Приложение А

Код программных модулей

//creating module

angular.module('advertisment',

['ngRoute',

'isteven-multi-select',

'fiestah.money',

'ui.bootstrap.showErrors',

'FormErrors',

'ngTagsInput',

'jkuri.gallery',

'route-change-loader',

'restangular',

'cgBusy',

'infinite-scroll',

'ngRateIt',

'ui-notification',

'ngRateIt',

'ui-rangeSlider'

]);

//configuring modules

//cgBusy configuration

angular.module('advertisment').value('cgBusyDefaults', {

message: 'Loading Stuff',

backdrop: true,

templateUrl: 'Templates/Advertisment/spinnerTemplate.html',

delay: 300,

minDuration: 2000

});

//restangular configuration

angular.module('advertisment').config(function (RestangularProvider) {

RestangularProvider.setBaseUrl('/api/advertisment/');

});

moment.locale("ru");

//controllers

//useful functions for controllers

function convertAdvertismentsDates(advertisments) {

if (Array.isArray(advertisments)) {

advertisments = \_.map(advertisments, convertAdvertismentDate);

} else {

convertAdvertismentDate(advertisments);

}

}

function convertAdvertismentDate(advertisment) {

advertisment.creationDate=convertJSONDate(advertisment.creationDate);

}

function convertJSONDate(jsonDate) {

return moment(jsonDate).fromNow();

}

//конвертирует дату указанному массиву объектов с указанном именем свойства, в котором хранится дата

function convertArrayDates(array, dateProperyName) {

if (Array.isArray(array)) {

array = \_.map(array, createConvertFunction(dateProperyName));

} else {

convertObjectJSONDate(array,dateProperyName);

}

}

function convertObjectJSONDate(object,dateProperyName) {

object[dateProperyName]=moment(object[dateProperyName]).fromNow();

}

function createConvertFunction(dateProperyName) {

return function(elem) {

convertObjectJSONDate(elem, dateProperyName);

}

}

function convertImageForGalery(advertismentImage) {

var result = null;

if (advertismentImage) {

result = {

img: advertismentImage.url,

thumb: advertismentImage.url,

id: advertismentImage.id

};

}

return result;

}

function createImagesArrayForGallery(advertismentImages) {

var result=[];

result = \_.map(advertismentImages, convertImageForGalery);

return result;

}

function convertImageForTransfer(galleryImage) {

return {

url: galleryImage.img,

id: galleryImage.id

}

}

function convertImagesArrayForTransfer(galleryImages) {

var result=[];

result = \_.map(galleryImages, convertImageForTransfer);

return result;

}

function notMySearchesCount(searches) {

var count = 0;

for (var i = 0; i < searches.length; i++) {

if (searches[i].type !== SearchTypeEnum.MyAdvertisments) {

count++;

}

}

return count;

}

function selectSearchCategoriesInMultiselect(selectingCategories,allCategories) {

if (selectingCategories) {

//если пришел не массив

if (Array.isArray(selectingCategories)) {

resetMultiselect(allCategories);

//устанавливаем нужные значения

for (var i = 0; i < allCategories.length; i++) {

if (selectingCategories.some(elem => elem === allCategories[i].name)) {

allCategories[i].ticked = true;

}

}

}

//если пришел не массив, а 1 объект категория

else {

var selectableCategory= allCategories.find(cat => cat.name === selectingCategories);

selectableCategory.ticked = true;

}

}

}

function resetMultiselect(multiselectCategories) {

//ресетаем все значение

for (var j = 0; j < multiselectCategories.length; j++) {

multiselectCategories[j].ticked = false;

}

}

function convertCategoriesToString(categories) {

var result=null ;

if (Array.isArray(categories)) {

result=\_.map(categories, function(n) { return n.name; });

} else {

result=categories.name;

}

return result;

}

function convertCategoriesForAngularMultiselect(categories) {

categories = \_.map(categories, function(category) {

category.iconImage = "<img src='Content/icons/" + category.iconName + ".png'\>";

});

}

function convertAdvertismentTypeToString(type) {

var result = null;

switch(type) {

case AdvertismentTypeEnum.buy:

result= "Покупка";

break;

case AdvertismentTypeEnum.sell:

result= "Продажа";

break;

case AdvertismentTypeEnum.service:

result= "Услуга";

break;

}

return result;

}

function convertArrayAdvertismentTypesToString(array) {

array.forEach(function(elem) {

elem.advertismentType = convertAdvertismentTypeToString(elem.advertismentType);

});

}

//enu, для типа поиска

var SearchTypeEnum = {

Normal: '0',

GetLatest:'1',

MyAdvertisments:'2'

};

//enum для типов объявления

var AdvertismentTypeEnum = {

buy: 0,

sell:1,

service:2

}

//количество элементов на странице

var SEARCH\_COUNT = 10;

//максимальная и минимальаня цены

var MAX\_MIN\_COST\_RANGE = { min: 1, max: 1000 };

//создать Range модель по значениям константы

function RangeModel() {

return { min: MAX\_MIN\_COST\_RANGE.min, max: MAX\_MIN\_COST\_RANGE.max };

}

function SearchFilter(isTypeFilterOn,isCostFilterOn,advertismentType,minCost,maxCost) {

//default values

if (typeof isTypeFilterOn === 'undefined') { isTypeFilterOn = false; }

if (typeof isCostFilterOn === 'undefined') { isCostFilterOn = false; }

if (typeof advertismentType === 'undefined') { advertismentType = null; }

if (typeof minCost === 'undefined') { minCost = MAX\_MIN\_COST\_RANGE.min; }

if (typeof maxCost === 'undefined') { maxCost = MAX\_MIN\_COST\_RANGE.max; }

this.isTypeFilterOn = isTypeFilterOn;

this.isCostFilterOn = isCostFilterOn;

this.advertismentType = advertismentType;

this.minCost = minCost;

this.maxCost = maxCost;

}

//конструктор для объекта 'поиск'

function Search(type, query, categories,searchFilter ) {

this.type = type;

this.query = query;

this.categories = categories;

this.result = null;

this.searchFilter = searchFilter;

this.pageNumber = 1;

this.isDisabled = false;

this.deactivate=function() {

this.isDisabled = true;

};

this.getTitle=function() {

switch(this.type) {

case SearchTypeEnum.Normal:

return this.query;

case SearchTypeEnum.GetLatest:

return 'Свежие объявления';

}

};

this.displayInSidebar=function() {

return !(this.type === SearchTypeEnum.MyAdvertisments);

};

this.isMyAdvertisments=function() {

return this.type === SearchTypeEnum.MyAdvertisments;

};

this.isActive=function() {

return !this.isDisabled;

};

}

//CONTROLLERS

var module = angular.module('advertisment');

module.controller('searchCtrl', function ($scope, $location, Restangular, $routeParams,advertismentService,Notification) {

//INITIALIZING

$scope.searches = [];

$scope.query = '';

$scope.searchPromise = null;

$scope.categoriesPromise= null;

//массив для cg-busy директивы

$scope.promisesArray = [$scope.searchPromise, $scope.categoriesPromise];

$scope.selectedCategories = null;

$scope.allCategories = null;

$scope.selectedSearch = null;

$scope.rangeSliderMaxMin = new RangeModel();

$scope.searchFilter = new SearchFilter();

var stringCategories=null;

var params = $location.search();

var initialSearch;

//эта переменная нужна чтобы кнопка "Получить свежие объявления" не появлялась раньше времени

var initialSearchDone = false;

var searchFilter;

function copySearchFilterObject(object) {

var result;

if (object) {

result= new SearchFilter(object.isTypeFilterOn, object.isCostFilterOn, object.advertismentType, object.minCost, object.maxCost);

}

return result;

}

function checkSearchForEnd(searchObject) {

//если пришло меньше результатов, чем размер страницы- значит это была последняя страница

if ( searchObject.result.length < SEARCH\_COUNT\*searchObject.pageNumber) {

//деактивируем поиск, для того, чтобы он больше не вызывал запросов на сервер, тк все результаты уже получены

searchObject.deactivate();

} else {

searchObject.pageNumber++;

}

}

function checkSearchPagenumber(searchObject,result) {

//если номер страницы 1 - значит это первый поиск по этому запросу

if (searchObject.pageNumber === 1) {

searchObject.result = result;

//добавляем в список поисков

$scope.searches.push(searchObject);

}

//если номер не 1 - значит это повторный поиск, для подтягивания доп данных по запросу, и данные нужно добавить в имеющийся набор

else {

searchObject.result = searchObject.result.concat(result);

}

}

//SCOPE FUNCTIONS

//метод поиска по объекту 'Search'

$scope.searchAdvertisments=function(searchObject,callback) {

if (searchObject) {

var searchInfo = {

query: searchObject.query,

categories: searchObject.categories,

pageNumber: searchObject.pageNumber,

//passing filter object to get

isTypeFilterOn: searchObject.searchFilter.isTypeFilterOn,

isCostFilterOn: searchObject.searchFilter.isCostFilterOn,

minCost: searchObject.searchFilter.minCost,

maxCost: searchObject.searchFilter.maxCost,

advertismentType: searchObject.searchFilter.advertismentType

};

switch (searchObject.type) {

case SearchTypeEnum.Normal:

//запрос на поиск по параметрам

$scope.searchPromise = Restangular.all('search').getList(searchInfo);

break;

case SearchTypeEnum.GetLatest:

//запрос на получение последних объявлений

$scope.searchPromise = Restangular.all('latest').getList(searchInfo);

break;

case SearchTypeEnum.MyAdvertisments:

//запрос на получение пользователей, принадлежащих пользователю, который находится в системе

$scope.searchPromise = Restangular.all('my').getList(searchInfo);

break;

}

//действия после того как придет результат запроса

$scope.searchPromise.then(function(result) {

//если не принес результатов

if ((result.length === 0)&&(searchObject.pageNumber===1)) {

Notification.warning('поиск не принёс результатов ');

} else {

//конвертируем дату из строки в объект Date

convertArrayDates(result, "creationDate");

convertArrayAdvertismentTypesToString(result);

checkSearchPagenumber(searchObject, result);

checkSearchForEnd(searchObject);

//вызываем callback если он ёсть

if (callback) callback();

//выбрать данный поиск как активный

$scope.selectSearch(searchObject);

}

//очистка строки поиска

$scope.query = '';

});

}

};

//действия при нажатии на кнопку поиск

$scope.searchClick = function () {

//меняем маршрут

$location.path("/search");

//очищение массива поисков

stringCategories = convertCategoriesToString($scope.selectedCategories);

//создание объекта поиска

searchFilter =copySearchFilterObject($scope.searchFilter);

var search = new Search(SearchTypeEnum.Normal, $scope.query, stringCategories,searchFilter);

//поиск

$scope.searchAdvertisments(search,$scope.deleteSelectedSearch);

//изменение параметров запроса

};

$scope.openConcreteAdvertisment=function(id) {

$location.path('/concrete/' + id);

}

$scope.applyFiltersForSelectedSearchClick=function() {

stringCategories = convertCategoriesToString($scope.selectedCategories);

searchFilter = copySearchFilterObject($scope.searchFilter);

var search = new Search($scope.selectedSearch.type, $scope.selectedSearch.query, stringCategories, searchFilter);

$scope.searchAdvertisments(search,$scope.deleteSelectedSearch);

}

$scope.addSearchClick = function () {

//поиск

stringCategories = convertCategoriesToString($scope.selectedCategories);

//создание объекта поиска

searchFilter = copySearchFilterObject($scope.searchFilter);

var search = new Search(SearchTypeEnum.Normal, $scope.query, stringCategories,searchFilter);

//поиск

$scope.searchAdvertisments(search);

//изменение параметров запроса

};

$scope.myAdvertismentsClick=function() {

if ($scope.searches.some(search => search.type === SearchTypeEnum.MyAdvertisments)) {

var mySearch = $scope.searches.find(s => s.type === SearchTypeEnum.MyAdvertisments);

$scope.selectSearch(mySearch);

}

else {

searchFilter = copySearchFilterObject($scope.searchFilter);

var searchObj = new Search(SearchTypeEnum.MyAdvertisments,null,null,searchFilter);

$scope.searchAdvertisments(searchObj);

}

};

$scope.reloadMyAdvertisments=function() {

var mySearch = $scope.searches.find(s => s.type === SearchTypeEnum.MyAdvertisments);

$scope.deleteSearch(mySearch);

searchFilter = copySearchFilterObject($scope.searchFilter);

var searchObj = new Search(SearchTypeEnum.MyAdvertisments,null,null,searchFilter);

$scope.searchAdvertisments(searchObj);

};

$scope.resetFiltersClick=function() {

$scope.searchFilter = new SearchFilter();

}

$scope.getLatestAdvertismentsButtonClick=function() {

$scope.searchAdvertisments(new Search(SearchTypeEnum.GetLatest, null, null,new SearchFilter()));

}

$scope.deleteSearchButtonClick = function (deletingSearch) {

var index = $scope.searches.indexOf(deletingSearch);

$scope.deleteSearch(deletingSearch);

//если удаляется выбранный в данный момент поиск-нужно убрать результаты

if (deletingSearch === $scope.selectedSearch) {

if ($scope.searches.length > 0) {

if (index === 0) {

$scope.selectSearch($scope.searches[index]);

} else {

$scope.selectSearch($scope.searches[index - 1]);

}

} else {

$scope.selectedSearch = null;

$location.url($location.path());

resetMultiselect($scope.allCategories);

}

}

};

//устанавливает оображаемый поиск

$scope.selectSearch=function(selectingSearch) {

$scope.selectedSearch = selectingSearch;

//установить фильтр

$scope.searchFilter = copySearchFilterObject(selectingSearch.searchFilter);

//если это оыбчный поиск по параметрам- устанавливаем нужные значения в мультиселект и запихиваем параметры поиск в url

//если есть категории поиска-установить их в мультиселекте

selectSearchCategoriesInMultiselect(selectingSearch.categories, $scope.allCategories);

//установление в параметры url параметров запроса

advertismentService.changeSearchLocation(selectingSearch);

};

//удалить поиск

$scope.deleteSearch=function(search) {

if (search) {

var index = $scope.searches.indexOf(search);

$scope.searches.splice(index, 1);

}

}

$scope.deleteSelectedSearch= function() {

$scope.deleteSearch($scope.selectedSearch);

}

//условия активности

$scope.addSearchDisabled = function () {

return ($scope.searchDisabled())||(notMySearchesCount($scope.searches)===0);

};

$scope.searchDisabled = function () {

return !($scope.query.length > 0);

};

$scope.showSidebarSearchHeading = function() {

return notMySearchesCount($scope.searches) > 0;

}

$scope.applyFiltersButtonDisabled=function() {

return $scope.selectedSearch===null;

}

$scope.getLatestAdvertismentsButtonDisabled=function() {

return $scope.searches.some(search => search.type === SearchTypeEnum.GetLatest)||(!initialSearchDone);

}

// COMMON ACTIVITY, действия, которые нужно выполнить сразу как загружается контроллер

if (($location.path() === '') || ($location.path() === '/search')) {

//проверка параметров запроса, и заполнение объекта "Search" в зависимости от параметров

searchFilter = new SearchFilter(params.isTypeFilterOn, params.isCostFilterOn, params.advertismentType, parseInt(params.minCost), parseInt(params.maxCost));

if (params.type) {

//если также указаны категории, то ищем еще и по ним, если нет- только по запросу поиска без категорий

if (params.categories) {

//нужно чтобы из параметров приходил массив каегорий, но если категория одна- то приходит не массив а просто один объект, тогда переделываем в массив с одним элементом

if (!Array.isArray(params.categories))params.categories = [params.categories];

}

initialSearch = new Search(params.type, params.query, params.categories, searchFilter);

}

//если параметров нет- подгружаем свежие объявления

else {

//получение свежих объявлений при загрузке страницы

initialSearch = new Search(SearchTypeEnum.GetLatest, null, null, new SearchFilter());

}

//поиск по объекту 'Search'

$scope.searchAdvertisments(initialSearch, function() { initialSearchDone = true; });

} else {

initialSearchDone = true;

}

//получение списка категорий с сервера

$scope.categoriesPromise = Restangular.all('categories').getList();

$scope.categoriesPromise.then(function (result) {

convertCategoriesForAngularMultiselect(result);

$scope.allCategories = result;

});

});

module.controller('crudCtrl', function ($scope, $routeParams, $location,Restangular,Notification) {

//INITIALIZATION

$scope.advertisment = null;

$scope.galleryImages = [];

$scope.galleryMainImage = null;

$scope.addImageUrl = '';

$scope.crudPromise = null;

$scope.crudCategories = null;

$scope.crudSelectedCategory = null;

$scope.selectedOption = null;

$scope.selectOptions = [

{

text: 'Покупка',

value: AdvertismentTypeEnum.buy

},

{

text: 'Продажа',

value: AdvertismentTypeEnum.sell

},

{

text: 'Услуга',

value: AdvertismentTypeEnum.service

} ];

//SCOPE FUNCTIONS

//функция, срабатывающая при нажатии на кнопку 'отмена'

$scope.cancel = function () {

$location.path('');

};

//NOT IMPLEMENTED

$scope.checkImageUrlForValid=function()

{

return true;

};

$scope.addImage = function () {

//если данного url нету -то добавить

if ($scope.galleryImages.every(image => image.img !== $scope.addImageUrl)) {

var addImage = {

img: $scope.addImageUrl,

thumb: $scope.addImageUrl

};

$scope.galleryImages.push(addImage);

} else {

Notification.warning('данное изображение уже есть');

}

//очищаем строку ввода url для добавления

$scope.addImageUrl = '';

};

$scope.deleteImage = function (image) {

var index = $scope.galeryImages.indexOf(image);

$scope.galeryImages.splice(index, 1);

};

//здесь будет сохранять изменения на сервер

$scope.save = function () {

//подготовить объект к отправке

//нужно подготовить картинки к отправке

$scope.advertisment.images = convertImagesArrayForTransfer($scope.galleryImages);

//костыль который делает первый image в списке главным если главный не был установлен

if ($scope.advertisment.images.length > 0) {

if ($scope.galleryMainImage) {

$scope.advertisment.mainImage = convertImageForTransfer($scope.galleryMainImage);

} else {

$scope.advertisment.mainImage = $scope.advertisment.images[0];

}

}

$scope.advertisment.advertismentType = $scope.selectedOption.value;

$scope.advertisment.category = $scope.crudSelectedCategory[0];

//послать post запрос

$scope.crudPromise = Restangular.all('save').post($scope.advertisment);

$scope.crudPromise.then(function() {

Notification.success('Сохранено');

$scope.reloadMyAdvertisments();

});

};

//здесь будет удалять с сервера

$scope.delete = function () {

$scope.crudPromise = Restangular.all('delete/'+$scope.advertisment.id).remove();

$scope.crudPromise.then(function() {

Notification.success('Удалено');

$scope.reloadMyAdvertisments();

});

};

//COMMON ACTIVITY , то, что делается при загрузке контроллера

//если есть id то взять данные с сервера

if ($routeParams.id !== undefined) {

//получение данных с REST сервиса

$scope.crudPromise = Restangular.one('getForEdit', $routeParams.id).get();

$scope.crudPromise.then(function(advertisment) {

//parse Date string

convertArrayDates(advertisment, "creationDate");

//создание данных для галереи

$scope.galleryImages = createImagesArrayForGallery(advertisment.images);

$scope.galleryMainImage = convertImageForGalery(advertisment.mainImage);

//занесение данных в scope

$scope.advertisment = advertisment;

//устанавливаем модель, которая будет использоваться в списке типов объявления

$scope.selectedOption = $scope.selectOptions.find(opt => opt.value == advertisment.advertismentType);

//получаем список всех категорий

$scope.crudCategoriesPromise = Restangular.all('categories').getList();

//делаем так, чтобы была выбрана категория, к которой относится объявление

$scope.crudCategoriesPromise.then(function(result) {

convertCategoriesForAngularMultiselect(result);

$scope.crudCategories = result;

selectSearchCategoriesInMultiselect(convertCategoriesToString($scope.advertisment.category),$scope.crudCategories);

});

});

} else {

$scope.crudCategoriesPromise = Restangular.all('categories').getList();

//делаем так, чтобы была выбрана категория, к которой относится объявление

$scope.crudCategoriesPromise.then(function (result) {

convertCategoriesForAngularMultiselect(result);

$scope.crudCategories = result;

});

}

});

module.controller('concreteCtrl', function($scope, $routeParams, Restangular,Notification) {

//INITIALIZATION

$scope.advertisment = null;

$scope.galleryImages = [];

$scope.stars = null;

$scope.commentText = null;

$scope.canRate = null;

$scope.canComment = null;

//promises

$scope.advertismentPromise = null;

$scope.getCanRatePromise = null;

$scope.postCommentPromise = null;

$scope.postRatePromise = null;

$scope.getCommentsPromise = null;

var MAX\_STARS\_COUNT = 5;

//FUNCTIONS

function sendCommend(commentText, advertismentId) {

$scope.postCommentPromise = Restangular.all("addComment").post({

commentText: commentText,

advertismentId: advertismentId

});

$scope.postCommentPromise.then(function(addedComment) {

convertArrayDates(addedComment, "date");

$scope.advertisment.comments.unshift(addedComment);

Notification.success('Комментарий оставлен');

$scope.commentText = '';

angular.element('#close-review-box').triggerHandler('click');

}, function() {

Notification.error("Ошибка сохранения");

});

}

function rateAdvertisment() {

if (($scope.canRate)&&($scope.stars)) {

$scope.postRatePromise = Restangular.all("rateAdvertisment").post({

value: $scope.stars,

advertismentId: $scope.advertisment.id

});

$scope.postRatePromise.then(function(newStatistics) {

$scope.advertisment.statistics=newStatistics;

Notification.success('Оценка выставлена');

$scope.canRate = false;

$scope.stars = 0;

}, function() {

Notification.error("Ошибка при оценке");

});

}

}

//COMMON ACTIVITY , то, что делается при загрузке контроллера

//если есть параметры запроса то тянем с web api объявление с таким id

if ($routeParams.id !== undefined) {

//получение данных о объявлении

$scope.advertismentPromise = Restangular.one('advertisment', $routeParams.id).get();

$scope.advertismentPromise.then(function(advertisment) {

//converting dates from json string

advertisment.creationDate = moment(advertisment.creationDate).format('MMMM Do YYYY, h:mm:ss a');

convertArrayDates(advertisment.comments, "date");

//создание данных для галереи

$scope.galleryImages = createImagesArrayForGallery(advertisment.images);

$scope.advertisment = advertisment;

$scope.advertisment.stringAdvertismentType=convertAdvertismentTypeToString($scope.advertisment.advertismentType);

// SCOPE METHODS that depends on advertisment object

$scope.getAverageCount=function () {

var result;

if (!Number.isInteger) {

result = $scope.advertisment.statistics.averageMark.toFixed(1);

} else {

result = $scope.advertisment.statistics.averageMark;

}

return result;

}

$scope.getShiningStarsCount = function() {

return new Array(Math.floor($scope.advertisment.statistics.averageMark));

}

$scope.getEmptyStarsCount = function() {

return new Array(MAX\_STARS\_COUNT - Math.floor($scope.advertisment.statistics.averageMark));

}

$scope.getMarkPercentage = function(mark) {

var ratingPercentage = $scope.advertisment.statistics.ratingPercentages.find(value => value.mark === mark);

return ratingPercentage.percent.toString().substr(0, 4);

}

//посылаем запрос чтобы узнать может ли данный пользователь оценивать объявление

$scope.getCanRatePromise = Restangular.one("canRate",$scope.advertisment.id).get();

$scope.getCanRatePromise.then(function(boolResult) {

$scope.canRate = boolResult;

});

});

}

$scope.SendCommentClick=function() {

sendCommend($scope.commentText, $scope.advertisment.id);

}

$scope.rateClick=function() {

rateAdvertisment();

}

});

Класс AdvertismentRepository

using GS.DAL;

using GS.DAL.Models;

using System;

using System.Collections.Generic;

using System.Data.Entity;

using System.Data.Entity.Infrastructure;

using System.Linq;

using GS.DAL.DTOs;

using AutoMapper;

namespace GS.Core

{

public static class AdvertismentRepository

{

static int searchCount = 10;

public static void Configure()

{

Mapper.CreateMap<Image, ImageDTO>();

Mapper.CreateMap<Tag, TagDTO>();

Mapper.CreateMap<Comment, CommentDTO>()

.ForMember(dest => dest.User,

opts => opts.MapFrom(src => src.User.UserName));

Mapper.CreateMap<Category, CategoryDTO>();

Mapper.CreateMap<AdvertismentRatingPercentage, AdvertismentRatingPercentageDTO>();

Mapper.CreateMap<AdvertismentMark, AdvertismentMarkDTO>();

Mapper.CreateMap<AdvertismentStatistics, AdvertismentStatisticsDTO>();

Mapper.CreateMap<Advertisment, AdvertismentBasicDTO>();

Mapper.CreateMap<Advertisment, AdvertismentForViewDTO>();

Mapper.CreateMap<Advertisment, AdvertismentForEditDTO>();

}

public static void UpdateAdvertismentEntity(Advertisment value, string userId)

{

using (GSDbContext context = new GSDbContext())

{

Advertisment changingAdvertisment = AdvertismentRepository.GetAdertismentForEdit(value.Id, userId, context);

changingAdvertisment.Description = value.Description;

changingAdvertisment.Cost = value.Cost;

changingAdvertisment.Title = value.Title;

changingAdvertisment.AdvertismentType = value.AdvertismentType;

//обновление категории

UpdateAdvertismentCategory(value, changingAdvertisment, context);

//обновление тегов

UpdateAdvertismentTags(value, changingAdvertisment, context);

//обновление imageв

UpdateAdvertismentImages(value, changingAdvertisment);

//обновить MainImage

UpdateAdvertismentMainImage(value, changingAdvertisment);

context.SaveChanges();

}

}

private static void UpdateAdvertismentMainImage(Advertisment value, Advertisment changingAdvertisment)

{

if (value.MainImage != null)

{

changingAdvertisment.MainImage = value.MainImage;

}

}

private static void UpdateAdvertismentImages(Advertisment value, Advertisment changingAdvertisment)

{

foreach (Image image in value.Images)

{

//если картинка новая- добавляем

if (image.Id == 0)

{

//если появился новый image

changingAdvertisment.Images.Add(image);

}

else

{

//если image с таким id уже был, но его свойства изменили

changingAdvertisment.Images.First(i => i.Id == image.Id).Url = image.Url;

}

}

//удаляем из коллекции картинки, которых нету в ответном запросе

changingAdvertisment.Images.RemoveAll(image => value.Images.All(i => i.Id != image.Id));

}

private static void UpdateAdvertismentTags(Advertisment value, Advertisment changingAdvertisment,

GSDbContext context)

{

ReplaceTagsIfTheyExist(value, context);

//добавляем в рабочую коллекцию новые теги, которые пришли из запроса

foreach (Tag tag in value.Tags)

{

if (changingAdvertisment.Tags.All(t => t.Name != tag.Name))

{

tag.Name = tag.Name.ToLower();

changingAdvertisment.Tags.Add(tag);

}

}

//удаляем из коллекции теги, которых не осталось в запросе

changingAdvertisment.Tags.RemoveAll(tag => value.Tags.All(t => t.Name != tag.Name));

}

private static void UpdateAdvertismentCategory(Advertisment value, Advertisment changingAdvertisment,

GSDbContext context)

{

ReplaceCategoryObjectOnExisting(value, context);

changingAdvertisment.Category = value.Category;

}

private static void ReplaceTagsIfTheyExist(Advertisment value, GSDbContext context)

{

if (value.Tags != null)

{

for (int i = 0; i < value.Tags.Count; i++)

{

string name = value.Tags[i].Name;

if (context.Tags.Any(tag => tag.Name.Equals( name, StringComparison.OrdinalIgnoreCase)))

{

value.Tags[i] = context.Tags.First(tag => tag.Name.Equals(name, StringComparison.OrdinalIgnoreCase));

}

}

}

}

private static void ReplaceCategoryObjectOnExisting(Advertisment value, GSDbContext context)

{

if (value.Category != null)

{

Category category = context.Categories.First(cat => cat.Name == value.Category.Name);

value.Category = category;

}

}

public static List<AdvertismentBasicDTO> GetUserAdvertisments(string userId, int pageNumber,List<String> categories, AdvertismentSearchFilterDTO searchFilter)

{

using (GSDbContext context = new GSDbContext())

{

var query =

Queryable.OrderByDescending<Advertisment, DateTime>(

(from advertisment in context.Advertisments

where advertisment.User.Id == userId

select advertisment), adv => adv.CreationDate)

.Skip(searchCount\*(pageNumber - 1))

.Take(searchCount);

//категории

query = ApplyCategoriesToSearchQuery(categories, query);

//фильтры

query = ApplyFiltersToSearchQuery(searchFilter, query);

List<AdvertismentBasicDTO> result = Mapper.Map<List<AdvertismentBasicDTO>>(query.ToList());

return result;

}

}

private static bool AdvertismentExist(int id, GSDbContext context)

{

return Queryable.Any<Advertisment>(context.Advertisments, adv => adv.Id == id);

}

public static Advertisment GetAdertismentById(int id)

{

using (GSDbContext context = new GSDbContext())

{

if (AdvertismentRepository.AdvertismentExist(id, context))

{

Advertisment result;

result = context.Advertisments.First(adv => adv.Id == id);

return result;

}

else

{

throw new ArgumentException();

}

}

}

public static Advertisment GetAdertismentById(int id, GSDbContext context)

{

if (AdvertismentRepository.AdvertismentExist(id, context))

{

Advertisment result;

result = context.Advertisments.First(adv => adv.Id == id);

return result;

}

else

{

throw new ArgumentException();

}

}

public static AdvertismentForViewDTO GetAdvertismentForView(int id)

{

using (GSDbContext context = new GSDbContext())

{

Advertisment advertisment = GetAdertismentById(id, context);

IncrementViewNumber(advertisment);

context.SaveChanges();

advertisment.Comments= advertisment.Comments.OrderByDescending(com => com.Date).ToList();

AdvertismentForViewDTO result = Mapper.Map<AdvertismentForViewDTO>(advertisment);

return result;

}

}

public static AdvertismentForEditDTO GetAdertismentForEdit(int id, string userId)

{

using (GSDbContext context = new GSDbContext())

{

if (IsAdvertismentOwner(id, userId, context))

{

Advertisment advertisment = GetAdertismentById(id, context);

AdvertismentForEditDTO result = Mapper.Map<AdvertismentForEditDTO>(advertisment);

return result;

}

else

{

throw new ArgumentException();

}

}

}

public static Advertisment GetAdertismentForEdit(int id, string userId, GSDbContext context)

{

if (IsAdvertismentOwner(id, userId, context))

{

return GetAdertismentById(id, context);

}

else

{

throw new ArgumentException();

}

}

private static bool IsAdvertismentOwner(int advertismentId, string userId)

{

using (GSDbContext context = new GSDbContext())

{

return

Queryable.First<Advertisment>((context.Advertisments),

adv => adv.Id == advertismentId).User.Id ==

userId;

}

}

private static bool IsAdvertismentOwner(int advertismentId, string userId, GSDbContext context)

{

return GetAdertismentById(advertismentId,context).User.Id == userId;

}

public static void AddAdvertisment(Advertisment value, string userId)

{

using (GSDbContext context = new GSDbContext())

{

//меняем на существующие в базе значения

ReplaceTagsIfTheyExist(value, context);

ReplaceCategoryObjectOnExisting(value, context);

value.User = context.AspNetUsers.First(usr => usr.Id == userId);

value.CreationDate = DateTime.Now;

value.Statistics=new AdvertismentStatistics();

context.Entry(value).State=EntityState.Added;

try

{

context.SaveChanges();

}

catch (DbUpdateException dbUpdateException)

{

}

}

}

public static void DeleteAdvertisment(int id, string userId)

{

try

{

using (GSDbContext context = new GSDbContext())

{

Advertisment deletingAdvertisment = GetAdertismentForEdit(id, userId, context);

context.Entry(deletingAdvertisment).State= EntityState.Deleted;

context.SaveChanges();

}

}

catch (Exception exception)

{

throw new ArgumentException();

}

}

public static List<AdvertismentBasicDTO> SearchAdvertisments(string query, List<string> category, int pageNumber,

AdvertismentSearchFilterDTO searchFilter)

{

using (GSDbContext context = new GSDbContext())

{

//парсим запрос пробелами

string[] splittedQuery = query.Split(' ');

List<Advertisment> list;

var selectQuery = (from advertisment in context.Advertisments

//проверяем содержит ли хоть один тег к объявлению значение запроса

where advertisment.Tags.Any(tag => splittedQuery.Any(quer => tag.Name.ToLower().Contains(quer.ToLower())))

select advertisment

).OrderByDescending(adv => adv.CreationDate)

.Skip(searchCount\*(pageNumber - 1))

.Take(searchCount);

//если есть категории поиска- добавляем проверку на категории

selectQuery = ApplyCategoriesToSearchQuery(category, selectQuery);

//проверка на выполнение фильтров

selectQuery = ApplyFiltersToSearchQuery(searchFilter, selectQuery);

List<AdvertismentBasicDTO> result = Mapper.Map<List<AdvertismentBasicDTO>>(selectQuery.ToList());

return result;

}

}

private static IQueryable<Advertisment> ApplyCategoriesToSearchQuery(List<string> category, IQueryable<Advertisment> selectQuery)

{

if (category.Count > 0)

{

selectQuery =

selectQuery.Where(

advertisment => category.Any(categoryName => categoryName == advertisment.Category.Name));

}

return selectQuery;

}

private static IQueryable<Advertisment> ApplyFiltersToSearchQuery(AdvertismentSearchFilterDTO searchFilter, IQueryable<Advertisment> selectQuery)

{

//если проверка цены включена

if (searchFilter.IsCostFilterOn)

{

selectQuery =

selectQuery.Where(

advertisment =>

(advertisment.Cost <= searchFilter.MaxCost) &&

(advertisment.Cost >= searchFilter.MinCost));

}

//если проверка типа включена

if (searchFilter.IsTypeFilterOn)

{

selectQuery =

selectQuery.Where(advertisment => advertisment.AdvertismentType == searchFilter.AdvertismentType);

}

return selectQuery;

}

public static List<CategoryDTO> GetCategories()

{

using (GSDbContext context = new GSDbContext())

{

List<Category> categories;

categories = (from category in context.Categories

select category).ToList();

List<CategoryDTO> result = Mapper.Map<List<CategoryDTO>>(categories);

return result;

}

}

public static List<AdvertismentBasicDTO> GetLatestAdvertisments(int pageNumber,List<string> category,AdvertismentSearchFilterDTO filter)

{

using (GSDbContext context = new GSDbContext())

{

var query = (from advertisment in context.Advertisments

select advertisment

).OrderByDescending(adv => adv.CreationDate)

.Skip(searchCount \* (pageNumber - 1))

.Take(searchCount);

query = ApplyCategoriesToSearchQuery(category,query);

query= ApplyFiltersToSearchQuery(filter,query);

List<AdvertismentBasicDTO> result = Mapper.Map<List<AdvertismentBasicDTO>>(query.ToList());

return result;

}

}

public static CommentDTO AddComment(string commentText, string userId, int advertismentId)

{

using (GSDbContext context = new GSDbContext())

{

CommentDTO result= new CommentDTO();

Comment addComment=new Comment();

addComment.Text = commentText;

addComment.Date = DateTime.Now;

addComment.User = context.AspNetUsers.First(user => user.Id == userId);

Advertisment advertisment = GetAdertismentById(advertismentId, context);

advertisment.Comments.Add(addComment);

context.SaveChanges();

result = Mapper.Map<CommentDTO>(addComment);

return result;

}

}

public static void DeleteComment(int commentId, string userId)

{

using (GSDbContext context = new GSDbContext())

{

if (IsCommentOwner(commentId, userId, context))

{

Comment deletingComment = context.Comments.First(com => com.Id == commentId);

context.Comments.Remove(deletingComment);

}

else

{

throw new Exception("You're not comment owner");

}

}

}

public static bool IsCommentOwner(int commentId, string userId, GSDbContext context)

{

if (CommentExist(commentId, context))

{

return context.Comments.First(com => com.Id == commentId).User.Id == userId;

}

else

{

throw new Exception("такого комментария не существует");

}

}

public static bool CommentExist(int commentId, GSDbContext context)

{

return context.Comments.Any(com => com.Id == commentId);

}

//добавляет новую оценку объявлению

public static AdvertismentStatisticsDTO RateAdvertisment(int value, string userId, int advertismentId)

{

using (GSDbContext context = new GSDbContext())

{

AdvertismentStatisticsDTO result;

if (AdvertismentExist(advertismentId, context))

{

AdvertismentMark mark = new AdvertismentMark();

mark.User = context.AspNetUsers.First(usr => usr.Id == userId);

mark.Value = value;

Advertisment advertisment = GetAdertismentById(advertismentId,context);

advertisment.Statistics.Marks.Add(mark);

context.SaveChanges();

result = Mapper.Map<AdvertismentStatisticsDTO>(advertisment.Statistics);

return result;

}

else

{

throw new Exception("такого объявления не существует");

}

}

}

public static bool CanRateAdvertisment(string userId,int advertismentId)

{

using (GSDbContext context = new GSDbContext())

{

Advertisment advertisment = GetAdertismentById(advertismentId, context);

return (userId!=null)&&(advertisment.Statistics.Marks.All(mark => mark.User.Id != userId))&&(advertisment.User.Id != userId);

}

}

//увеличить колилчество просмотров

private static void IncrementViewNumber(Advertisment advertisment)

{

advertisment.Statistics.Views++;

}

public static List<CommentDTO> GetAdvertismentComments(int advertismentId)

{

using (GSDbContext context = new GSDbContext())

{

List<CommentDTO> result;

Advertisment advertisment = GetAdertismentById(advertismentId, context);

var query = (from comment in advertisment.Comments

select comment)

.OrderByDescending(com => com.Date);

result = Mapper.Map<List<CommentDTO>>(query.ToList());

return result;

}

}

}

}