- Info
- Source
- Template
- Styles
- DOM Tree

File

`src/app/components/login/login.component.ts`

Description

Component for login user

Implements

`OnInit`

Metadata

selector	`app-login`
styleUrls	`./login.component.scss`
templateUrl	`./login.component.html`

Index

Properties

- Public datosCorrectos
- Public hide
- Public LoginForm
- Public message
- Public token
- Public user
- Public UserLog

Methods

- getErrorMessage
- ngOnInit
- signIn

Constructor

```
constructor(formBuilder: FormBuilder, _userService: UserService, router: Router)
```

Defined in src/app/components/login/login.component.ts:45

Constructor in which we inject our services and different elements

Parameters :

Name	Type	Optional
formBuilder	`FormBuilder`	No
_userService	`UserService`	No
router	`Router`	No

Methods

getErrorMessage

`getErrorMessage(dato)`

Defined in src/app/components/login/login.component.ts:85

function to control error messages

Parameters :

Name	Optional
dato	No

Returns : `"This information is required" | "You must enter at least 6 characters" | "The maximum of charact...`

message

ngOnInit

`ngOnInit()`

Defined in src/app/components/login/login.component.ts:61

Start when the component inits

Returns : `void`

signIn

`signIn()`

Defined in src/app/components/login/login.component.ts:106

SignIn function

Returns : `void`

Properties

Public datosCorrectos

Type : `boolean`

Defined in src/app/components/login/login.component.ts:34

variable to show or hide message if there is an error in login

Public hide

Default value : `true`

Defined in src/app/components/login/login.component.ts:45

Public LoginForm

Type : `FormGroup`

Defined in src/app/components/login/login.component.ts:22

Login form for FormGroup

Public message

Type : `string`

Defined in src/app/components/login/login.component.ts:30

variable for form message

Public token
Defined in src/app/components/login/login.component.ts:42

to save the token

Public user
Type : `User`
Defined in src/app/components/login/login.component.ts:26

Variable to save user in it

Public UserLog
Defined in src/app/components/login/login.component.ts:38

userLog variable

```
import { Component, OnInit } from '@angular/core'
import { FormBuilder } from '@angular/forms'
import { Validators } from '@angular/forms'
import { FormGroup, FormControl, AbstractControl } from '@angular/forms'
import { User } from '../../models/user'
import { UserService } from 'src/app/services/user.service'
import { Router } from '@angular/router'

/**
 * Component for login user
 */
@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.scss']
})

export class LoginComponent implements OnInit {
  /**
   * Login form for FormGroup
   */
  public LoginForm: FormGroup
  /**
   * Variable to save user in it
   */
  public user: User
  /**
   * variable for form message
   */
  public message: string
  /**
   * variable to show or hide message if there is an error in login
   */
  public datosCorrectos: boolean
  /**
   * userLog variable
   */
  public UserLog
  /**
   * to save the token
   */
  public token


  public hide = true;

  /**
   * Constructor in which we inject our services and different elements
   */
  constructor(
    private formBuilder: FormBuilder,
    private _userService: UserService,
    private router: Router
  ) {
    this.user = new User(0, '', '', '', '', '', new Date(0), '', false)
  }

  /**
   * Start when the component inits
   */
  ngOnInit() {
    this.datosCorrectos = true
    this.LoginForm = this.formBuilder.group({
      email: [
        '',
        [
          Validators.required,
          Validators.email,
          Validators.minLength(6),
          Validators.maxLength(30)
        ]
      ],
      password: [
        '',
        [Validators.required, Validators.minLength(6), Validators.maxLength(30)]
      ]
    })
  }

  /**
   * function to control error messages
   * @param {string} dato
   * @returns message
   */
  getErrorMessage(dato) {
    var result: string
    if (this.LoginForm.controls[dato].hasError('required')) {
      return (result = 'This information is required')
    } else if (this.LoginForm.controls[dato].hasError('minlength')) {
      return (result = 'You must enter at least 6 characters')
    } else if (this.LoginForm.controls[dato].hasError('maxlength')) {
      return (result = 'The maximum of characters is 30')
    } else if (
      this.LoginForm.controls[dato].hasError('email') &&
      dato === 'email'
    ) {
      return (result = 'You have to enter a valid email')
    } else {
      return (result = '')
    }
  }

  /**
   * SignIn function
   */
  signIn() {
    this._userService.LoginUser(this.user).subscribe(
      res => {
        this.UserLog = res['user']
        localStorage.setItem('User', JSON.stringify(this.UserLog))

        // save the user photo
        localStorage.setItem('UserPhoto', this.UserLog.photo)

        console.log('Successfully logged')
        this.router.navigate(['/'])
        this.token = res['token']
        localStorage.setItem('token', this.token)
        console.log(this.token)
      },
      error => {
        this.datosCorrectos = false

        var element = document.getElementById('ErroInfo');
        element.classList.remove('d-none');
        setTimeout('document.getElementById("ErroInfo").classList.add("d-none")', 3500);


        if (error.status === 400) {
          this.message = 'The password is not correct'
          console.log(error.status)
          console.log(this.message)
        } else if (error.status === 402) {
          this.message = 'Username does not exist'
          console.log(error.status)
          console.log(this.message)
        } else {
          console.log(error.status)
          this.message = 'Login failed'
          console.log(this.message)
        }
      }
    )
  }
}

<div id="form">
```

```
  <h1>LOGIN</h1>
  <form [formGroup]="LoginForm" (ngSubmit)="signIn()">
    <div class="fildP">
      <div class="logo">
        <img class="icon" src="../../../assets/img/email.svg" alt="" />
      </div>
      <mat-form-field appearance="legacy">
        <mat-label>Email</mat-label>
        <input
          matInput
          type="email"
          formControlName="email"
          placeholder="Email"
          [(ngModel)]="user.email"
        />
        <mat-error *ngIf="!LoginForm.controls['email'].valid">{{
          getErrorMessage("email")
        }}</mat-error>
      </mat-form-field>
    </div>
    <br />
    <div class="fildP">
      <div class="logo">
        <img class="icon2" src="../../../assets/img/lock.svg" alt="" />
      </div>
      <mat-form-field appearance="legacy">
        <mat-label>Password</mat-label>
        <input
          matInput
          [type]="hide ? 'password' : 'text'"
          formControlName="password"
          placeholder="Password"
          [(ngModel)]="user.password"
        />
        <mat-error *ngIf="!LoginForm.controls['password'].valid">
          {{ getErrorMessage("password") }}</mat-error
        >
      </mat-form-field>
      <mat-icon
        (click)="hide = !hide"
        [attr.aria-label]="'Hide password'"
        [attr.aria-pressed]="hide"
      >
        {{ hide ? "visibility_off" : "visibility" }}</mat-icon
      >
    </div>
    <br />
    <div id="ErroInfo" class="d-none">
      <p class="errorM">{{ this.message }}</p>
    </div>
    <button type="submit" class="UnfollowB">
      <span>LOG IN</span>
    </button>
  </form>
  <p class="NotRegister">
    Not registered?
    <b><a class="registrate" [routerLink]="['/register']">Sign up</a></b>
  </p>
</div>
```

```
./login.component.scss
```

```
#form {
  background-color: #272727;
  border: #00a23d 2px solid;
  box-shadow: 0 0 21px 9px rgba(8, 17, 10, 0.63);
  display: flex;
  flex-direction: column;
  font-family: "B612";
  align-items: center;
  justify-content: center;
  text-align: center;
  margin: 4em 1em 2em 1em;
  padding: 2em 0em 2em 0em;
  height: 40%;
}

// button
$bg: #272727;
$fg: #f3d403;
$border-width: 0.2rem;
$corner-size: 3rem;
$dur: 0.3s;

.followB,
.UnfollowB {
```

```scss
  margin: 0 0.5vw 0 0.5vw;
  outline: none;

  font-family: "B612";
  letter-spacing: 0.02rem;
  cursor: pointer;
  background: transparent;
  border: $border-width solid currentColor;
  padding: 0.5rem 6.5rem;
  font-size: 1rem;
  color: $fg;
  position: relative;
  transition: color $dur;

  &:hover {
    color: #cc4224;
    &::before {
      width: 0;
    }
    &::after {
      height: 0;
    }
  }
  &:active {
    border-width: $border-width / 2;
  }

  span {
    position: relative;
    z-index: 2;
  }
  &::before,
  &::after {
    content: "";
    position: absolute;
    background: $bg;
    z-index: 1;
    transition: all $dur;
  }
  &::before {
    width: calc(100% - #{$corner-size});
    height: calc(101% + #{$border-width * 2});
    top: -$border-width;
    left: 50%;
    transform: translateX(-50%);
  }
  &::after {
    height: calc(100% - #{$corner-size});
    width: calc(101% + #{$border-width * 2});
    left: -$border-width;
    top: 50%;
    transform: translateY(-50%);
  }
}

.NotRegister {
  padding-top: 1.5em;
  color: #00a23d;
}

#form ::ng-deep.mat-form-field-appearance-legacy .mat-form-field-flex {
  background: rgb(223, 221, 221);
}
#form ::ng-deep.mat-form-field-infix {
  width: 12em;
  font-family: "B612";
}

#form ::ng-deep.mat-form-field-label-wrapper {
  left: 1em !important;
}

#form ::ng-deep.mat-input-element {
}

.errorM {
  color: #cc4224;
}

.fildP {
  display: flex;
  flex-direction: row;
}

.icon {
  height: 1.2em;
}
```

```
.icon2 {
  height: 1.5em;
}

.logo {
  background-color: #00a23d;
  height: 2.82em;
  width: 3em;
  display: flex;
  align-items: center;
  justify-content: center;
  text-align: center;
}

.mat-icon {
  padding-left: 1em !important;
  color: white !important;
}
h1 {
  padding-bottom: 1vw;
  color: #cc4224;
  font-family: "Bangers", cursive;
  text-decoration: underline;
  -webkit-text-decoration-color: #00a23d; /* Safari */
  text-decoration-color: #00a23d;
}
@media (min-width: 992px) {
  #form {
    margin: 4em 26% 2em 26%;
  }

  #form::ng-deep.mat-form-field-infix {
    width: 25em;
  }

  .UnfollowB {
    padding: 0.5rem 10.5rem;
  }
}
```

Legend
Html element
Component
Html element with directive

result-matching ""

No results matching ""