File

`src/app/components/modals/coment-hero-dialog/coment-hero-dialog.component.ts`

Description

Component to comment a hero

Implements

`OnInit`

Metadata

selector	`app-coment-hero-dialog`
styleUrls	`./coment-hero-dialog.component.scss`
templateUrl	`./coment-hero-dialog.component.html`

Index

Constructor

```
constructor(dialogRef: MatDialogRef, formBuilder: FormBuilder, _snackBar: MatSnackBar, _UserHeroService:
UserHeroService, data: any)
```

Defined in src/app/components/modals/coment-hero-dialog/coment-hero-dialog.component.ts:31

Constructor in which we inject our services and different elements

Parameters :

Name	Type	Optional
dialogRef	`MatDialogRef<ComentHeroDialogComponent>`	No
formBuilder	`FormBuilder`	No
_snackBar	`MatSnackBar`	No
_UserHeroService	`UserHeroService`	No
data	`any`	No

Methods

deleteComment

`deleteComment()`

Defined in src/app/components/modals/coment-hero-dialog/coment-hero-dialog.component.ts:130

Delete comment

Returns : `void`

getErrorMessage

`getErrorMessage(dato)`

Defined in src/app/components/modals/coment-hero-dialog/coment-hero-dialog.component.ts:65

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 ch...`

message

ngOnInit

`ngOnInit()`

Defined in src/app/components/modals/coment-hero-dialog/coment-hero-dialog.component.ts:47

Start when the component inits

Returns : `void`

openSnackBar

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

Defined in src/app/components/modals/coment-hero-dialog/coment-hero-dialog.component.ts:154

function to open snackBars

Parameters :

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

Returns : `void`

submit

`submit(commentForm)`

Defined in src/app/components/modals/coment-hero-dialog/coment-hero-dialog.component.ts:82

function to submit form

Parameters :

Name	Optional
commentForm	No

Returns : void

Properties

Public commentForm

Type : FormGroup

Defined in src/app/components/modals/coment-hero-dialog/coment-hero-dialog.component.ts:23

to add FormGroup

Public correctdata

Type : boolean

Defined in src/app/components/modals/coment-hero-dialog/coment-hero-dialog.component.ts:31

variable to check if the function was ok

Public data

Type : any

Decorators :

`@Inject(MAT_DIALOG_DATA)`

Defined in src/app/components/modals/coment-hero-dialog/coment-hero-dialog.component.ts:41

Public dialogRef

Type : MatDialogRef<ComentHeroDialogComponent>

Defined in src/app/components/modals/coment-hero-dialog/coment-hero-dialog.component.ts:37

Public message

Type : string

Defined in src/app/components/modals/coment-hero-dialog/coment-hero-dialog.component.ts:27

variable to save message info

```
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'
import { UserHeroService } from 'src/app/services/user-hero.service'

/**
 * Component to comment a hero
 */
@Component({
  selector: 'app-coment-hero-dialog',
  templateUrl: './coment-hero-dialog.component.html',
  styleUrls: ['./coment-hero-dialog.component.scss']
})

export class ComentHeroDialogComponent implements OnInit {
  /**
  * to add FormGroup
  */
  public commentForm: 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<ComentHeroDialogComponent>,
    private formBuilder: FormBuilder,
    private _snackBar: MatSnackBar,
    private _UserHeroService: UserHeroService,
    @Inject(MAT_DIALOG_DATA) public data: any
  ) { }

  /**
   * Start when the component inits
   */
  ngOnInit() {
    this.commentForm = this.formBuilder.group({
      comment: [
        '',
        [
          Validators.required,
          Validators.minLength(6),
          Validators.maxLength(300)
        ]
      ]
    })
  }

  /**
   * function to control error messages
   * @param {string} dato
   * @returns message
   */
  getErrorMessage(dato) {
    var result: string
    if (this.commentForm.controls[dato].hasError('required')) {
      return (result = 'This information is required')
    } else if (this.commentForm.controls[dato].hasError('minlength')) {
      return (result = 'You must enter at least 6 characters')
    } else if (this.commentForm.controls[dato].hasError('maxlength')) {
      return (result = 'The maximum of characters is 300')
    } else {
      return (result = '')
    }
  }

  /**
   * function to submit form
   * @param {any} commentForm
   */
  submit(commentForm) {
    if (this.data.status === 'new') {
      var commentObj = {
        comment: commentForm.value.comment,
        idUsu: this.data.idUsu,
        idHero: this.data.idHero
      }

      this._UserHeroService.commentHero(commentObj).subscribe(
        res => {
          this.openSnackBar('YOUR COMMENT HAS BEEN SAVE', 'Close')
          this.correctdata = true
          this.dialogRef.close('Close modal!')
        },
        err => {
          this.correctdata = false
          console.log(err.status)
          this.message = 'Error saving your comment'
          console.log(this.message)
        }
      )
    } else {
      var commentObj = {
        comment: commentForm.value.comment,
        idUsu: this.data.idUsu,
        idHero: this.data.idHero
      }

      console.log(commentObj)
      this._UserHeroService.commentHero(commentObj).subscribe(
        res => {
          this.openSnackBar('YOUR COMMENT HAS BEEN MODIFY', 'Close')
          this.correctdata = true
          this.dialogRef.close('Close modal!')
        },
        err => {
          this.correctdata = false
          console.log(err.status)
          this.message = 'Error modifin your comment'
          console.log(this.message)
        }
      )
    }
  }

  /**
   * Delete comment
   */
  deleteComment() {
    var commentObj = { idUsu: this.data.idUsu, idHero: this.data.idHero }

    this._UserHeroService.deleteCHero(commentObj).subscribe(
      res => {
        console.log(res)
        this.openSnackBar('YOUR COMMENT HAS BEEN DELETE', 'Close')
        this.correctdata = true
        this.dialogRef.close('Close modal!')
      },
      err => {
        this.correctdata = false
        console.log(err.status)
        this.message = 'Error deleting your commet'
        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">
  <div *ngIf="data.status === 'new' || data.status === 'edit'" id="form">
    <div id="form">
      <h1 *ngIf="data.status === 'new'">Add comment</h1>
      <h1 *ngIf="data.status === 'edit'">Modify commet</h1>
      <form [formGroup]="commentForm" (ngSubmit)="submit(commentForm)">
        <mat-form-field appearance="outline">
          <mat-label>comment</mat-label>
          <input
            *ngIf="!data.comment"
            matInput
            type="text"
            formControlName="comment"
            placeholder="comment"
          />
          <input
            *ngIf="data.comment"
            matInput
            type="text"
            formControlName="comment"
            placeholder="comment"
            placeholder="{{ data.comment }}"
          />
          <mat-error *ngIf="!commentForm.controls['comment'].valid">{{
            getErrorMessage("comment")
          }}</mat-error>
        </mat-form-field>
        <div *ngIf="!correctdata">
          <p class="mensajeError">{{ this.message }}</p>
        </div>
        <div class="buttContainer">
          <button
            type="submit"
            class="LoginButton i"
            [disabled]="!commentForm.valid"
          >
            Send comment
          </button>
          <button class="cancel" mat-button [mat-dialog-close]="">
            No Thanks
          </button>
        </div>
      </form>
    </div>
  </div>
  <div *ngIf="data.status === 'delete'">
    <h1>Delete Comment</h1>
    <h4>Are you sure about deleting your comment</h4>
    <div *ngIf="!correctdata">
      <p class="mensajeError">{{ this.message }}</p>
    </div>
    <div class="buttContainer">
      <button class="cancel" mat-button [mat-dialog-close]="">No Thanks</button>
      <button type="button" mat-raised-button (click)="deleteComment()">
        Delete comment
      </button>
    </div>
  </div>
</div>
```

```
./coment-hero-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 ""