

Como criar formulário de login no angular ↩️

1. INDEX

1. [Resumo do conteúdo](#)

2. [Introdução](#)

1. [Objetivo.](#)
2. [Pre-requisitos.](#)
3. [Benefícios.](#)

3. [Como criar formulário de login no Angular 2.](#)

1. [Passo 01](#) - Criar a classe **LoginComponent**...
2. [Passo 02](#) - Habilitar a diretiva **[(ngModel)]** em **src/app/login.component.html**...
3. [Passo 03](#) - Criar classe **Usuario** em **src/app/page/login/usuario.ts**...
4. [Passo 04](#) - Criar serviço **AuthService** em **src/app/page/login/auth.service.ts**...
5. [Passo 05](#) - Editar o template **src/app/login.component.html** para ler nome do **usuário** e **senha**...
6. [Passo 06](#) - Registrar o componente **LoginComponent** na matriz **const routes: Routes = []** do arquivo **src/app/app-routing.module.ts**...
7. [Passo 07](#) - Criar menubar em : **src/app/app.component.html**...
8. [Passo 08](#) - Criar a classe **HomeComponent** em **pages/home/home.component.ts** para demonstrar o passo 09.
9. [Passo 09](#) - Ocultar a barra de menu quando o usuário não tiver autenticado
10. [Passo 10](#) - Usando guarda de rotas para bloquear página html do usuário não autenticado

4. [Referências globais.](#)

5. [Histórico.](#)

2. CONTEÚDO

1. **Resumo do conteúdo:**

1. Este documento encina o passo a passo de como implementar com o framework angular 12+ a autenticação do usuário.
2. A autenticação cria dois serviços sendo o **AuthService** para autenticação e o **AuthGuardService** para guarda de rotas.
3. Os **passos 01 a 06** é criado o **formulário de login** e o serviço de autenticação **AuthService**.
 1. Ao clicar no botão **enviar do formulário** de login, o sistema checa se o campo **usuário.nome** é igual a **paulospacheco@yahoo.com.br** e o campo **usuário.senha** é

igual a **12345**, em seguida imprime um formulário de mensagem informando se usuário está conectado ou não.

4. Os **passos 07 a 09** é implementado a **página home** com o propósito de implementar o **menu de opções** da aplicação.

1. Quando o usuário está conectado o menu é mostrado e quando o usuário está desconectado o menu é omitido.

5. O **passo 10** é implementado a trava nas rotas onde a página só é visualizada se o usuário estiver conectado, exceto a página de login.

2. Introdução

1. Objetivo:

1. Esse exemplo descreve todos os passos para criar um formulário de login com autenticação usando um serviço do angular .

2. [←
BACK]

2. Pre-requisitos:

1. Nodejs instalado.

2. Framework Angular 2 ou superior instalado.

3. Projeto deve ter sido criado com o comando **ng new nome-do-projeto**

4. Domínio da linguagem typescript

5. Domínio da linguagem html

6. Domínio da linguagem css

7. [←
BACK]

3. Benefícios:

1. Conhecimento de tudo que precisa saber para autenticar usuários.

2. [←
BACK]

3. Como criar formulário de login no Angular 2

1. Passo 01

1. Criar a classe **LoginComponent** em **pages/login/login.component.ts**.

1. Código shellscript

```
# Entrar na raiz do projeto e executar o comando:  
ng generate component pages/login
```

2. Referências:

1. [Criando component angular 2 com a cli](#)

3. [\[← BACK\]](#)

2. Passo 02

1. Para usar a diretiva **[(ngModel)]** em **src/app/login.component.html** é necessário importar em **src/app/app.module.ts** os seguintes componentes:

1. **FormsModule**, exporta os provedores e diretivas necessários para formulários orientados a modelos, tornando-os disponíveis para importação por **NgModules** que importam este módulo.
2. **ReactiveFormsModule**. exporta a infraestrutura e diretivas necessárias para os formulários reativos, disponibilizando-os para importação pelos **NgModules** que importam este módulo.

1. Código typescript

```
//file : **src/app/app.module.ts**  
  
import { FormsModule } from '@angular/forms';  
import { ReactiveFormsModule } from  
'@angular/forms';  
  
@NgModule(  
  {  
    declarations: [],  
    imports: [FormsModule,  
              ReactiveFormsModule  
            ],  
  }  
)
```

2. Referências:

1. [FormsModule](#)
2. [ReactiveFormsModule](#)

3. [\[← BACK\]](#)

3. Passo 03

1. Criar classe **Usuario** em **src/app/page/login/usuario.ts**:

1. Código shellscript e typescript.

1. Criar class usuario no prompt de comandos:

```
ng generate class pages/login/usuario
```

2. Adicionar os atributos **nome**, **senha** e **autenticado** na classe **Usuario** em **src/app/pages/login/usuario.ts**

```
//file: **src/app/pages/login/usuario.ts**

/**
 * Class Usuario
 * Objetivo: Guardar na memória o usuario
conectado
 */
export class Usuario {
  nome:string = "";
  senha:string = "";
  /**
   * o atributo autenticado é usado para
sinalizar se o usuário está conectado ou não.
   * False usuário desconectado,
   * true usuário conectado
   */
  autenticado : boolean = false;
}
```

3. Criar atributo **usuario**, injetar a classe **AuthService** e criar método **fazerLogin()** na classe **LoginComponent** do arquivo **src/app/login.component.ts**

```
//file: *src/app/login.component.ts**

export class LoginComponent implements OnInit {
  /**
   * Declarar o atributo usuário
   */
  public usuario:Usuario = new Usuario();
  /**
   * Injetar AuthService na classe
LoginComponent
   * @param authService
   */
  constructor(private authService : AuthService) {
  }

  /**
   * Método fazerLogin() usado para autenticar o
usuário usando o serviço injetado AuthService
   */
}
```

```
fazerLogin(){  
    /**  
     * Autentica usuário  
     */  
    this.authService.fazerLogin(this.usuario);  
  
    /**  
     * Imprime no console nome, senha e flag se  
    autenticado .  
     * ATENÇÃO: As 3 linhas abaixo não deve  
    existir em ambiente de produção.  
     */  
    console.log('Usuario: '+this.usuario.nome);  
    console.log('Senha: '+this.usuario.senha);  
  
    console.log('Autenticado: '+this.authService.autentica  
    cado);  
  
    /**  
     * Imprime mensagem se usuário foi  
    conectado ou não.  
     */  
    alert(this.authService.autenticado ? 'Usuário  
    autenticado' : 'Usuário não autenticado');  
}  
  
ngOnInit(): void {  
}  
  
}
```

2. Referências:

1. [ng generate class \[options\]](#)
2. [Class typescript](#)

3. BACK

4. Passo 04

1. Criar serviço **AuthService** em **src/app/page/login/auth.service.ts**:

1. Código shellscript e typescript.

1. Criar serviço **AuthService** no prompt de comandos:

```
ng generate service pages/login/auth
```

2. Registrar o serviço **AuthService** em **@NgModule(providers: [AuthService])** no arquivo **src/app/app.module.ts**:

```
//file: **src/app/app.module.ts**

import { FormsModule } from '@angular/forms';
import { ReactiveFormsModule } from
 '@angular/forms';

@NgModule({
  declarations: [],
  imports: [],
  /**
   * Registra serviço AuthService em
providers
   */
  providers: [AuthService],
})
```

3. Criar a propriedade **autenticado()** e **Injetar em constructor()** a classe **Router**

```
//file : **src/app/page/login/auth.service.ts**

import { Injectable } from '@angular/core';
import { Router, RouterModule } from
 '@angular/router';

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

  /**
   * A propriedade autenticado é usado para
sinalizar se o usuário está conectado ou não.
   * False usuário desconectado,
   * true usuário conectado
   */
  private _autenticado : boolean = false;
  get autenticado() : boolean {
    return this._autenticado;
  };

  /**
   * Injeta o parâmetro router na classe.
   * @param router
   */
  constructor(private router : Router) { }
```

4. Cria método **fazerLogin()** na classe **AuthService** do arquivo **src/app/page/login/auth.service.ts**

```
//file : **src/app/page/login/auth.service.ts**

import { Injectable } from '@angular/core';
import { Usuario } from './usuario';

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

    /**
     * Método fazerLogin usado par autenticar
     usuário.
     * @param usuario:Usuario
     */
    fazerLogin(usuario:Usuario)
    {
        /**
         * O nome 'paulosspacheco@yahoo.com.br' a
         senha '12345' deve ser substituído
         * por um serviço de autenticação no
         servidor.
         */
        if
        ((usuario.nome=== 'paulosspacheco@yahoo.com.br') &&
         (usuario.senha === '12345' ))
        {
            usuario.autenticado = true;
            this.router.navigate(['/']);
        } else
        {
            usuario.autenticado = false;
        }
    }
}
```

5.

2. **Referências:**

3. [ng generate service \[options\]](#)
4. [Injeção de dependência em Angular](#)

2. 

5. **Passo 05**

1. Editar o template **src/app/login.component.html** para ler nome do **usuário** e **senha**:

1. Página html sem interação com a classe **src/app/login.component.ts**

```

<!-- file: **src/app/login.component.html** -->
<p>login works!</p>
<div class="row">
  <div class="input-field" col-s-12>
    <label class="active"
for="usuario">Usuário</label>
    <input id="usuario" type="text"
class="validate">
  </div>

  <div class="input-field" col-s-12>
    <label class="active" for="senha">Senha</label>
    <input id="senha" type="password"
class="validate">
  </div>

  <button class="button" type="submit" name="action"
(onclick)="alert('Ok login');"> Login</button>
</div>

```

2. Página html com interação com a classe **src/app/login.component.ts**

```

<!-- file: **src/app/login.component.html** -->

<p>login works!</p>
<div class="row">
  <div class="input-field" col-s-12>
    <label class="active"
for="usuario">Usuário</label>
    <input [(ngModel)]="usuario.nome" id="usuario"
type="text" class="validate">

  </div>
  <div class="input-field" col-s-12>
    <label class="active" for="senha">Senha</label>
    <input [(ngModel)]="usuario.senha" id="senha"
type="password" class="validate">
  </div>

  <button class="button" type="submit" name="action"
(click)="fazerLogin()"> Login</button>
</div>

```

2. Referências:

1. [Usando ngModel em um controle autônomo](#)
2. [FormsModule](#)
3. [ReactiveFormsModule](#)

3. 

6. Passo 06

1. Para que o template **src/app/login.componente.html** seja visualizado é necessários registrar o componente **LoginComponent** na matriz **const routes: Routes = []** do arquivo **src/app/app-routing.module.ts**.

1. Código typescript

```
//file:**src/app/app-routing.module.ts**  
/**  
 * Importar componente na classe  
 */  
import { LoginComponent } from  
'./pages/login/login.component';  
  
/**  
 * Criar rotas para que seja reconhecida pelos  
 templates.html  
 */  
const routes: Routes = [  
  {path:'login', component:LoginComponent},  
];
```

2. Referências:

1. Importe **RouterModule** e **Routes** em seu módulo de roteamento.
2. Matriz de objetos **Routes**

3. 

7. Passo 07

1. Criar menubar em : **src/app/app.component.html**

1. Apagar todo o código criado automaticamente do arquivo **src/app/app.component.html**
2. Adicionar o código abaixo no arquivo **src/app/app.component.html**

```
<!--file: **src/app/app.component.html** -->  
  
<style>  
body {  
  margin: 0;  
  font-family: Arial, Helvetica, sans-serif;  
}  
  
.topnav {
```

```

        overflow: hidden;
        background-color: #333;
    }

    .topnav a {
        float: left;
        color: #f2f2f2;
        text-align: center;
        padding: 14px 16px;
        text-decoration: none;
        font-size: 17px;
    }

    .topnav a:hover {
        background-color: #ddd;
        color: black;
    }

    .topnav a.active {
        background-color: #04AA6D;
        color: white;
    }
</style>

<h1>Angular Router App</h1>
<p>Aqui deve ser inserido o menu com todas as opções
do projeto</p>
<!-- This nav gives you links to click, which tells
the router which route to use (defined in the routes
constant in AppRoutingModuleModule) -->
<!-- Menu de opções -->
<nav>
    <div class="topnav" id="myTopnav">
        <a routerLink="/login"
routerLinkActive="selected">login</a> |
    </div>
</nav>

<!-- The routed views render in the <router-outlet>--
>

<hr>
<router-outlet></router-outlet>

```

2. Referências:

1. Descrição da diretiva **router-outlet**
2. Diretiva **router-outlet**
3. Diretiva **RouterLink**

3. 

8. Passo 08

1. Criar a classe **HomeComponent** em **pages/home/home.component.ts** para demonstrar o passo 09.

1. Código shellscript

```
# Entrar na raiz do projeto e executar o comando:  
ng generate component pages/home
```

2. Para que o template **src/app/home.componente.html** seja visualizado é necessários registrar o componente **HomeComponent** na matriz **const routes: Routes = []** do arquivo **src/app/app-routing.module.ts**.

1. Código typescript

```
file:**src/app/app-routing.module.ts**  
/**  
 * Importar componente na classe  
 */  
import { HomeComponent } from  
'./pages/home/home.component';  
  
/**  
 * Criar rotas para que seja reconhecida pelos  
 templates.html  
 */  
const routes: Routes = [  
  {path: 'home', component: HomeComponent},  
];
```

3. Adicionar no menubar em **src/app/app.component.html** a classe **HomeComponent**:

```
<!-- Menu de opções -->  
<div class="topnav" id="myTopnav">  
  <a routerLink="/home"  
routerLinkActive="selected">home</a> |  
  <a routerLink="/login"  
routerLinkActive="selected">login</a> |  
</div>
```

4. Referências:

1. [Criando component angular 2 com a cli](#)
2. [Importe RouterModule e Routes em seu módulo de roteamento.](#)
3. [Matriz de objetos Routes](#)
4. [Diretiva RouterLink](#)

5. 

9. Passo 09

1. Ocultar a barra de menu quando o usuário não tiver autenticado.

1. Na classe **AuthService** do arquivo **src/app/page/login/auth.service.ts**:

1. Criar método **mostrarMenuEmitter** do tipo **EventEmitter**.

2. No método **fazerLogin()** executar os métodos:

1. **mostrarMenuEmitter.emit(true)** se usuário estiver autenticado;
2. **mostrarMenuEmitter.emit(false)** se usuário não tiver autenticado.

3. Código typescript da classe **AuthService**

```
//file:**src/app/page/login/auth.service.ts**

import { Injectable, EventEmitter } from
'@angular/core';
import { Router, RouterModule } from
'@angular/router';
import { Usuario } from './usuario';

@Injectable({
  providedIn: 'root'
})
export class AuthService {
  /**
   * O atributo autenticado é usado para
   sinalizar se o usuário está conectado ou não.
   * False usuário desconectado,
   * true usuário conectado
   */
  private _autenticado : boolean = false;
  get autenticado() : boolean {
    return this._autenticado;
  };

  /**
   * Método para enviar evento para a classe
   AppComponent
   * Nota: A Classe EventEmitter pertence ao pacote
   '@angular/core'.
   * ??? O Vscode encontra essa classe no
   pacote stream
   */
  public mostrarMenuEmitter :
  EventEmitter<boolean>;

  /**
```

```

    * Injeta o parâmetro router na classe.
    * @param router
    */
    constructor(private router : Router) {
        this.mostrarMenuEmitter = new
EventEmitter<boolean>();
    }

    /**
     * Método fazerLogin usado par autenticar
usuário.
     * @param usuario:Usuario
     */
    fazerLogin(usuario:Usuario)
    {
        /**
         * O nome 'paulosspacheco@yahoo.com.br' a
senha '12345' deve ser substituído
         * por um serviço de autenticação no
servidor.
         */
        if
((usuario.nome=== 'paulosspacheco@yahoo.com.br') &&
        (usuario.senha === '12345' ))
        {
            this._autenticado = true;

            this.mostrarMenuEmitter.emit(true);

            //mostra menu
            this.router.navigate(['/']);
        } else
        {
            this._autenticado = false;

            //esconde menu
            this.mostrarMenuEmitter.emit(false);
        }
    }
}

```

2. Na classe **AppComponent** do arquivo **src/app/app.component.ts**:

1. Criar atributo **mostrarMenu:boolean** e inicializar com false;
2. Criar **constructor** para **injetar** a classe **AuthService** na classe **AppComponent**;
3. Implementar a interface **OnInit()** na classe **AppComponent** para que o atributo **mostrarMenu** seja reconhecido pelo template **src/app/app.component.html**.

1. O método **ngOnInit()** override é usado para registrar o atributo **mostrarMenu** na classe **this.authService.mostrarMenuEmitter**.
2. Código typescript da classe **AppComponent**:

```
//file: **src/app/app.component.ts**

import { Component } from '@angular/core';
import { AuthService } from
'./pages/login/auth.service';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})

export class AppComponent {
  title:string = 'login';
  //Atributo usado pelo pela diretiva
  *ngIf do template './app.component.html'
  mostrarMenu:boolean = false;

  // Injetar o serviço AuthService na
  classe AppComponent.
  constructor(private
  authService:AuthService){}

  //Método override usado para implementar
  a interface OnInit().
  ngOnInit(){

    this.authService.mostrarMenuEmitter.subscribe(
      //Registrar o atributo
      this.mostrarMenu para que o template
      app.component.html o reconheça.
      register => this.mostrarMenu =
      register
    )
  }
}
```

3. Na barra de menu **<nav>** do template **src/app/app.component.html** implementar o código **<nav *ngIf = "mostrarMenu">**

1. Código html do template **src/app/app.component.html**

```
<!--file: src/app/app.component.html -->
```

```
<nav *ngIf = "mostrarMenu">
  <div class="topnav" id="myTopnav">
    <a routerLink="/home"
routerLinkActive="selected">home</a> |
    <a routerLink="/login"
routerLinkActive="selected">login</a> |
  </div>
</nav>
```

2. Referências:

1. [OnInit\(\)](#)
2. [Método EventEmitter.subscribe.](#)
3. [mostrarMenuEmitter.emit\(true\)](#)

3. [\[← BACK\]](#)

10. Passo 10 - Usando guarda de rotas para bloquear página html do usuário não autenticado

1. Criar serviço **AuthGuardService**:

1. Código shellscript

```
ng generate service guards/login/auth-guard
```

2. Código do serviço criado pelo comando acima:

```
//file : src/app/guards/login/auth-guard.service.ts

import { Injectable } from '@angular/core';

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

  constructor() { }
}
```

3. Registrar o serviço **AuthGuardService** em **@NgModule(providers: [AuthService,AuthGuardService])** no arquivo **src/app/app.module.ts**:

```
//file: **src/app/app.module.ts**
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppRoutingModule } from './app-
```

```
routing.module';
import { AppComponent } from './app.component';
import { LoginComponent } from
'./pages/login/login.component';
import { AuthService } from
'./pages/login/auth.service';
import { FormsModule, ReactiveFormsModule } from
'@angular/forms';
import { HomeComponent } from
'./pages/home/home.component';
import { AuthGuardService } from './guards/login/auth-
guard.service';

@NgModule({
  declarations: [
    AppComponent,
    LoginComponent,
    HomeComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    FormsModule,
    ReactiveFormsModule
  ],
  providers: [AuthService, //Torna público o
serviço de autenticação
AuthGuardService], //Torna público o
serviço de guarda de rotas
  bootstrap: [AppComponent]
})
export class AppModule { }
```

4. Referências:

1. [Importe RouterModule e Routes em seu módulo de roteamento.](#)
 2. [Matriz de objetos Routes.](#)
2. No serviço **AuthGuardService** implementar os métodos da interface **CanActivate()** e em **AuthGuardService.constructor()** injetar o serviço **AuthService** e a classe **Router**;

```
//file : src/app/guards/login/auth-guard.service.ts
import { AuthService } from
'./../../pages/login/auth.service';
import { Injectable } from '@angular/core';
import { ActivatedRoute, ActivatedRouteSnapshot,
CanActivate, RouterStateSnapshot, Router } from
'@angular/router';
import { Observable } from 'rxjs';
```



```

@Injectable({ providedIn: 'root'})
export class AuthGuardService implements CanActivate {

    //Injetar o serviço AuthService e a classe Router
    constructor(private authService:AuthService,
                 private router:Router) { }

    //Método da interface CanActivate
    canActivate( route: ActivatedRouteSnapshot, state:
    RouterStateSnapshot ):Observable<boolean>|boolean {
        if (this.authService.autenticado == true) {
            //usuário autenticado
            return true
        };

        // Se usuário não está conectado então executa a
        página de login
        this.router.navigate(['./login']);

        //usuário não autenticado
        return false;
    }
}

```

3. Em **const routes: Routes = []** do arquivo **src/app/app-routing.module.ts** passar como parâmetro em casa rota a classe **AuthGuardService**.

1. Código typescript

```

import { NgModule } from '@angular/core';
import { RouterModule, Routes } from
'@angular/router';
import { LoginComponent } from
'./pages/login/login.component';
import { HomeComponent } from
'./pages/home/home.component';
import { AuthGuardService } from './guards/login/auth-
guard.service';

/**
 * Criar rotas para que seja reconhecida pelos
 templates.html
 */
const routes: Routes = [
    {path:'login', component:LoginComponent},
    {path:'home', component:HomeComponent,
    //A propriedade abaixo deve ser inserida em toda
 rota com acesso autenticado
    canActivate:[AuthGuardService]},

```

```
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```


4. Referências:

1. Importe **RouterModule** e **Routes** em seu módulo de roteamento.
2. Matriz de objetos **Routes**
3. API > @angular/router > CanActivate
4. .


5. 

11. 


4. REFERÊNCIAS GLOBAIS

1. Curso Angular #63: Rotas: Tela de Login e como não mostrar o Menu (NavBar)
2. Curso Angular #64: Usando Guarda de Rotas: CanActivate
3. #
4. #
5. #
6. 


5. HISTÓRICO

1. 30/06/2021
 - ☒ Criar este documento baseado no modelo03.md ;
 - ☒ Escrever tópico Objetivos;
 - ☒ Escrever tópico Pre-requisitos
 - ☒ Escrever tópico Benefícios
 - ☒ Escrever tópico Conteúdo
 - ☒ Escrever tópico Exemplos
 - ☒ Escrever tópico Referências
 - 


2. 01/07/2021

- ☒ Escrever os passo 01 a 08
- 

3. 02/06/2021

- ☒ Passo 09 - Ocultar a barra de menu quando o usuário não tiver autenticado
- ☒ Passo 10 - Usando guarda de rotas para bloquear página html do usuário não autenticado
- ☒ Escrever resumo do documento
- 

4. 03/06/2021

- ☐ Testar os passos 01 a 06 para saber se é este documento pe util.
- ☐ Testar os passos 07 a 09 para saber se é este documento pe util.
- ☐ Testar os passos 106 para saber se é este documento pe util.
- ☐ Atualizar o histórico deste documento.
- 

5. .

