**Додаток Б**

**WEB-ДОДАТОК СИСТЕМА ТЕСТУВАННЯ РІВНЯ КВАЛІФІКАЦІЇ ПРАЦІВНИКА. FRONT-END WEB-ДОДАТКУ**

Текст програми

482.ЧДТУ. 191539.03 12 01

Листів 17

Розробник: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Гаврилюк В. Є.

Черкаси, 2019

**Файл app.module.ts**

2

import {BrowserModule} from '@angular/platform-browser';

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

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

import {HttpClientModule} from '@angular/common/http';

import {NgbModule} from '@ng-bootstrap/ng-bootstrap';

import {FontAwesomeModule} from '@fortawesome/angular-fontawesome';

import {NgHttpLoaderModule} from 'ng-http-loader';

import {ClipboardModule} from 'ngx-clipboard';

import {AppRoutingModule} from './app-routing.module';

import {AppComponent} from './app.component';

import {LoginFormComponent} from './login-form/login-form.component';

import {AppWrapperComponent} from './app-wrapper/app-wrapper.component';

import {NotFoundComponent} from './not-found/not-found.component';

import {AlertCloseableComponent} from './alert-closeable/alert-closeable.component';

import {HomeContentComponent} from './home-content/home-content.component';

import {SurveysListComponent} from './surveys-list/surveys-list.component';

import {CompletedSurveyInfoComponent} from './completed-survey-info/completed-survey-info.component';

import {SurveyAnswerItemComponent} from './survey-answer-item/survey-answer-item.component';

import {CreateNewSurveyComponent} from './create-new-survey/create-new-survey.component';

import {SidebarMenuItemComponent} from './sidebar-menu-item/sidebar-menu-item.component';

import {UserSurveyPassComponent} from './user-survey-pass/user-survey-pass.component';

import {AuthService} from './services/auth.service';

import {AlertService} from './services/alert.service';

import {SurveysService} from './services/surveys.service';

import {UsersService} from './services/users.service';

import {AuthGuard} from './guards/auth.guard';

import {SurveyGuard} from './guards/survey.guard';

import { HomeStatisticItemComponent } from './home-statistic-item/home-statistic-item.component';

@NgModule({

declarations: [

AppComponent,

LoginFormComponent,

AppWrapperComponent,

NotFoundComponent,

AlertCloseableComponent,

HomeContentComponent,

SurveysListComponent,

CompletedSurveyInfoComponent,

SurveyAnswerItemComponent,

CreateNewSurveyComponent,

SidebarMenuItemComponent,

UserSurveyPassComponent,

HomeStatisticItemComponent

],

imports: [

BrowserModule,

AppRoutingModule,

NgbModule,

FontAwesomeModule,

FormsModule,

HttpClientModule,

3

NgHttpLoaderModule.forRoot(),

ReactiveFormsModule,

ClipboardModule

],

providers: [

AuthGuard,

SurveyGuard,

AuthService,

AlertService,

SurveysService,

UsersService

],

bootstrap: [AppComponent]

})

export class AppModule {

}

**Файл app-routing.module.ts**

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

import {RouterModule, Routes} from '@angular/router';

import {LoginFormComponent} from './login-form/login-form.component';

import {AppWrapperComponent} from './app-wrapper/app-wrapper.component';

import {NotFoundComponent} from './not-found/not-found.component';

import {HomeContentComponent} from './home-content/home-content.component';

import {SurveysListComponent} from './surveys-list/surveys-list.component';

import {CompletedSurveyInfoComponent} from './completed-survey-info/completed-survey-info.component';

import {CreateNewSurveyComponent} from './create-new-survey/create-new-survey.component';

import {UserSurveyPassComponent} from './user-survey-pass/user-survey-pass.component';

import {AuthGuard} from './guards/auth.guard';

import {SurveyGuard} from './guards/survey.guard';

const appRouter: Routes = [

{path: '', component: LoginFormComponent},

{

path: 'home', component: AppWrapperComponent, canActivate: [AuthGuard], children: [

{path: '', component: HomeContentComponent},

{path: 'surveys-list', component: SurveysListComponent},

{path: 'survey-info', component: CompletedSurveyInfoComponent},

{path: 'create-survey', component: CreateNewSurveyComponent}

]

},

{path: 'survey', component: UserSurveyPassComponent, canActivate: [SurveyGuard]},

{path: '\*\*', component: NotFoundComponent}

];

@NgModule({

imports: [RouterModule.forRoot(appRouter)],

exports: [RouterModule]

})

export class AppRoutingModule {

}

**Файл app.component.ts**

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

@Component({

4

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: ['./app.component.scss']

})

export class AppComponent { }

**Файл app.component.html**

<app-alert-closeable class="alert-container"></app-alert-closeable>

<ng-http-loader backgroundColor="#009688"></ng-http-loader>

<router-outlet></router-outlet>

**Файл interfaces-list.ts**

export interface ResponseMessage {

status?: number;

message: string;

}

export interface ServerResponseError {

status: number;

error: ResponseMessage;

}

export interface ServerResponseOk {

status: number;

token: string;

username: string;

}

export interface AlertObject {

messageAlert: string;

typeAlert: string;

timeAlert: number;

}

export interface SurveyDetails {

survey\_id: number;

firstname: string;

lastname: string;

status: string;

description: string;

}

export interface SurveyInfo {

question: string;

answers: Array<SurveyInfoAnswerItem>;

}

export interface SurveyInfoAnswerItem {

is\_right: boolean;

answer: string;

}

export interface SurveysDegreesItem {

id: number;

description: string;

}

export interface UserListItem {

id: number;

name: string;

email: string;

}

export interface CreateSurveyResponse {

5

random\_url: string;

survey\_id: number;

}

export interface SurveyQuestionsForPass {

question: string;

answers: Array<SurveyAnswerItemForPass>;

}

export interface SurveyAnswerItemForPass {

id: number;

answer: string;

}

export interface AnswerItemForSend {

surv\_id: number;

answer\_id: number;

full\_answer: string;

}

export interface SurveyStatisticItem {

description: string;

count: number;

}

**Файл auth.guard.ts**

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

import {ActivatedRouteSnapshot, CanActivate, Router, RouterStateSnapshot} from '@angular/router';

import {Observable} from 'rxjs';

import {AuthService} from '../services/auth.service';

import {AlertService} from '../services/alert.service';

import {ResponseMessage, ServerResponseError} from '../interfaces/interfaces-list';

@Injectable()

export class AuthGuard implements CanActivate {

constructor(

private router: Router,

private authService: AuthService,

private alertService: AlertService) {

}

canActivate(route: ActivatedRouteSnapshot, state: RouterStateSnapshot): Observable<boolean> | Promise<boolean> | boolean {

return !!this.isAuthUser();

}

isAuthUser() {

let isAuth: any;

if (localStorage.getItem('tokenSession')) {

isAuth = this.authService.postAuthentication()

.subscribe(

(data: ResponseMessage) => {

return true;

},

(error: ServerResponseError) => {

if (!error.error.message) {

this.router.navigate(['']);

this.alertService.alertSetSubject(

'You are not authorized',

'danger',

error.error.status);

} else {

this.alertService.alertSetSubject(

6

error.error.message,

'danger',

error.status);

}

return false;

}

);

} else {

this.router.navigate(['']);

this.alertService.alertSetSubject(

'You are not authorized',

'danger',

401);

return false;

}

return isAuth;

}

}

**Файл survey.guard.ts**

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

import {ActivatedRouteSnapshot, CanActivate, Router, RouterStateSnapshot} from '@angular/router';

import {Observable} from 'rxjs';

@Injectable()

export class SurveyGuard implements CanActivate {

constructor(private router: Router) {

}

canActivate(route: ActivatedRouteSnapshot, state: RouterStateSnapshot): Observable<boolean> | Promise<boolean> | boolean {

const result = confirm('Are you sure that you want to start testing?');

if (result) {

return true;

} else {

this.router.navigate(['']);

return false;

}

}

}

**Файл alert.service.ts**

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

import {Observable, Subject} from 'rxjs';

import {AlertObject} from '../interfaces/interfaces-list';

@Injectable()

export class AlertService {

public alertSubscription$: Observable<any>;

private alertSubject = new Subject<any>();

constructor() {

this.alertSubscription$ = this.alertSubject.asObservable();

}

alertSetSubject(data: string, type: string, status?: number, time?: number) {

if (status === 0) {

data = 'No internet connection or server error.';

}

const temp: AlertObject = {

7

messageAlert: data,

typeAlert: type,

timeAlert: time

};

this.alertSubject.next(temp);

}

}

**Файл auth.service.ts**

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

import {HttpClient, HttpHeaders} from '@angular/common/http';

@Injectable()

export class AuthService {

constructor(private http: HttpClient) {

}

static createHeaderToken() {

return new HttpHeaders().set('token', localStorage.getItem('tokenSession'));

}

postLogIn(loginData: any) {

return this.http.post(

'https://web-app-practice.herokuapp.com/api/signIn',

{

email: loginData.loginEmail,

password: loginData.loginPassword

}

);

}

postLogOut() {

return this.http.post(

'https://web-app-practice.herokuapp.com/api/logOut',

{},

{headers: AuthService.createHeaderToken()}

);

}

postAuthentication() {

return this.http.post(

'https://web-app-practice.herokuapp.com/api/authentication',

{},

{headers: AuthService.createHeaderToken()}

);

}

}

**Файл surveys.service.ts**

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

import {HttpClient, HttpHeaders, HttpParams} from '@angular/common/http';

import {AnswerItemForSend} from '../interfaces/interfaces-list';

@Injectable()

export class SurveysService {

constructor(private http: HttpClient) {

}

static createHeaderToken() {

return new HttpHeaders().set('token', localStorage.getItem('tokenSession'));

}

8

getSurveysList() {

return this.http.get(

'https://web-app-practice.herokuapp.com/api/surveys',

{headers: SurveysService.createHeaderToken()}

);

}

getSurveyInfo(idSurvey) {

const parameters = new HttpParams().set('survey\_id', idSurvey.toString());

return this.http.get(

'https://web-app-practice.herokuapp.com/api/questionPersonAnswers',

{

headers: SurveysService.createHeaderToken(),

params: parameters

}

);

}

getSurveyDegrees() {

return this.http.get(

'https://web-app-practice.herokuapp.com/api/getDegrees',

{headers: SurveysService.createHeaderToken()}

);

}

getSurveyQuestionsForPass(token: string) {

const parameters = new HttpParams().set('random\_url', token);

return this.http.get(

'https://web-app-practice.herokuapp.com/api/surveyQuestions',

{params: parameters}

);

}

getSurveyStatistic() {

return this.http.get(

'https://web-app-practice.herokuapp.com/api/statistics',

{headers: SurveysService.createHeaderToken()}

);

}

postCreateNewSurvey(createNewSurveyData: any) {

return this.http.post(

'https://web-app-practice.herokuapp.com/api/surveys',

{

person\_id: createNewSurveyData.personId,

degree\_id: createNewSurveyData.degreeId

},

{headers: SurveysService.createHeaderToken()}

);

}

postSendAnswerSurvey(data: Array<AnswerItemForSend>) {

return this.http.post(

'https://web-app-practice.herokuapp.com/api/questionPersonAnswers',

{

answers: data

}

);

}

}

**Файл users.service.ts**

9

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

import {HttpClient, HttpHeaders} from '@angular/common/http';

@Injectable()

export class UsersService {

constructor(private http: HttpClient) {

}

static createHeaderToken() {

return new HttpHeaders().set('token', localStorage.getItem('tokenSession'));

}

getUserList() {

return this.http.get(

'https://web-app-practice.herokuapp.com/api/personsList',

{

headers: UsersService.createHeaderToken()

}

);

}

}

**Файл login-form.component.ts**

import {Component, OnInit} from '@angular/core';

import {Router} from '@angular/router';

import {FormControl, FormGroup, Validators} from '@angular/forms';

import {faSignInAlt, faUnlock, faUserCircle} from '@fortawesome/free-solid-svg-icons';

import {AuthService} from '../services/auth.service';

import {AlertService} from '../services/alert.service';

import {ResponseMessage, ServerResponseError, ServerResponseOk} from '../interfaces/interfaces-list';

@Component({

selector: 'app-login-form',

templateUrl: './login-form.component.html',

styleUrls: ['./login-form.component.scss']

})

export class LoginFormComponent implements OnInit {

public emailIcon = faUserCircle;

public passwordIcon = faUnlock;

public buttonIcon = faSignInAlt;

public errorMessageEmail = 'Please provide a email.';

public errorMessagePassword = 'Please provide a password.';

loginForm: FormGroup = new FormGroup({

loginEmail: new FormControl('', [

Validators.required,

Validators.email

]),

loginPassword: new FormControl('', [

Validators.required

]),

});

constructor(

private router: Router,

private authService: AuthService,

private alertService: AlertService) {

}

ngOnInit() {

if (localStorage.getItem('tokenSession')) {

10

this.authService.postLogOut()

.subscribe(

(data: ResponseMessage) => {

localStorage.removeItem('tokenSession');

localStorage.removeItem('userName');

},

(error: ServerResponseError) => {

error.status === 401 ?

localStorage.removeItem('tokenSession')

: this.alertService.alertSetSubject(error.error.message, 'danger', error.status);

}

);

}

}

validateField(value: string, type: number) {

switch (type) {

case 1: {

value === '' ?

this.errorMessageEmail = 'Please provide a email.'

: this.errorMessageEmail = 'Please provide a valid email.';

break;

}

case 2: {

if (value === '') {

this.errorMessagePassword = 'Please provide a password.';

}

break;

}

default: {

break;

}

}

}

onSubmit() {

this.authService.postLogIn(this.loginForm.value)

.subscribe(

(data: ServerResponseOk) => {

localStorage.setItem('tokenSession', data.token);

localStorage.setItem('userName', data.username);

this.loginForm.reset();

this.router.navigate(['home']);

this.alertService.alertSetSubject(null, null);

},

(error: ServerResponseError) => {

this.alertService.alertSetSubject(error.error.message, 'danger', error.status);

this.loginForm.get('loginPassword').reset();

}

);

}

}

**Файл login-form.component.html**

<div class="container">

<div class="row justify-content-center h-100">

<div class="col-12 col-sm-12 col-md-6 col-lg-4 align-self-center">

11

<div class="card bg-light">

<div class="card-body">

<h5 class="card-title text-center mb-4">Sign in to Test System</h5>

<form (ngSubmit)="onSubmit()" [formGroup]="loginForm" novalidate>

<div class="form-group mb-4">

<div class="input-group mb-2">

<div class="input-group-prepend mr-3">

<span class="input-group-text">

<fa-icon [icon]="emailIcon"></fa-icon>

</span>

</div>

<input (input)="validateField($event.target.value, 1)" autofocus

class="form-control" formControlName="loginEmail" id="loginEmail"

placeholder="Enter email" type="email">

</div>

<small

\*ngIf="loginForm.controls['loginEmail'].invalid && loginForm.controls['loginEmail'].touched"

class="form-text alert alert-danger">{{ errorMessageEmail }}</small>

</div>

<div class="form-group mb-4">

<div class="input-group mb-2">

<div class="input-group-prepend mr-3">

<span class="input-group-text">

<fa-icon [icon]="passwordIcon"></fa-icon>

</span>

</div>

<input (input)="validateField($event.target.value, 2)" class="form-control"

formControlName="loginPassword" id="loginPassword"

placeholder="Enter password" type="password">

</div>

<small

\*ngIf="loginForm.controls['loginPassword'].invalid && loginForm.controls['loginPassword'].touched"

class="form-text alert alert-danger">{{ errorMessagePassword }}</small>

</div>

<div class="text-center">

<button [disabled]="!loginForm.valid" class="btn btn-raised btn-primary" type="submit">

<fa-icon [icon]="buttonIcon" class="mr-2"></fa-icon>

Login

</button>

</div>

</form>

</div>

</div>

</div>

</div>

</div>

**Файл alert-closeable.component.ts**

import {Component, OnInit} from '@angular/core';

import {AlertService} from '../services/alert.service';

12

import {AlertObject} from '../interfaces/interfaces-list';

@Component({

selector: 'app-alert-closeable',

templateUrl: './alert-closeable.component.html',

styleUrls: ['./alert-closeable.component.scss']

})

export class AlertCloseableComponent implements OnInit {

public alertMessage: string;

public typeMessage: string;

public showingAlert = false;

private timerContainer: any;

private timeMessage = 5000;

constructor(private alertService: AlertService) {

this.alertService.alertSubscription$.subscribe((temp: AlertObject) => {

if (!temp.messageAlert && !temp.typeAlert) {

this.closeAlert();

} else {

this.closeAlert();

this.alertMessage = temp.messageAlert;

this.typeMessage = temp.typeAlert;

if (temp.timeAlert) {

this.timeMessage = temp.timeAlert;

}

setTimeout(() => this.showAlert(), 300);

}

});

}

ngOnInit() {

}

showAlert() {

this.showingAlert = true;

this.timerContainer = setTimeout(() => this.closeAlert(), this.timeMessage);

}

closeAlert() {

this.showingAlert = false;

clearTimeout(this.timerContainer);

}

}

**Файл alert-closeable.component.html**

<p class="alert-container">

<ngb-alert (click)="closeAlert()" \*ngIf="showingAlert" [dismissible]="false"

[type]="typeMessage" class="alert-visible">

{{ alertMessage }}

</ngb-alert>

</p>

**Файл app-wrapper.component.ts**

import {Component, OnInit} from '@angular/core';

import {faBars, faSignOutAlt} from '@fortawesome/free-solid-svg-icons';

@Component({

selector: 'app-app-wrapper',

templateUrl: './app-wrapper.component.html',

styleUrls: ['./app-wrapper.component.scss']

})

13

export class AppWrapperComponent implements OnInit {

public sidebarVisible = false;

public burgerIcon = faBars;

public logoutIcon = faSignOutAlt;

public userName: string;

constructor() {

}

ngOnInit() {

this.userName = localStorage.getItem('userName');

}

toggleSidebar() {

this.sidebarVisible = !this.sidebarVisible;

}

stopEventClick(event) {

event.stopPropagation();

}

closeMenu(closeCommand: boolean) {

this.sidebarVisible = closeCommand;

}

}

**Файл app-wrapper.component.html**

<div class="wrapper">

<header class="header bg-primary">

<button (click)="toggleSidebar()" class="btn sidebar-toggler" type="button">

<fa-icon [icon]="burgerIcon" class="fa-icon"></fa-icon>

</button>

<div class="header-content">

<a class="btn header-brand" routerLink="/home">

<img alt="Logo" class="header-logo" src="../../assets/img/logo.png">

Test System

</a>

</div>

</header>

<nav (click)="toggleSidebar()" [class.visible]="sidebarVisible" class="sidebar-wrapper">

<ul (click)="stopEventClick($event)" [class.visible]="sidebarVisible" class="sidebar-nav bg-primary">

<li class="sidebar-block">

<div class="account-info">

<img alt="User icon" class="user-icon" src="../../assets/img/user-icon-128.png">

<div class="nickname">{{ userName }}</div>

</div>

</li>

<app-sidebar-menu-item (changeMenuState)="closeMenu($event)"></app-sidebar-menu-item>

<li class="sidebar-block">

<hr class="separator">

</li>

<li [routerLinkActiveOptions]="{exact:true}" class="sidebar-block" routerLinkActive="active">

<a class="sidebar-link" routerLink="/">

<fa-icon [icon]="logoutIcon" class="fa-icon"></fa-icon>

Logout

</a>

</li>

14

</ul>

</nav>

<main [class.fix-main-content]="sidebarVisible" class="content-wrapper">

<router-outlet></router-outlet>

</main>

</div>

**Файл create-new-survey.component.ts**

import {Component, OnInit} from '@angular/core';

import {FormControl, FormGroup, Validators} from '@angular/forms';

import {Observable} from 'rxjs';

import {debounceTime, map} from 'rxjs/operators';

import {SurveysService} from '../services/surveys.service';

import {UsersService} from '../services/users.service';

import {AlertService} from '../services/alert.service';

import {

CreateSurveyResponse,

ServerResponseError,

SurveysDegreesItem,

UserListItem

} from '../interfaces/interfaces-list';

@Component({

selector: 'app-create-new-survey',

templateUrl: './create-new-survey.component.html',

styleUrls: ['./create-new-survey.component.scss']

})

export class CreateNewSurveyComponent implements OnInit {

public degreesList: Array<SurveysDegreesItem>;

public arrayLinksToCopy: Array<String> = [];

public tooltipMessage: string;

private userList: Array<UserListItem>;

createSurveyForm: FormGroup = new FormGroup({

employeeName: new FormControl('', [

Validators.required,

CreateNewSurveyComponent.selectEmployeeValidator

]),

surveyDegree: new FormControl('', [])

});

constructor(

private surveysService: SurveysService,

private userService: UsersService,

private alertService: AlertService) {

}

static selectEmployeeValidator(control: FormControl): { [s: string]: boolean } {

return control.value && control.value.id ? null : {'employeeName': true};

}

static createLinkSurvey(token: string, id: number) {

return 'http://localhost:4200/survey?token=' + token + '&id=' + id;

}

ngOnInit() {

this.userService.getUserList()

.subscribe(

(data: Array<UserListItem>) => {

this.userList = data;

},

(error: ServerResponseError) => {

this.alertService.alertSetSubject(error.error.message, 'warning');

15

}

);

this.surveysService.getSurveyDegrees()

.subscribe(

(data: Array<SurveysDegreesItem>) => {

this.degreesList = data;

this.createSurveyForm.get('surveyDegree').setValue(data[0].id);

},

(error: ServerResponseError) => {

this.alertService.alertSetSubject(error.error.message, 'warning');

}

);

this.changeTooltipMessage(2);

}

searchEmployee = (text$: Observable<string>) => {

return text$.pipe(

debounceTime(200),

map(term => {

return term === '' ? []

: this.userList.filter(item => {

return item.name.toLowerCase().indexOf(term.toLowerCase()) > -1;

}).slice(0, 10);

})

);

}

formatterResult(item: { name: string }) {

return item.name;

}

changeTooltipMessage(type: number) {

switch (type) {

case 1: {

this.tooltipMessage = 'Copied!';

break;

}

case 2: {

this.tooltipMessage = 'Click to copy to clipboard';

break;

}

default: {

this.tooltipMessage = 'Click to copy to clipboard';

break;

}

}

}

onSubmit() {

const formData = {

personId: parseInt(this.createSurveyForm.get('employeeName').value.id, 10),

degreeId: parseInt(this.createSurveyForm.get('surveyDegree').value, 10)

};

this.surveysService.postCreateNewSurvey(formData)

.subscribe(

(data: CreateSurveyResponse) => {

this.alertService.alertSetSubject('New survey created', 'success');

this.createSurveyForm.get('employeeName').reset();

this.arrayLinksToCopy.push(CreateNewSurveyComponent.createLinkSurvey(data.random\_url, data.survey\_id));

},

(error: ServerResponseError) => {

16

this.alertService.alertSetSubject(error.error.message, 'warning');

}

);

}

}

**Файл create-new-survey.component.html**

<section class="container-fluid">

<div class="col-12 mb-3">

<div class="card">

<h5 class="card-header">Create new survey</h5>

<div class="card-body">

<form (ngSubmit)="onSubmit()" [formGroup]="createSurveyForm" novalidate>

<div class="form-group">

<label for="selectEmployee">Select employee</label>

<input [inputFormatter]="formatterResult" [ngbTypeahead]="searchEmployee"

[resultTemplate]="resultTemplate" class="form-control"

formControlName="employeeName" id="selectEmployee" type="text">

<ng-template #resultTemplate let-r="result" let-t="term">

<ngb-highlight [result]="r.name" [term]="t"></ngb-highlight>

<span class="ml-1">| {{ r.email }}</span>

</ng-template>

<small

\*ngIf="createSurveyForm.controls['employeeName'].invalid && createSurveyForm.controls['employeeName'].touched"

class="form-text alert alert-danger">

Please select a employee.

</small>

</div>

<div class="form-group">

<label for="selectDegrees">Select degree survey</label>

<select class="form-control" formControlName="surveyDegree" id="selectDegrees">

<option \*ngFor="let item of degreesList"

value="{{ item.id }}">{{ item.description }}</option>

</select>

</div>

<button [disabled]="!createSurveyForm.valid" class="btn btn-primary btn-raised w-sm-100"

type="submit">

Create survey

</button>

</form>

</div>

</div>

</div>

<div class="col-12 mb-3">

<div class="card">

<h5 class="card-header">Generated links for survey</h5>

<div class="card-body">

<div \*ngFor="let link of arrayLinksToCopy; let i = index" class="input-group mb-3">

<input #i class="form-control pr-2" type="text" value="{{ link }}">

<div class="input-group-append">

<button (click)="changeTooltipMessage(1)" (mouseout)="changeTooltipMessage(2)"

17

[attr.aria-label]="tooltipMessage" [ngxClipboard]="i"

class="btn btn-outline-info hint-top-left hint-bounce hint-info" type="button">

Copy

</button>

</div>

</div>

</div>

</div>

</div>

</section>