File

`src/app/components/modals/change-pass-dialog/change-pass-dialog.component.ts`

Description

Component for change pass

Implements

`OnInit`

Metadata

selector	`app-change-pass-dialog`
styleUrls	`./change-pass-dialog.component.scss`
templateUrl	`./change-pass-dialog.component.html`

Index

Properties

- Public correctdata
- Public data
- Public dialogRef
- Public message
- Public passForm

Methods

- getErrorMessage
- ngOnInit
- openSnackBar
- passwordsShouldMatch
- submit

Constructor

```
constructor(dialogRef: MatDialogRef, formBuilder: FormBuilder, _snackBar: MatSnackBar, _UserService: UserService, data: any)
```

Defined in src/app/components/modals/change-pass-dialog/change-pass-dialog.component.ts:30

Constructor in which we inject our services and different elements

Parameters :

Name	Type	Optional
dialogRef	`MatDialogRef<ChangePassDialogComponent>`	No
formBuilder	`FormBuilder`	No
_snackBar	`MatSnackBar`	No
_UserService	`UserService`	No
data	`any`	No

Methods

getErrorMessage

```
getErrorMessage(dato)
```

Defined in src/app/components/modals/change-pass-dialog/change-pass-dialog.component.ts:66

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/modals/change-pass-dialog/change-pass-dialog.component.ts:47

Start when the component inits

Returns : `void`

openSnackBar

```
openSnackBar(message: string, action: string)
```

Defined in src/app/components/modals/change-pass-dialog/change-pass-dialog.component.ts:137

function to open snackBars

Parameters :

Name	Type	Optional
message	`string`	No
action	`string`	No

Returns : `void`

passwordsShouldMatch

```
passwordsShouldMatch(control: AbstractControl)
```

Defined in src/app/components/modals/change-pass-dialog/change-pass-dialog.component.ts:85

Validation to verify that the passwords match

Parameters :

Name	Type	Optional
control	`AbstractControl`	No

Returns : `{ isError: boolean; }`

submit

```
submit(passForm)
```

Defined in src/app/components/modals/change-pass-dialog/change-pass-dialog.component.ts:105

function to submit form

Parameters :

Name	Optional
passForm	No

Returns : void

Properties

Public correctdata

Type : boolean

Defined in src/app/components/modals/change-pass-dialog/change-pass-dialog.component.ts:30

variable to check if the function was ok

Public data

Type : any

Decorators :

`@Inject(MAT_DIALOG_DATA)`

Defined in src/app/components/modals/change-pass-dialog/change-pass-dialog.component.ts:41

Public dialogRef

Type : MatDialogRef<ChangePassDialogComponent>

Defined in src/app/components/modals/change-pass-dialog/change-pass-dialog.component.ts:36

Public message

Type : string

Defined in src/app/components/modals/change-pass-dialog/change-pass-dialog.component.ts:26

variable to save message info

Public passForm

Type : FormGroup

Defined in src/app/components/modals/change-pass-dialog/change-pass-dialog.component.ts:22

to add FormGroup

```
import { Component, OnInit, Inject } from '@angular/core'
import { MatDialogRef, MAT_DIALOG_DATA } from '@angular/material/dialog'
import { FormBuilder } from '@angular/forms'
import { Validators } from '@angular/forms'
import { FormGroup, FormControl, AbstractControl } from '@angular/forms'
import { MatSnackBar } from '@angular/material'
import { UserService } from 'src/app/services/user.service'

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

export class ChangePassDialogComponent implements OnInit {
  /**
  * to add FormGroup
  */
  public passForm: FormGroup
  /**
   * variable to save message info
   */
  public message: string
  /**
   * variable to check if the function was ok
   */
  public correctdata: boolean

  /**
   * Constructor in which we inject our services and different elements
   */
```

```
constructor(
  public dialogRef: MatDialogRef<ChangePassDialogComponent>,
  private formBuilder: FormBuilder,
  private _snackBar: MatSnackBar,
  private _UserService: UserService,

  @Inject(MAT_DIALOG_DATA) public data: any
) { }

/**
 * Start when the component inits
 */
ngOnInit() {
  this.passForm = this.formBuilder.group({
    oldPassword: [
      '',
      [Validators.required, Validators.minLength(6), Validators.maxLength(30)]
    ],
    password: [
      '',
      [Validators.required, Validators.minLength(6), Validators.maxLength(30)]
    ],
    confirmPass: ['', [Validators.required, this.passwordsShouldMatch]]
  })
}

/**
 * function to control error messages
 * @param {string} dato
 * @returns message
 */
getErrorMessage(dato) {
  var result: string
  if (this.passForm.controls[dato].hasError('required')) {
    return (result = 'This information is required')
  } else if (this.passForm.controls[dato].hasError('minlength')) {
    return (result = 'You must enter at least 6 characters')
  } else if (this.passForm.controls[dato].hasError('maxlength')) {
    return (result = 'The maximum of characters is 30')
  } else if (dato === 'confirmPass') {
    return (result = 'Passwords do not match')
  } else {
    return (result = '')
  }
}

/**
 * Validation to verify that the passwords match
 * @param {any} control
 */
passwordsShouldMatch(control: AbstractControl) {
  if (control && (control.value !== null || control.value !== undefined)) {
    const password2Value = control.value
    const passControl = control.root.get('password')
    if (passControl) {
      const passValue = passControl.value
      if (passValue !== password2Value) {
        return {
          isError: true
        }
      }
    }
  }
  return null
}

/**
 * function to submit form
 * @param {any} passForm
 */
submit(passForm) {
  var passObj = {
    email: this.data.email,
    password: passForm.value.oldPassword,
    newPassword: passForm.value.password
  }
  this._UserService.updatePass(this.data.userId, passObj).subscribe(
    res => {
      this.openSnackBar('YOUR PASSWORD HAS BEEN UPDATED', 'Close')
      this.correctdata = true
      this.dialogRef.close('Close modal!')
    },
    err => {
      this.correctdata = false
      if (err.status === 400) {
        this.message = 'Invalid OLD Password'
        console.log(err.status)
```

```
          console.log(this.message)
        } else {
          console.log(err.status)
          this.message = 'Error changing password'
          console.log(this.message)
        }
      }
    )
  }

  /**
   * function to open snackBars
   *  @param {string} message
   *  @param {string} action
   */
  openSnackBar(message: string, action: string) {
    this._snackBar.open(message, action, {
      duration: 8000,
      panelClass: ['blue-snackbar']
    })
  }
}
```

```
<div id="containerModal">
  <h5>{{ data.alias }}, are you sure you want to change your password?</h5>
  <div id="form">
    <h1>Change information</h1>
    <form [formGroup]="passForm" (ngSubmit)="submit(passForm)">
      <mat-form-field appearance="outline">
        <mat-label> Old Password</mat-label>
        <input
          matInput
          type="password"
          formControlName="oldPassword"
          placeholder="Old Password"
        />
        <mat-error *ngIf="!passForm.controls['oldPassword'].valid">
          {{ getErrorMessage("oldPassword") }}</mat-error
        >
      </mat-form-field>
      <br />
      <mat-form-field appearance="outline">
        <mat-label>New Password</mat-label>
        <input
          matInput
          type="password"
          formControlName="password"
          placeholder="Password"
        />
        <mat-error *ngIf="!passForm.controls['password'].valid">
          {{ getErrorMessage("password") }}</mat-error
        >
      </mat-form-field>
      <br />
      <mat-form-field appearance="outline">
        <mat-label>Confirm new Password</mat-label>
        <input
          matInput
          type="password"
          formControlName="confirmPass"
          placeholder="Password"
        />
        <mat-error
          *ngIf="
            passForm.get('confirmPass').invalid &&
            passForm.get('confirmPass').touched &&
            passForm.get('password').touched
          "
        >
          {{ getErrorMessage("confirmPass") }}</mat-error
        >
      </mat-form-field>
      <br />
      <div *ngIf="!correctdata">
        <p class="mensajeError">{{ this.message }}</p>
      </div>
      <br />
      <div class="buttContainer">
        <button class="cancel" mat-button [mat-dialog-close]="">
          No Thanks
        </button>
        <button
          type="submit"
          class="LoginButton i"
          [disabled]="!passForm.valid"
        >
          Change password
```

```
        </button>
      </div>
    </form>
  </div>
</div>
```

./change-pass-dialog.component.scss

```
#containerModal {
  font-family: "B612";
}

#containerModal h1 {
  color: #00a23d;
  font-size: 1.5rem;
}

#containerModal button {
  background-color: #f3d403;
  border: none;
  border-radius: 10px;
  border: 2.5px solid #00a23d;
  padding: 1.5vw;
  font-size: 0.8rem;
}
.buttContainer {
  display: flex;
  justify-content: space-between;
}

#containerModal button:hover {
  background-color: #cc4224;
}

#containerModal button .cancel {
  background-color: #2a75b3;
}
@media (min-width: 992px) {
  #containerModal h1 {
    font-size: 2.5rem;
  }

  #containerModal button {
    font-size: 1rem;
    padding: 0.5vw;
  }
}
```

Legend
Html element
Component
Html element with directive

result-matching ""

No results matching ""