import { Component, OnInit, OnDestroy, AfterViewInit, HostListener,

ViewChild, ElementRef } from '@angular/core';

import { FlashMessagesService } from 'angular2-flash-messages';

import \* as createjs from 'createjs-module';

import { ProjectHandlingService } from '../../services/project-handling.service';

import { LogicService } from '../../services/logic.service';

import { AHPService } from '../../services/ahp.service';

import { ChartService } from '../../services/chart.service';

import { ConstantsService } from '../../services/constants.service';

import { ValidateService } from '../../services/validate.service';

import { AuthenticationService } from '../../services/authentication.service';

@Component({

selector: 'app-principal',

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

styleUrls: ['./principal.component.css']

})

export class PrincipalComponent implements OnInit, OnDestroy, AfterViewInit {

project;

criteriaPairwise = [];

alternativesPairwise = [];

criteriaComparisons = [];

alternativesComparisons = [[]];

criteriaPriorities = [];

goalMatrix = [];

evaluatorsNames = [];

globalCriteriaMatrix = []

goalMatrixReady = false;

globalDecisionVector = [];

stage: any;

diagram: any;

diagramLabels = [];

@ViewChild('diagram') canvas: ElementRef;

constructor(private chartService: ChartService,

private flashMessagesService: FlashMessagesService,

private projectHandlingService:ProjectHandlingService,

private logicService: LogicService,

private ahpService: AHPService,

private constantsService: ConstantsService,

private validateService: ValidateService,

private authenticationService: AuthenticationService) {}

ngOnInit() {

this.project = this.projectHandlingService.getLoadedProject();

this.project.evaluators.forEach(evaluator => {

this.getEvaluatorData(evaluator);

});

if(this.project.imMaster) {

this.criteriaComparisons = this.project.criteriaComparisons;

this.alternativesComparisons = this.project.alternativesComparisons;

for(let i=0; i<this.project.evaluators.length; i++) {

if (this.project.evaluators[i].status) {

let criteriaPriorities = [];

let prioritiesMatrix = [];

let goalMatrix = [];

const matrices = this.ahpService.calculateDecision(

this.project.alternatives.length,

this.project.criteria.length,

this.project.evaluators[i].alternativesComparisons,

this.project.evaluators[i].criteriaComparisons);

prioritiesMatrix = matrices.prioritiesMatrix;

criteriaPriorities = matrices.criteriaPriorities

goalMatrix = this.ahpService.getGoalMatrix(prioritiesMatrix);

setTimeout(\_ => this.chartService.drawStackedChart(document.getElementById('chartDisplay'+i),

this.logicService.prepareStackedChartData(this.project.alternatives, this.project.criteria, prioritiesMatrix),

goalMatrix, 'Evaluación', 'Alternativa', 'Criterio'), 3000);

}

}

}

else {

const userId = JSON.parse(localStorage.getItem('user')).id;

for (let evaluator of this.project.evaluators) {

if(evaluator.\_id == userId){

this.criteriaComparisons = evaluator.criteriaComparisons;

this.alternativesComparisons = evaluator.alternativesComparisons;

if(this.criteriaComparisons.length == 0){

for(let i=0; i<this.project.criteriaComparisons.length; i++) {

this.criteriaComparisons.push(this.constantsService.getSliderInitialvalue());

}

for(let i=0; i<this.project.criteria.length; i++) {

this.alternativesComparisons.push([]);

for(let j=0; j<this.project.alternativesComparisons[i].length; j++)

this.alternativesComparisons[i].push(this.constantsService.getSliderInitialvalue());

}

}

}

}

}

this.stage = new createjs.Stage("diagram");

this.diagram = new createjs.Shape();

this.stage.addChild(this.diagram);

this.onNodeChanges();

}

@HostListener ('window:resize') onResize() {

this.drawDiagram(this.canvas.nativeElement.scrollWidth, this.canvas.nativeElement.scrollHeight);

}

ngAfterViewInit() {

setTimeout(\_ => this.onResize());

}

// Form combinations ---------------------------------------------------------

onNodeChanges() {

this.goalMatrixReady = false;

this.criteriaPairwise = this.logicService.getCombinations(this.project.criteria);

this.alternativesPairwise = this.logicService.getCombinations(this.project.alternatives);

this.drawDiagram(this.canvas.nativeElement.scrollWidth, this.canvas.nativeElement.scrollHeight);

}

onSearchEvaluator(evaluator) {

if(this.validateService.validateEmail(evaluator.email)) {

this.authenticationService.searchUser(evaluator).subscribe(data => {

if(data.success) {

console.log("usuario encontrado");

evaluator.name = data.user.name;

evaluator.\_id = data.user.id

}

else {

console.log("el usuario no existe");

}

});

}

}

getEvaluatorData(evaluator) {

this.authenticationService.getUserData(evaluator).subscribe(data => {

if(data.success) {

console.log("usuario encontrado");

evaluator.name = data.user.name;

evaluator.email = data.user.email;

}

else {

console.log("el usuario no existe");

}

});

}

drawDiagram(width, height) {

const objectiveWidth = this.constantsService.getObjectiveWidth();

const nodewidth = this.constantsService.getNodeWidth();

const nodeHeight = this.constantsService.getNodeHeight();

const nodeSeparation = this.constantsService.getNodeSeparation();

const criteriaPosition = this.constantsService.getCriteriaPosition();

const alternativesPosition = this.constantsService.getAlternativesPosition();

const criteriaOffset = objectiveWidth/2 - (this.project.criteria.length\*nodewidth + (this.project.criteria.length-1)\*(nodeSeparation-nodewidth))/2;

const alternativesOffset = objectiveWidth/2 - (this.project.alternatives.length\*nodewidth + (this.project.alternatives.length-1)\*(nodeSeparation-nodewidth))/2;

const nodeRadius = this.constantsService.getNodeRadius();

let k = 0; // Label index

this.diagram.graphics.clear();

for (let i=0; i<this.diagramLabels.length; i++) {

this.stage.removeChild(this.diagramLabels[i]);

}

// Objective

this.diagram.graphics.f("#C2EA98").rr(0, 0, objectiveWidth, nodeHeight, nodeRadius);

this.diagramLabels[k] = new createjs.Text(this.project.name, "30px Arial bold", "#000");

let bounds = this.diagramLabels[k].getBounds();

let pt = this.diagram.localToLocal((objectiveWidth-bounds.width)/2, (nodeHeight-bounds.height)/2, this.diagramLabels[k])

this.diagramLabels[k].x = pt.x;

this.diagramLabels[k].y = pt.y;

k++;

// Criteria

for (let i=0; i<this.project.criteria.length; i++, k++) {

this.diagram.graphics.f("#FFE74E").rr(criteriaOffset+i\*nodeSeparation, criteriaPosition, nodewidth, nodeHeight, nodeRadius);

this.diagram.graphics.s('black').mt(objectiveWidth/2, nodeHeight).lt(criteriaOffset+i\*nodeSeparation+nodewidth/2, criteriaPosition).es();

this.diagramLabels[k] = new createjs.Text(this.project.criteria[i].name, "18px Arial bold", "#000");

bounds = this.diagramLabels[k].getBounds();

pt = this.diagram.localToLocal(criteriaOffset+i\*nodeSeparation+nodewidth/2-bounds.width/2, criteriaPosition+(nodeHeight-bounds.height)/2, this.diagramLabels[k]);

this.diagramLabels[k].x = pt.x;

this.diagramLabels[k].y = pt.y;

}

// Alternatives

for (let i=0; i<this.project.alternatives.length; i++, k++) {

this.diagram.graphics.f("#6FE9EF").rr(alternativesOffset+i\*nodeSeparation, alternativesPosition, nodewidth, nodeHeight, nodeRadius);

for (let j=0; j<this.project.criteria.length; j++) {

this.diagram.graphics.s('black').mt(criteriaOffset+j\*nodeSeparation+nodewidth/2, criteriaPosition+nodeHeight).lt(alternativesOffset+i\*nodeSeparation+nodewidth/2, alternativesPosition).es();

}

this.diagramLabels[k] = new createjs.Text(this.project.alternatives[i].name, "18px Arial bold", "#000");

bounds = this.diagramLabels[k].getBounds();

pt = this.diagram.localToLocal(alternativesOffset+i\*nodeSeparation+nodewidth/2-bounds.width/2, alternativesPosition+(nodeHeight-bounds.height)/2, this.diagramLabels[k]);

this.diagramLabels[k].x = pt.x;

this.diagramLabels[k].y = pt.y;

}

this.stage.canvas.width = width;

this.stage.canvas.height = height;

const ratio = 100/100; // 100 is the width and height of the circle content.

const windowRatio = width/height;

let scale = width/100;

if (windowRatio > ratio) {

scale = height/100;

}

// Scale up to fit width or height

this.diagram.scaleX = this.diagram.scaleY = scale/5;

for (let i=0; i<k; i++) {

this.stage.addChild(this.diagramLabels[i]);

this.diagramLabels[i].scaleX = this.diagramLabels[i].scaleY = scale/5;

}

this.stage.update();

}

ngOnDestroy() {

this.projectHandlingService.storeProject(this.project);

}

// AHP -----------------------------------------------------------------------

onCalculateDecision() {

let prioritiesMatrix = [];

const matrices = this.ahpService.calculateDecision(

this.project.alternatives.length,

this.project.criteria.length,

this.alternativesComparisons,

this.criteriaComparisons);

prioritiesMatrix = matrices.prioritiesMatrix;

this.criteriaPriorities = matrices.criteriaPriorities

this.goalMatrix = this.ahpService.getGoalMatrix(prioritiesMatrix);

this.goalMatrixReady = true;

window.scrollTo(0,document.body.scrollHeight);

setTimeout(\_ => {

this.chartService.drawNodesChart(document.getElementById('nodesChartDisplay'),

this.logicService.prepareNodesChartData(this.project.name,

this.project.criteria,

this.project.alternatives,

this.criteriaPriorities, prioritiesMatrix));

this.chartService.drawSimpleChart(document.getElementById('criteriaChartDisplay'),

this.logicService.prepareSimpleChartData(this.project.criteria, this.criteriaPriorities),

'Prioridad', 'Criterio');

this.chartService.drawStackedChart(document.getElementById('chartDisplay'),

this.logicService.prepareStackedChartData(this.project.alternatives,

this.project.criteria,

prioritiesMatrix), this.goalMatrix,

'Evaluación', 'Alternativa', 'Criterio');

if(this.project.imMaster) {

const globalMatrices= this.ahpService.calculateGlobalDecision(matrices, this.project.evaluators,

this.project.alternatives.length, this.project.criteria.length);

const globalCriteriaPriorities = globalMatrices.globalCriteriaPriorities

this.evaluatorsNames = this.logicService.getEvaluatorsNames(JSON.parse(localStorage.getItem('user')).name, this.project.evaluators);

this.globalCriteriaMatrix = this.ahpService.getGoalMatrix(globalCriteriaPriorities);

this.chartService.drawStackedChart(document.getElementById('globalCriteriaChartDisplay'),

this.logicService.prepareStackedChartData(this.project.criteria,

this.evaluatorsNames, globalCriteriaPriorities), this.globalCriteriaMatrix,

'Evaluación', 'Criterio', 'Evaluador');

this.chartService.drawSimpleChart(document.getElementById('globalChartDisplay'),

this.logicService.prepareSimpleChartData(this.project.alternatives, globalMatrices.finalVector),

'Decisión Global', 'Alternativa');

this.globalDecisionVector = globalMatrices.finalVector;

}

}, 750);

}

// Project -------------------------------------------------------------------

onSaveProject() {

if(this.project.imMaster) {

this.projectHandlingService.saveProject(this.project).subscribe(data => {

if(data.success) {

this.flashMessagesService.show(data.msg, { cssClass: 'alert-success', timeout: 3000 });

}

else {

this.flashMessagesService.show(data.msg, { cssClass: 'alert-info', timeout: 3000 });

}

});

}

else {

const projectUpdate = {

projectId: this.project.\_id,

evaluatorId: JSON.parse(localStorage.getItem('user')).id,

criteriaComparisons: this.criteriaComparisons,

alternativesComparisons: this.alternativesComparisons

}

this.projectHandlingService.updateProject(projectUpdate).subscribe(data => {

if(data.success) {

this.flashMessagesService.show(data.msg, { cssClass: 'alert-success', timeout: 3000 });

}

else {

this.flashMessagesService.show(data.msg, { cssClass: 'alert-info', timeout: 3000 });

}

});

}

}

}