desenvolvimento.

Importe o bootstrap no arquivo `styles.scss`:

```
@import '../node_modules/bootstrap/dist/css/bootstrap.min.css';
```

Teste o bootstrap com um parágrafo vermelho no arquivo `app.component.html`:

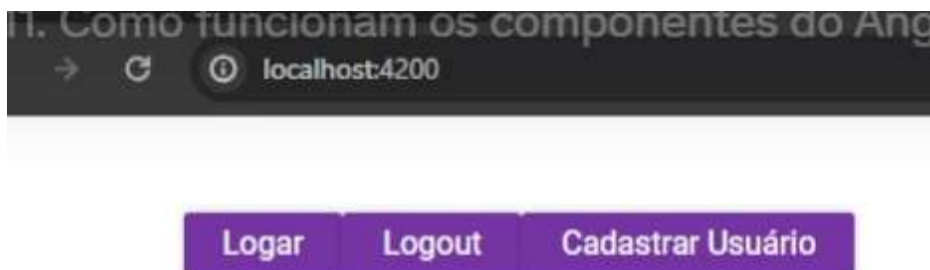
```
<p class="bg-danger">ARR000000H</p>
```

## Aula 11) Componentes em Angular

```
button.component.html U    TS button.component.ts U    app.component.html M X
src > app > components > button > TS button.component.ts > ButtonComponent
1  import { Component, Input } from '@angular/core';
2
3  @Component({
4    selector: 'app-button',
5    templateUrl: './button.component.html',
6    styleUrls: ['./button.component.scss']
7  })
8  export class ButtonComponent {
9
10   @Input() textButton: string;
11
12 }
13
```

```
button.component.html U    TS button.component.ts U    app.component.html M X
src > app > app.component.html > div.container.mt-5 > app-button
1
2 <router-outlet></router-outlet>
3
4 <div class="container mt-5">
5   <app-button textButton="Logar"></app-button>
6   <app-button textButton="Logout"></app-button>
7   <app-button textButton="Cadastrar Usuário"></app-button>
8 </div>
```

Resultado:



Se fizer assim:

```
<app-button [textButton]="textRegisterUser"></app-button>
```

```
export class AppComponent {
  textRegisterUser = 'Cadastrar Usuários';
  title = 'crud-usuarios';
}
```

Também terá o mesmo resultado.

## Aula 12) Inserindo imagens e css

```
styles.scss M ● app.component.html M ●
src > styles.scss > .c-primary
1  /* You can add global styles to this file, and also import other style files */
2  @import '../node_modules/bootstrap/dist/css/bootstrap.min.css';
3  html, body { height: 100%; }
4  body { margin: 0; font-family: Roboto, "Helvetica Neue", sans-serif; }
5
6  :root {
7    --primary-color: #7533a8;
8    --error-color: #ac2828;
9    --gray-color: #747474;
10 }
11
12 .c-primary{
13   color: var(--)
14 }
```

--error-color	#ac2828
--gray-color	#747474
--primary-color	#7533a8

## Aula 13) Ciclo de vida de um componente Angular

constructor	A component has a lifecycle managed by Angular.
ngOnChanges	Angular creates it, renders it, creates and renders its children, checks it when its data-bound properties change, and destroys it before removing it from the DOM.
ngOnInit	
ngDoCheck	
ngAfterContentInit	Angular offers <b>lifecycle hooks</b> that provide visibility into these key life moments and the ability to act when they occur.
ngAfterContentChecked	
ngAfterViewInit	
ngAfterViewChecked	A directive has the same set of lifecycle hooks, minus the hooks that are specific to component content and views.
ngOnDestroy	

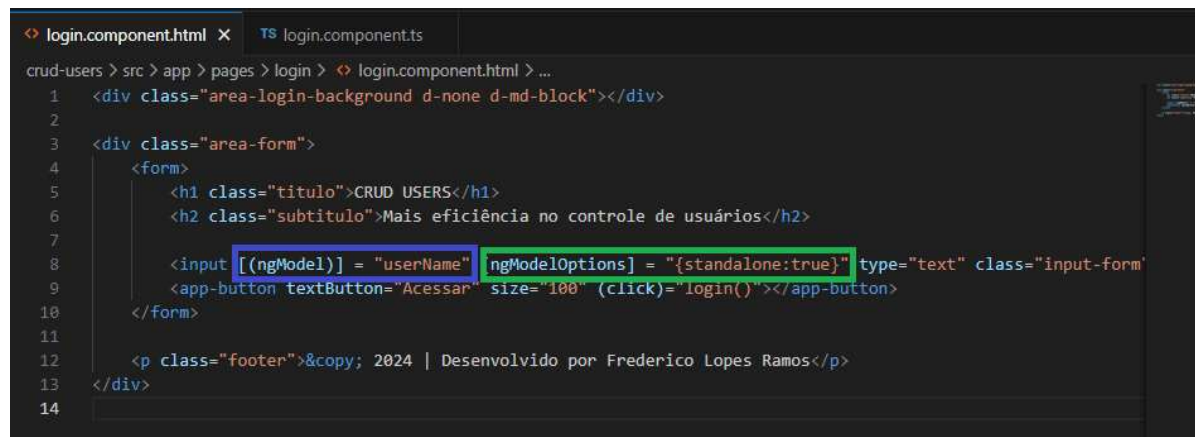
```
app / components / button / → buttoncomponent / → ngOnInit  
selector: 'app-button',  
templateUrl: './button.component.html',  
styleUrl: './button.component.scss'  
})  
export class ButtonComponent {  
  
  @Input() textButton: string = 'LifeCycle Hooks';  
  
  constructor() {  
    console.log('Componente construido');  
  }  
  
  ngOnChanges() {  
    console.log('componente changes');  
  }  
  
  ngOnInit() {  
    console.log('componente inicializado');  
  }  
  
  ngDoCheck() {  
    console.log('componente check');  
  }  
  
  ngAfterViewInit() {  
    console.log('componente view inicializada');  
  }  
  
  ngOnDestroy() {  
    window.alert('destruido')  
  }  
}
```

## Aula 15 - Criação do componente Login

1) na pasta do projeto, rode o comando: `ng g c pages/login`

2) Adicione as rotas

## Aula 17) Estrutura da tela



```
login.component.html x TS login.component.ts
crud-users > src > app > pages > login > login.component.html > ...
1 <div class="area-login-background d-none d-md-block"></div>
2
3 <div class="area-form">
4   <form>
5     <h1 class="titulo">CRUD USERS</h1>
6     <h2 class="subtitulo">Mais eficiência no controle de usuários</h2>
7
8     <input [(ngModel)] = "userName" [ngModelOptions] = "{standalone:true}" type="text" class="input-form" />
9     <app-button textButton="Acessar" size="100" (click)="login()"></app-button>
10   </form>
11
12   <p class="footer">&copy; 2024 | Desenvolvido por Frederico Lopes Ramos</p>
13 </div>
14
```

Para o `ngModel` funcionar, tem que usar o `ngModelOptions`.

## Aula 18) Criação do componente home

## Aula 19) Menu lateral

## Aula 21) Estrutura da home

\*\* Se mexeu no arquivo de modulos ou no de rotas, pare o servidor e rode-o novamente para certificar-se de que as alteracoes foram compiladas.

## Aula 22) Construção da tela Home

## Aula 24) Criando projeto no firebase



```
passos-firebase.txt - Bloco de Notas
Arquivo  Editar  Formatar  Exibir  Ajuda
1 - Criar projeto no Firebase
2 - Criar projeto
3 - Criar Firestore DataBase (nosso banco de dados)
4 - Criar um app (para adicionar no Environment (criar environment ng g environments)
4 - Incluir código do app no environment
5 - Rodar no terminal "npm install firebase"
6 - Rodar no terminal "ng add @angular/fire"
7 - Adicionar no APP MODULE:
AngularFireModule.initializeApp(environment.firebaseConfig),
import { AngularFireModule } from '@angular/fire/compat';
```

Rode os comandos:

`ng g environments` e `ng add @angular/fire`

Adicione em [app.module.ts](#):

```
import {AngularFireModule} from '@angular/fire/compat';
```

e nos imports:

```
imports:[
  AngularFireModule.initializeApp(environment.firebaseConfig)
]
```

`ng g interface interfaces/user`

## Aula 27) Criação do componente de CRUD

`ng g component pages/crud`



## Aula 30) Consumindo serviço Firebase e preenchendo a tabela de usuarios

Forma nova de se fazer if:

```
<ng-container matColumnDef="benefits">
  <th class="th-table" mat-header-cell *matHeaderCellDef mat-sort-header>Beneficios </th>
  <td mat-cell *matCellDef="let user">
    <!-- FORMA ANTIGA DE SE FAZER ngIf -->
    <mat-icon class="c-primary" *ngIf="user.health || user.dentalPlan">check</mat-icon>
    <mat-icon class="c-error" *ngIf="!user.health && !user.dentalPlan">remove</mat-icon>
  </td>-->

  <!-- FORMA NOVA DE SE FAZER ngIf -->
  @if(user.healthPlan || user.dentalPlan){
    <mat-icon class="c-primary">check</mat-icon>
  }
  @if(!user.healthPlan && !user.dentalPlan){
    <mat-icon class="c-error">remove</mat-icon>
  }
</td>
</ng-container>
```

## Aula 31) Modal e passagem de dados

ng g c pages/crud/modal-view-user

## Aula 33) Criação do modal adicionar usuarios

ng g c pages/crud/modal-form-user

## Aula 34) Modal para adicionar usuarios + typescript

\*\* Criando os selects dos benefícios plano de saúde e odontológico

- 1) Crie as listas de objetos no typescript
- 2) Insira o MatSelectModule em app.module.ts

## Aula 35) Enviando dados para o firebase

Para pegar os dados do formulario com apenas uma linha:

```
saveUser(){  
  const objUserForm = this.formUser.getRawValue();  
}
```

Funcao salvar completa:

```
saveUser(){  
  const objUserForm:User = this.formUser.getRawValue();  
  this.userService.addUser(objUserForm).then(  
    (response:any) => {  
      window.alert('Usuario salvo com sucesso.');      this.closeModal();  
    })  
    .catch(  
      err => {  
        window.alert('Um erro ocorreu...');        console.error(err);  
      });  
}
```

## Aula 37) Exclusão com double click

```
<mat-icon class="icone-tabela" (click)="openModalEditUser(user)">edit</mat-icon>  
<mat-icon class="icone-tabela"(dblclick)="deleteUser(user.firebaseId)">delete</mat-icon>  
</td>
```

## Aula 38) LazyLoading com difer

```
<div class="col-12 col-md-4">
  @defer {
    
  } @placeholder (minimum 2000ms) {
    Carregando imagem ...
  } @error {
    Falha ao carregar a imagem
  }
</div>
```