File

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

Description

Component that brings user profile

Implements

`OnInit`

Metadata

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

Index

Properties

- Public dialog
- Public FollowedUsers
- Public FollowersUsers
- Public identity
- page
- page2
- pageSize
- Public profileForm
- Public textError
- Public user

Methods

- avatarDialog
- changePassDialog
- deleteUserDialog
- getErrorMessage
- getFollowedUsers
- getFollowersUsers
- getInfoUser
- ngOnInit
- openSnackBar
- submit

Constructor

`constructor(_snackBar: MatSnackBar, _adapter: DateAdapter, formBuilder: FormBuilder, _UserService: UserService, dialog: MatDialog)`

Defined in src/app/components/profile/profile.component.ts:50

Constructor in which we inject user service,material module for snackBar, form builder, date adaptar and modals

Parameters :

Name	Type	Optional
_snackBar	`MatSnackBar`	No
_adapter	`DateAdapter<any>`	No
formBuilder	`FormBuilder`	No
_UserService	`UserService`	No
dialog	`MatDialog`	No

Methods

avatarDialog

`avatarDialog()`

Defined in src/app/components/profile/profile.component.ts:217

function to open modal to change avatar

Returns : `void`

changePassDialog

`changePassDialog()`

Defined in src/app/components/profile/profile.component.ts:185

function to open modal to change pass

Returns : `void`

deleteUserDialog

`deleteUserDialog()`

Defined in src/app/components/profile/profile.component.ts:201

function to open modal to delete user

Returns : `void`

getErrorMessage

`getErrorMessage(dato)`

Defined in src/app/components/profile/profile.component.ts:134

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

getFollowedUsers

`getFollowedUsers()`

Defined in src/app/components/profile/profile.component.ts:104

Get users followed by the user

Returns : `void`

getFollowersUsers

`getFollowersUsers()`

Defined in src/app/components/profile/profile.component.ts:118

Get users followers

Returns : void

getInfoUser

`getInfoUser()`

Defined in src/app/components/profile/profile.component.ts:89

Get user info

Returns : void

ngOnInit

`ngOnInit()`

Defined in src/app/components/profile/profile.component.ts:68

Start when the component inits

Returns : void

openSnackBar

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

Defined in src/app/components/profile/profile.component.ts:175

function to open snackBars

Parameters :

Name	Type	Optional
message	string	No
action	string	No

Returns : void

submit

`submit()`

Defined in src/app/components/profile/profile.component.ts:160

submit form function

Returns : void

Properties

Public dialog

Type : `MatDialog`

Defined in src/app/components/profile/profile.component.ts:60

Public FollowedUsers

Type : User[]

Default value : `[]`

Defined in src/app/components/profile/profile.component.ts:46

variable to store info about followed users

Public FollowersUsers

Type : User[]

Default value : `[]`

Defined in src/app/components/profile/profile.component.ts:50

variable to store info about followers

Public identity

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

variable to store user identity

page

Type : number

Default value : `1`

Defined in src/app/components/profile/profile.component.ts:24

page2

Type : `number`

Default value : `1`

Defined in src/app/components/profile/profile.component.ts:25

pageSize

Type : `number`

Default value : `4`

Defined in src/app/components/profile/profile.component.ts:26

Public profileForm

Type : `FormGroup`

Defined in src/app/components/profile/profile.component.ts:38

to add FormGroup

Public textError

Type : `any`

Defined in src/app/components/profile/profile.component.ts:42

variable to save message info

Public user

Type : `User`

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

variable to save user in it

```
import { Component, OnInit } from '@angular/core'
import { User } from '../../models/user'
import { UserService } from 'src/app/services/user.service'
import { FormBuilder } from '@angular/forms'
import { Validators } from '@angular/forms'
import { FormGroup, FormControl, AbstractControl } from '@angular/forms'
import { DateAdapter } from '@angular/material/core'
import { MatSnackBar } from '@angular/material'
import { MatDialog } from '@angular/material/dialog'
import { ChangePassDialogComponent } from '../modals/change-pass-dialog/change-pass-dialog.component'
import { DeleteUserDialogComponent } from '../modals/delete-user-dialog/delete-user-dialog.component'
import { AvatarDialogComponent } from '../modals/avatar-dialog/avatar-dialog.component'

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

export class ProfileComponent implements OnInit {
  page = 1;
  page2 = 1;
  pageSize = 4;
  /**
   * variable to store user identity
   */
  public identity
  /**
   * variable to save user in it
   */
  public user: User
  /**
   * to add FormGroup
   */
  public profileForm: FormGroup
  /**
   * variable to save message info
   */
  public textError: any
  /**
    * variable to store info about followed users
    */
  public FollowedUsers: User[] = []
  /**
```

```
  * variable to store info about followers
  */
public FollowersUsers: User[] = []

/**
 * Constructor in which we inject user service,material module for snackBar, form builder, date adaptar and modals
 */
constructor(
  private _snackBar: MatSnackBar,
  private _adapter: DateAdapter<any>,
  private formBuilder: FormBuilder,
  private _UserService: UserService,
  public dialog: MatDialog
) {
  this.user = new User(0, '', '', '', '', '', new Date(0), '', false)
}

/**
 * Start when the component inits
 */
ngOnInit() {
  this.identity = this._UserService.getIdentity()
  this.getInfoUser()
  this.getFollowersUsers()
  this.getFollowedUsers()

  this.profileForm = this.formBuilder.group({
    alias: ['', [Validators.required, Validators.minLength(3), Validators.maxLength(30)]],
    userName: ['', [Validators.minLength(3), Validators.maxLength(30)]],
    surname: ['', [Validators.minLength(3), Validators.maxLength(30)]],
    dateOfBirth: [this._adapter.setLocale('us')],
    email: [
      '',
      [Validators.email, Validators.required, Validators.minLength(6), Validators.maxLength(30)]
    ]
  })
}

/**
 * Get user info
 */
getInfoUser() {
  window.scrollTo(0, 0);
  this._UserService.getUser(this.identity.id).subscribe(
    res => {
      this.user = res
    },
    error => {
      console.log(error)
    }
  )
}

/**
 * Get users followed by the user
 */
getFollowedUsers() {
  this._UserService.getFollowUsers(this.identity.id).subscribe(
    res => {
      this.FollowedUsers = res
    },
    error => {
      console.log(error)
    }
  )
}

/**
 * Get users followers
 */
getFollowersUsers() {
  this._UserService.getFollowersUsers(this.identity.id).subscribe(
    res => {
      this.FollowersUsers = res
    },
    error => {
      console.log(error)
    }
  )
}

/**
  * function to control error messages
  * @param {string} dato
  * @returns message
  */
getErrorMessage(dato) {
  var result: string
```

```
  if (this.profileForm.controls[dato].hasError('required')) {
    return (result = 'This information is required')
  }
  else if (this.profileForm.controls[dato].hasError('minlength')) {
    if (dato === 'email') {
      return (result = 'You must enter at least 6 characters')
    } else {
      return (result = 'You must enter at least 3 characters')
    }
  } else if (this.profileForm.controls[dato].hasError('maxlength')) {
    return (result = 'The maximum of characters is 30')
  } else if (
    this.profileForm.controls[dato].hasError('email') &&
    dato === 'email'
  ) {
    return (result = 'You have to enter a valid email')
  } else {
    return (result = '')
  }
}

/**
 * submit form function
 */
submit() {
  this._UserService.updateUser(this.identity.id, this.user).subscribe(
    res => {
      this.openSnackBar('YOUR PROFILE HAS BEEN UPDATED', 'Close')
    },
    err => {console.log(err)
      this.openSnackBar('UPDATE HAS FAILED', 'Close')}
  )
}

/**
 * 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']
  })
}

/**
 * function to open modal to change pass
 */
changePassDialog(): void {
  const dialogRef = this.dialog.open(ChangePassDialogComponent, {
    data: {
      userId: this.identity.id,
      alias: this.user.alias,
      email: this.user.email
    }
  })
  dialogRef.afterClosed().subscribe(result => {
    console.log('The dialog was closed')
  })
}

/**
 * function to open modal to delete user
 */
deleteUserDialog(): void {
  const dialogRef = this.dialog.open(DeleteUserDialogComponent, {
    data: {
      userId: this.identity.id,
      alias: this.user.alias,
      email: this.user.email
    }
  })
  dialogRef.afterClosed().subscribe(result => {
    console.log('The dialog was closed')
  })
}

/**
 * function to open modal to change avatar
 */
avatarDialog(): void {
  const dialogRef = this.dialog.open(AvatarDialogComponent, {
    data: {
      userId: this.identity.id,
      userPhoto: this.user.photo
    }
  })
```

```
    dialogRef.afterClosed().subscribe(result => {
      console.log('The dialog was closed')
      this.getInfoUser()
    })
  }
}
```

```
<div class="color1">
  <div class="cardHero">
    <h1>PROFILE</h1>
    <div class="UserInfo">
      <div class="imgUserInfo">
        <img [src]="user.photo" [alt]="user.alias" class="" />
        <br />
        <button
          type="button"
          class="followB"
          mat-raised-button
          (click)="avatarDialog()"
        >
          <span>CHANGE AVATAR</span>
        </button>
      </div>
      <div class="infoCard3">
        <p><span> ALIAS:</span> {{ user.alias }}</p>
        <p><span>EMAIL:</span> {{ user.email }}</p>
        <p><span>NAME:</span> {{ user.userName }}</p>
        <p ><span>DATE OF BIRTH: </span>{{ user.dateOfBirth | date }}</p>
        <p><span>SURNAME:</span> {{ user.surname }}</p>
      </div>
      <div class="tableMat">
        <mat-tab-group>
          <mat-tab label="FOLLOWING">
            <div class="tablaContainer">
              <table *ngIf="FollowedUsers" class="table">
                <tbody>
                  <tr
                    *ngFor="
                      let user of FollowedUsers
                        | slice
                          : (page - 1) * pageSize
                          : (page - 1) * pageSize + pageSize;
                      index as i
                    "
                  >
                    <td class="usu imgUser">
                      <div class="userAvatar2 img">
                        <img
                          class="imgUser"
                          *ngIf="user.photo"
                          src="{{ user.photo }}"
                          alt="Photo a book"
                        />
                      </div>
                    </td>
                    <td
                      class="textName"
                      [routerLink]="['/publicProfile', user.idUsu]"
                    >
                      <p>
                        {{
                          user.alias.length > 20
                            ? (user.alias | slice: 0:20) + "..."
                            : user.alias
                        }}
                      </p>
                    </td>
                  </tr>
                </tbody>
              </table>
            </div>
            <div *ngIf="FollowedUsers" class="pag">
              <ngb-pagination
                [(page)]="page"
                [pageSize]="pageSize"
                [collectionSize]="FollowedUsers.length"
              >
              </ngb-pagination>
            </div>
          </mat-tab>

          <mat-tab label="FOLLOWERS">
            <div class="tablaContainer">
              <table *ngIf="FollowersUsers" class="table">
                <tbody>
                  <tr
                    *ngFor="
                      let user of FollowersUsers
```

```
                    | slice
                      : (page2 - 1) * pageSize
                      : (page2 - 1) * pageSize + pageSize;
                    index as i
                  "
                >
                  <td class="usu imgUser">
                    <div class="userAvatar2 img">
                      <img
                        class="imgUser"
                        *ngIf="user.photo"
                        src="{{ user.photo }}"
                        alt="Photo a book"
                      />
                    </div>
                  </td>
                  <td
                    class="textName"
                    [routerLink]="['/publicProfile', user.idUsu]"
                  >
                    <p>
                      {{
                        user.alias.length > 20
                          ? (user.alias | slice: 0:20) + "..."
                          : user.alias
                      }}
                    </p>
                  </td>
                </tr>
              </tbody>
            </table>
          </div>
          <div *ngIf="FollowersUsers" class="pag">
            <ngb-pagination
              [(page)]="page2"
              [pageSize]="pageSize"
              [collectionSize]="FollowersUsers.length"
            >
            </ngb-pagination>
          </div>
        </mat-tab>
      </mat-tab-group>
    </div>
  </div>
</div>

<div class="cardHero pad">
  <div id="form">
    <h1>Change information</h1>
    <form [formGroup]="profileForm" (ngSubmit)="submit()">
      <mat-form-field appearance="legacy">
        <mat-label>Alias</mat-label>
        <input
          matInput
          type="text"
          formControlName="alias"
          placeholder="Alias"
          [(ngModel)]="user.alias"
        />
        <mat-error *ngIf="!profileForm.controls['alias'].valid">{{
          getErrorMessage("alias")
        }}</mat-error>
      </mat-form-field>
      <br />
      <mat-form-field appearance="legacy">
        <mat-label>Email</mat-label>
        <input
          matInput
          type="email"
          formControlName="email"
          placeholder="Email"
          [(ngModel)]="user.email"
        />
        <mat-error *ngIf="!profileForm.controls['email'].valid">{{
          getErrorMessage("email")
        }}</mat-error>
      </mat-form-field>
      <br />
      <mat-form-field appearance="legacy">
        <mat-label>Name</mat-label>
        <input
          matInput
          type="text"
          formControlName="userName"
          placeholder="userName"
          [(ngModel)]="user.userName"
        />
        <mat-error *ngIf="!profileForm.controls['userName'].valid">{{
```

```
            getErrorMessage("userName")
          }}</mat-error>
        </mat-form-field>
        <br />
        <mat-form-field appearance="legacy">
          <mat-label>Surname</mat-label>
          <input
            matInput
            type="text"
            formControlName="surname"
            placeholder="Surname"
            [(ngModel)]="user.surname"
          />
          <mat-error *ngIf="!profileForm.controls['surname'].valid">{{
            getErrorMessage("surname")
          }}</mat-error>
        </mat-form-field>
        <br />
        <mat-form-field>
          <input
            matInput
            [matDatepicker]="picker1"
            placeholder="Date Of Birth"
            [(ngModel)]="user.dateOfBirth"
            formControlName="dateOfBirth"
          />
          <mat-datepicker-toggle
            matSuffix
            [for]="picker1"
          ></mat-datepicker-toggle>
          <mat-datepicker #picker1></mat-datepicker>
        </mat-form-field>
        <br />
        <button type="submit" class="followB">
          <span>UPDATE PROFILE</span>
        </button>
      </form>
    </div>

    <button type="button" class="followB" (click)="changePassDialog()">
      <span>CHANGE PASSWORD</span>
    </button>

    <button type="button" class="followB2" (click)="deleteUserDialog()">
      <span>DELETE ACCOUNT</span>
    </button>
  </div>
</div>
```

./profile.component.scss

```
.cardHero {
  // border-radius: 2%;
  border: 2.5px solid #00a23d;
  box-shadow: 0 0 21px 9px rgba(8, 17, 10, 0.63);
  padding: 2em;
  background-color: #272727;
  color: white;
}
.cardHero h1 {
  text-align: center;
  padding-bottom: 1vw;
  color: #cc4224;
  font-family: "Bangers", cursive;
  text-decoration: underline;
  -webkit-text-decoration-color: #00a23d; /* Safari */
  text-decoration-color: #00a23d;
}
.UserInfo {
  display: flex;
  font-family: "B612";

  flex-direction: column;
  justify-content: space-around;
  //   padding-bottom: 2em;
  //  // justify-content: center;
  //   align-items: center;
  // height: 20%;
}

.imgUserInfo img {
  border-radius: 2%;
  width: 17em;
  height: 87%;
  box-shadow: 0 0 21px 9px rgba(8, 17, 10, 0.63);
}
.imgUserInfo {
  height: 25em;
```

```
}
.infoCard3 {
  padding-top: 1em;
}

.infoCard3 span {
  color: #00a23d;
}

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

.followB,
.followB2 {
  margin: 1vw 0.5vw 0 0.5vw;
  outline: none;

  font-family: "B612";
  letter-spacing: 0.02rem;
  cursor: pointer;
  background: transparent;
  border: $border-width solid currentColor;
  padding: 0.1rem 0.5rem;
  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;
  }

  //bit lame about the extra span.
  //it's to get the text to appear on top of the psewudo elements. is there a dom-less way to do it?
  span {
    position: relative;
    z-index: 2;
  }
  &::before,
  &::after {
    content: "";
    position: absolute;
    background: $bg;
    z-index: 1;
    transition: all $dur;
  }
  //the 101% is because of a pixel rounding issue in firefox
  &::before {
    width: calc(100% - #{$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%);
  }
}
.followB2 {
  color: #2a75b3;
}
//form
#form {
  display: flex;
  flex-direction: column;
  font-family: "B612";
  align-items: center;
  justify-content: center;
  text-align: center;
  //   width: 30%;
  margin: 0 1em 0em 1em;
```

```
  padding: 0em 0em 2em 0em;
  // height: 40%;
}

.pad {
  margin-top: 2%;
}

::ng-deep.mat-form-field-appearance-legacy.mat-form-field-can-float.mat-form-field-should-float
  .mat-form-field-label {
  color: #00a23d !important;
}

.userAvatar2 {
  height: 4em;
  width: 4em;
  background-repeat: no-repeat;
  background-position: 50%;
  border-radius: 50%;
  background-size: 100% auto;
  overflow: hidden;
  border: #00a23d 2px solid;
  height: auto;
  width: 4.1em;
  height: 4.1em;
}
.imgUser {
  height: auto;
  width: 4.1em;
  height: 4.1em;
}
.table tr {
  color: #00a23d;
}
.table td {
  color: rgb(223, 221, 221);
}

.imgUser img {
  width: 4.1em !important;
  height: 5.1em !important;
}

.textName {
  font-family: "B612";
  padding-top: 30px !important;
  cursor: pointer;
}
.textName:hover p {
  color: #cc4224;
}
::ng-deep.mat-tab-label .mat-tab-label-content {
  font-family: "B612" !important;
  color: #00a23d !important;
}

#form
  ::ng-deep.mat-form-field-appearance-legacy
  .mat-form-field-suffix
  .mat-datepicker-toggle-default-icon {
  width: 6em !important;
  color: white;
}
.pag {
  display: flex;
  align-items: center;
  justify-content: center;
  margin: 1em 0 1em 0;
}
//pagination

ngb-pagination ::ng-deep a {
  color: black !important;
}
ngb-pagination ::ng-deep .page-item.active .page-link {
  color: white !important;
  border-color: black !important;
  background-color: #00a23d;
}
ngb-pagination ::ng-deep .page-item.active .page-link:focus,
.page-link:not(:disabled):not(.disabled):active:focus {
  box-shadow: 0 0 0 0.1rem black;
}
ngb-pagination ::ng-deep .page-link:not(:disabled):not(.disabled):active:focus {
  box-shadow: 0 0 0 0.1rem black;
}

.tableMat {
```

```
  height: 100%;
}

#form ::ng-deep .mat-form-field-appearance-legacy .mat-form-field-label {
  color: #709e82;
}

@media (min-width: 992px) {
  .imgUserInfo {
    height: 31em;
  }

  .imgUserInfo img {
    width: 20em;
    height: 83%;
  }

  .UserInfo {
    flex-direction: row;
  }
  #form {
    margin: 0 26% 0em 26%;
    padding: 2em 0em 2em 0em;
  }

  ::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 ""