----------------------------TIME TABLE----application layer-----------------------------------------------

***Představení použitých entit***

MODEL: CityEntity

using System.ComponentModel.DataAnnotations;

namespace MVCTimetable.Models

{

public class CityEntity

{

[Display(Name ="Identifizierungsnummer")]

public int Id { get; set; }

[Display(Name ="Name\_der\_Stadt")]

public string CityName { get; set; }

public CityEntity(int iD,string cityName)

{

Id = iD;

CityName = cityName;

}

}

}

MODEL: ConnectionEntity

using System.ComponentModel.DataAnnotations;

using System;

namespace MVCTimetable.Models

{

public class ConnectionEntity

{

[Display(Name = "Abfahrtsdatum:")]

public DateTime DepartureDate { get; set; }

[Display(Name = "Abfahrtsstadt")]

public int DepartureCityId { get; private set; }

[Display(Name = "Abfahrtszeit")]

public TimeSpan DepartureTime { get; private set; }

[Display(Name = "Anfahrtsstadt")]

public int ArrivalCityId { get; private set; }

[Display(Name ="Anfahrtszeit")]

public TimeSpan ArrivalTime { get; private set; }

[Display(Name = "Identifizierungsnummer")]

public int ID { get; private set; }

public ConnectionEntity(int id, int departureCityId, TimeSpan departureTime, int arrivalCityId, TimeSpan arrivalTime)

{

ID = id;

DepartureCityId = departureCityId;

DepartureTime = departureTime;

ArrivalCityId = arrivalCityId;

ArrivalTime = arrivalTime;

}

public ConnectionEntity(int departureCityId, TimeSpan departureTime, int arrivalCityId, TimeSpan arrivalTime)

{

DepartureCityId = departureCityId;

DepartureTime = departureTime;

ArrivalCityId = arrivalCityId;

ArrivalTime = arrivalTime;

}

}

}

MODEL: FeedbackEntity

using System.ComponentModel.DataAnnotations;

namespace MVCTimetable.Models

{

public class FeedbackEntity

{

[Display(Name = "Identifizierungsnummer")]

public int Id { get; private set; }

[Display(Name = "Anforderungstyp")]

public string RequestType { get; private set; }

[Display(Name = "Bericht")]

[DataType(DataType.MultilineText)]

public string Message { get; private set; }

[Display(Name ="Email")]

[DataType(DataType.EmailAddress)]

public string Email { get; private set; }

[Display(Name = "Handynummer")]

public string PhoneNumber { get; private set; }

public FeedbackEntity(int id,string requestType,string message,string email,string phoneNumber)

{

Id = id;

RequestType = requestType;

Message = message;

Email = email;

PhoneNumber = phoneNumber;

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

***Nabídka funkcionalit adminovi***

CONTROLLER: AdminOptionController

using System.Web.Mvc;

namespace MVCTimetable.Controllers

{

public class AdminOptionController : Controller

{

[Authorize]

public ActionResult Option()

{

return View();

}

}

}

VIEW: AdminOption: Option

@{

ViewBag.Title = "Admin";

}

<h2>Administration</h2><h3>Städte</h3>

<h4> @Html.ActionLink("Eine Stadt einlegen", "AddCity", controllerName: "AdminCities", routeValues:null, htmlAttributes: new { @class="btn btn-default"})</h4>

<h4> @Html.ActionLink("Abbildung eingelegter Städte", "DisplayCities", "AdminCities", routeValues:null, htmlAttributes: new { @class="btn btn-default"})</h4>

<h4> @Html.ActionLink("Eine Stadt entfernen", "DeleteCities", "AdminCities", routeValues:null, htmlAttributes: new { @class="btn btn-default"})</h4>

<h3>Verkehrsverbindugen</h3>

<h4> @Html.ActionLink("Gründung neuer Verkehrsverbindung", "Create", controllerName:"AdminCreate", routeValues: null, htmlAttributes: new { @class = "btn btn-default" }) </h4><br/>

<h4> @Html.ActionLink("Abbildung gespeicherter Verkehrsverbindugen","Display", controllerName:"AdminDisplay", routeValues:null, htmlAttributes: new { @class= "btn btn-default" })</h4><br/>

<h4> @Html.ActionLink("Änderung gegründeter Verkehrsverbindung aufgrund ID", "Update", controllerName:"AdminUpdate", routeValues:null, htmlAttributes:new { @class= "btn btn-default" })</h4><br/>

<h4> @Html.ActionLink("Löschung gegründeter Verkehrsverbindung aufgrund ID","Delete", controllerName: "AdminDelete", routeValues:null, htmlAttributes:new {@class= "btn btn-default" })</h4><br/>

<h3>Berichten aus web Seiten</h3>

<h4> @Html.ActionLink("Abbildung der Berichten von Besuchern", "FeedbackDisplay", controllerName: "AdminFeedbackDisplay", routeValues:null,htmlAttributes:new { @class= "btn btn-default" })</h4><br/>

<h4> @Html.ActionLink("Löschung eines Berichtes vom Besucher aufgrund ID", "FeedbackDelete", controllerName: "AdminFeedbackDelete", routeValues:null, htmlAttributes:new { @class= "btn btn-default" })</h4><br/>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

***Založení, zobrazení a vymazání dat pro produkt adminem (města)***

CONTROLLER: AdminCitiesController

using System.Web.Mvc;

using MVCTimetable.Models;

namespace MVCTimetable.Controllers

{

public class AdminCitiesController : Controller

{

AdminCitiesViewModel adminCitiesViewModel = new AdminCitiesViewModel();

[Authorize]

public ActionResult AddCity()

{

return View(adminCitiesViewModel);

}

[Authorize]

[HttpPost]

public ActionResult AddCity(AdminCitiesViewModel adminCitiesViewModel)

{

if (adminCitiesViewModel.CityName == null)

{

return View(adminCitiesViewModel);

}

adminCitiesViewModel.AddCity(adminCitiesViewModel);

TempData["adminCitiesResult"] = "Anforderung wurde durchgeführt";

return RedirectToAction("Option", "AdminOption");

}

[Authorize]

public ActionResult DisplayCities()

{

adminCitiesViewModel.DisplayCities();

return View(adminCitiesViewModel);

}

[Authorize]

public ActionResult DeleteCities()

{

return View(adminCitiesViewModel);

}

[Authorize]

[HttpPost]

public ActionResult DeleteCities(AdminCitiesViewModel adminCitiesViewModel)

{

if (adminCitiesViewModel.IdCity == null)

{

return View(adminCitiesViewModel);

}

bool found = adminCitiesViewModel.FindCity(adminCitiesViewModel);

if (!found)

{

ModelState.AddModelError(nameof(adminCitiesViewModel.IdCity), $"Die Identifizierungsnummer {adminCitiesViewModel.IdCity} wurde nicht gefunden");

return View(adminCitiesViewModel);

}

return View("DeleteCitiesConfirmation",adminCitiesViewModel);

}

[Authorize]

[HttpPost]

public ActionResult DeleteCityConfirmed(AdminCitiesViewModel adminCitiesViewModel)

{

adminCitiesViewModel.DeleteCity(adminCitiesViewModel);

TempData["adminCitiesResult"] = "Anforderung wurde durchgeführt";

return RedirectToAction("Option", "AdminOption");

}

}

}

MODEL: AdminCitiesViewModel

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.Linq;

using CLTimeTableDB;

namespace MVCTimetable.Models

{

public class AdminCitiesViewModel

{

DbRepository dbRepository = new DbRepository();

CityCache cityCache = new CityCache();

[Display(Name = "Name der Stadt")]

[Required(ErrorMessage = "Geben Sie bitte einen Namen der Stadt an")]

public string CityName { get; set; }

[Display(Name ="Identifizierungsnummer")]

[Required(ErrorMessage = "Geben Sie bite eine Identifizierungsnummer an")]

public int? IdCity { get; set; }

public List<CityEntity> Cities { get; private set; }

public CityEntity CityToDelete { get; private set; }

public void AddCity(AdminCitiesViewModel adminCitiesViewModel)

{

dbRepository.AddCity(adminCitiesViewModel.CityName);

}

public void DisplayCities()

{

if (Cities == null)

Cities = new List<CityEntity>();

Cities.Clear();

Dictionary<int, CityEntityDL> citiesFromDB = cityCache.GetCities();

Cities = citiesFromDB.Select(x => new CityEntity(x.Key, x.Value.CityName)).ToList();

}

private int GetCityId(int? iDCity)

{

if (!IdCity.HasValue)

throw new InvalidOperationException($"{nameof(iDCity)} is empty");

return iDCity.Value;

}

public bool FindCity(AdminCitiesViewModel adminCitiesViewModel)

{

bool found;

int iDCity = GetCityId(adminCitiesViewModel.IdCity);

CityEntityDL cityFromDB = cityCache.GetCityById(iDCity);

if (cityFromDB == null)

return found = false;

CityToDelete = new CityEntity(cityFromDB.Id, cityFromDB.CityName);

return found = true;

}

public void DeleteCity(AdminCitiesViewModel adminCitiesViewModel)

{

int idCity = GetCityId(adminCitiesViewModel.IdCity);

dbRepository.DeleteCity(idCity);

}

}

}

VIEW: AdminCities: AddCity

@model MVCTimetable.Models.AdminCitiesViewModel

@{

ViewBag.Title = "AdminAddCity";

}

<h2>Einlegen einer Stadt</h2>

@using (Html.BeginForm())

{

Html.AntiForgeryToken();

Html.ValidationSummary();

<div class="form-horizontal">

<div class="form-group">

@Html.LabelFor(model => model.CityName, new { @class="control-label col-md-2"})

<div class="col-md-10">

@Html.EditorFor(model => model.CityName, new { @class="form-control"})

@Html.ValidationMessageFor(model => model.CityName,"", new { @class="text-danger"})

</div>

</div>

<div class="form-group">

<div class="col-md-offset-2 col-md-10">

<input type="submit" value="einlegen" class="btn btn-default" />

</div>

</div>

</div>

}

VIEW: AdminCities: DisplayCities

@model MVCTimetable.Models.AdminCitiesViewModel

@{

ViewBag.Title = "AdminDisplayCities";

}

<h2>Abbildung eingelegter Städte</h2>

@using (Html.BeginForm())

{

Html.AntiForgeryToken();

Html.ValidationSummary(true);

foreach (var city in Model.Cities)

{

<div class="row">

<div class="col-md-2">

@Html.LabelFor(model => city.Id, new { @class="control-label"})

@Html.TextBoxFor(model => city.Id, new { @class="form-control", @readonly= "readonly" })

</div>

<div class="col-md-2">

@Html.LabelFor(model => city.CityName, new { @class="control-label"})

@Html.TextBoxFor(model => city.CityName, new { @class="form-control", @readonly="readonly"})

</div>

</div><br/>

}

}

VIEW: AdminCities: DeleteCities

@model MVCTimetable.Models.AdminCitiesViewModel

@{

ViewBag.Title = "AdminDeleteCities";

}

<h2>Abbildung eingelegter Städte aufgrund ID</h2>

@using (Html.BeginForm())

{

Html.AntiForgeryToken();

Html.ValidationSummary(true);

<div class="form-horizontal">

<div class="form-group">

@Html.LabelFor(model => model.IdCity, new { @class="control-label col-md-2"})

<div class="col-md-10">

@Html.EditorFor(model => model.IdCity, new { @class="form-control"})

@Html.ValidationMessageFor(model => model.IdCity,"",new { @class="text-danger"})

</div>

</div><br/>

<div class="form-group">

<div class="col-md-offset-2 col-md-10">

<input type="submit" value="finden" class="btn btn-default" />

</div>

</div>

</div>

}

VIEW: AdminCities: DeleteCitiesConfirmation

@model MVCTimetable.Models.AdminCitiesViewModel

@{

ViewBag.Title = "DeleteCitiesConfirmation";

}

<h2>Abbildung der Stadt im Zwecke seiner Entfernung</h2>

@using (Html.BeginForm("DeleteCityConfirmed","AdminCities",Model))

{

Html.AntiForgeryToken();

Html.ValidationSummary(true);

<div class="row">

<div class="col-md-2">

@Html.LabelFor(model => model.CityToDelete.Id, new { @class="control-label col-md-2"})

@Html.TextBoxFor(model => model.CityToDelete.Id, new { @class="form-control", @readonly= "readonly" })

</div>

<div class="col-md-2">

@Html.LabelFor(model => model.CityToDelete.CityName, new { @class="control-label col-md-2"})

@Html.TextBoxFor(model => model.CityToDelete.CityName, new { @class="form-control", @readonly= "readonly" })

</div><br/>

</div><br/>

<div class="row">

<div class="col-md-1 col-md-10">

<input type="submit" value="entfernen" class="btn btn-default" />

@Html.ActionLink("abbrechen","Option","AdminOption", null, new { @class="btn btn-default"})

</div></div> }

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

***Založení produktu adminem (spojení)***

CONTROLLER: AdminCreateController

using System.Web.Mvc;

using MVCTimetable.Models;

namespace MVCTimetable.Controllers

{

public class AdminCreateController : Controller

{

[Authorize]

public ActionResult Create()

{

AdminCreateViewModel adminCreateViewModel = new AdminCreateViewModel();

return View(adminCreateViewModel);

}

[Authorize]

[HttpPost]

public ActionResult Create(AdminCreateViewModel adminCreateViewModel)

{

if (!ModelState.IsValid)

{

return View(adminCreateViewModel);

}

adminCreateViewModel.CreateConnection(adminCreateViewModel);

TempData["adminResult"] = "Die Verbindung wurde gegründet";

return RedirectToAction("Create");

}

}

}

MODEL: AdminCreateViewModel

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using CLTimeTableDB;

namespace MVCTimetable.Models

{

public class AdminCreateViewModel

{

CityCache cityCache = new CityCache();

[Display(Name = "Abfahrtszeit")]

[Required(ErrorMessage = "Geben Sie bitte die Abfahrtszeit an")]

public TimeSpan? DepartureTime { get; set; }

[Display(Name = "Anfahrtszeit")]

[Required(ErrorMessage = "Geben Sie bitte die Anfahrtszeit an")]

public TimeSpan? ArrivalTime { get; set; }

[Display(Name = "Abfahrtsstadt")]

public int DepartureCityId { get; set; }

[Display(Name = "Anfahrtsstadt")]

public int ArrivalCityId { get; set; }

public List<SelectListForAdmin> Cities { get; private set; }

public AdminCreateViewModel()

{

Cities = new List<SelectListForAdmin>();

foreach (var city in cityCache.GetCities())

{

Cities.Add(new SelectListForAdmin

(

city.Value.CityName,

city.Key

));

}

}

public TimeSpan GetDepartureTime()

{

if (!DepartureTime.HasValue)

throw new InvalidOperationException($"{nameof(DepartureTime)} is empty.");

return DepartureTime.Value;

}

public TimeSpan GetArrivalTime()

{

if (!ArrivalTime.HasValue)

throw new InvalidOperationException($"{nameof(ArrivalTime)} is empty.");

return ArrivalTime.Value;

}

public void CreateConnection(AdminCreateViewModel adminCreate)

{

ConnectionEntityDL connectionToDB = new ConnectionEntityDL(adminCreate.DepartureCityId,

adminCreate.GetDepartureTime(),

adminCreate.ArrivalCityId,

adminCreate.GetArrivalTime()

);

DbRepository dbRepository = new DbRepository();

dbRepository.InsertConnection(connectionToDB);

}

}

public class SelectListForAdmin

{

public string City { get; set; }

public int CityId { get; set; }

public SelectListForAdmin(string city, int cityId)

{

City = city;

CityId = cityId;

}

}

}

VIEW: AdminCreate: Create

@model MVCTimetable.Models.AdminCreateViewModel

@{

ViewBag.Title = "AdminCreate";

}

<h2>Gründung neuer Verkehrsverbindung</h2><br/>

@using (Html.BeginForm())

{

@Html.AntiForgeryToken()

@Html.ValidationSummary(true)

<div class="form-horizontal">

<div class="form-group">

@Html.LabelFor(model => model.DepartureCityId, new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.DropDownListFor(model => model.DepartureCityId, new SelectList(Model.Cities, "CityId", "City"))

@Html.ValidationMessageFor(model => model.DepartureCityId)

</div>

<br /><br />

</div>

<div class="form-group">

@Html.LabelFor(model => model.DepartureTime, new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.DepartureTime, new { @class = "form-control" })

@Html.ValidationMessageFor(model => model.DepartureTime, "", new { @class = "text-danger" })

</div>

<br /><br />

</div>

<div class="form-group">

@Html.LabelFor(model => model.ArrivalCityId, new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.DropDownListFor(model => model.ArrivalCityId, new SelectList(Model.Cities, "CityId", "City"))

@Html.ValidationMessageFor(model => model.ArrivalCityId)

</div>

<br /><br />

</div>

<div class="form-group">

@Html.LabelFor(model => model.ArrivalTime, new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.ArrivalTime, new { @class = "form-control" })

@Html.ValidationMessageFor(model => model.ArrivalTime, "", new { @class = "text-danger" })

</div>

<br /><br />

</div>

<div class="form-group">

<div class="col-md-offset-2 col-md-10">

<input type="submit" value="gründen" class="btn btn-default" />

</div>

</div>

</div>

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

***Zobrazení produktů adminem (spojení)***

CONTROLLER: AdminDisplayController

using System.Web.Mvc;

using MVCTimetable.Models;

namespace MVCTimetable.Controllers

{

public class AdminDisplayController : Controller

{

[Authorize]

public ActionResult Display()

{

AdminDisplayViewModel adminDisplayViewModel = new AdminDisplayViewModel();

return View(adminDisplayViewModel);

}

}

}

MODEL: AdminDisplayViewModel

using System.Collections.Generic;

using System.Linq;

using CLTimeTableDB;

namespace MVCTimetable.Models

{

public class AdminDisplayViewModel

{

DbRepository dbRepository = new DbRepository();

CityCache cityCache = new CityCache();

public List<ConnectionEntity> DisplayConnections { get; private set; }

public AdminDisplayViewModel()

{

DisplayConnections = new List<ConnectionEntity>();

List<ConnectionEntityDL> connectionsFromDB = dbRepository.GetConnections().ToList();

DisplayConnections = connectionsFromDB.Select(x => new ConnectionEntity(x.IdConnection, x.DepartureCityId,x.DepartureTime,x.ArrivalCityId,x.ArrivalTime)).ToList();

}

public string GetCityById(int placeId)

{

CityEntityDL cityEntity=cityCache.GetCityById(placeId);

string city = cityEntity.CityName;

return city;

}

}

}

VIEW: AdminDisplay: Display

@model MVCTimetable.Models.AdminDisplayViewModel

@{

ViewBag.Title = "AdminView";

}

@using CLTimeTableDB;

<h2>Abbildung gespeicherter Verkehrsverbindugen</h2>

@using (Html.BeginForm())

{

@Html.AntiForgeryToken()

@Html.ValidationSummary(true)

foreach (var connection in Model.DisplayConnections)

{

var departureCityName = Model.GetCityById(connection.DepartureCityId);

var arrivalCityName = Model.GetCityById(connection.ArrivalCityId);

<div class="row">

<div class="col-md-2">

@Html.LabelFor(m => connection.ID, new { @class = "control-label" })

@Html.TextBoxFor(m => connection.ID, new { @class = "form-control", @readonly= "readonly" })

</div>

<div class="col-md-2">

@Html.LabelFor(m => connection.DepartureCityId, new { @class = "control-label" })

@Html.TextBoxFor(m => departureCityName, new { @class = "form-control", @readonly = "readonly" })

</div>

<div class="col-md-2">

@Html.LabelFor(m => connection.DepartureTime, new { @class = "control-label" })

@Html.TextBoxFor(m => connection.DepartureTime, @"{0:hh\:mm}",new { @class = "form-control", @readonly = "readonly" })

</div>

<div class="col-md-2">

@Html.LabelFor(m => connection.ArrivalCityId, new { @class = "control-label" })

@Html.TextBoxFor(m => arrivalCityName, new { @class = "form-control", @readonly = "readonly" })

</div>

<div class="col-md-2">

@Html.LabelFor(m => connection.ArrivalTime, new { @class = "control-label" })

@Html.TextBoxFor(m => connection.ArrivalTime, @"{0:hh\:mm}",new { @class = "form-control", @readonly = "readonly" })

</div>

</div>

<br/>

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

***Změna produktů adminem (spojení)***

CONTROLLER: AdminUpdateController

using System.Web.Mvc;

using MVCTimetable.Models;

namespace MVCTimetable.Controllers

{

public class AdminUpdateController : Controller

{

[Authorize]

public ActionResult Update()

{

AdminUpdateViewModel adminUpdateViewModel = new AdminUpdateViewModel();

return View(adminUpdateViewModel);

}

[Authorize]

[HttpPost]

public ActionResult Update(AdminUpdateViewModel adminUpdateViewModel)

{

if (!ModelState.IsValid)

{

return View(adminUpdateViewModel);

}

bool found = adminUpdateViewModel.FindConnection(adminUpdateViewModel.IdConnection);

if (!found)

{

ModelState.AddModelError(nameof(adminUpdateViewModel.IdConnection), $"Die Identifizierungsnummer {adminUpdateViewModel.IdConnection} wurde nicht gefunden");

return View(adminUpdateViewModel);

}

return View("ConnectionForUpdate", adminUpdateViewModel);

}

[Authorize]

[HttpPost]

public ActionResult UpdateConnection(AdminUpdateViewModel adminUpdateViewModel)

{

if (adminUpdateViewModel.UpdateConnection.DepartureTime == null || adminUpdateViewModel.UpdateConnection.ArrivalTime == null)

{

TempData["MistakeAdminUpdate"] = "Neue Angaben wurden nicht ausgefüllt.Versuchen Sie es noch einmal.";

return RedirectToAction("Option","AdminOption");

}

adminUpdateViewModel.ChangeConnection(adminUpdateViewModel);

TempData["adminUpdate"]="Die Verkehresverbindung wurde korrigiert.";

return RedirectToAction("Option", "AdminOption");

}

}

}

MODEL: AdminUpdateViewModel

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using CLTimeTableDB;

namespace MVCTimetable.Models

{

public class AdminUpdateViewModel

{

DbRepository dbRepository = new DbRepository();

CityCache cityCache = new CityCache();

[Display(Name ="Identifizierungsnummer")]

[Required(ErrorMessage ="Geben Sie bitte eine Identifizierungsnummer an")]

public int? IdConnection { get; set; }

public List<SelectListForAdmin> Cities { get; private set; }

public ConnectionEntity DisplayOldConnection { get; private set; }

public ChangedConnection UpdateConnection { get; set; }

public AdminUpdateViewModel()

{

Cities = new List<SelectListForAdmin>();

foreach (var city in cityCache.GetCities())

{

Cities.Add(new SelectListForAdmin

(

city.Value.CityName,

city.Key

));

}

}

public string GetCityById(int cityId)

{

CityEntityDL cityEntity=cityCache.GetCityById(cityId);

string cityName = cityEntity.CityName;

return cityName;

}

private int GetIDConnection(int? idConnection)

{

if (!idConnection.HasValue)

throw new InvalidOperationException($"{nameof(idConnection)} is empty");

return idConnection.Value;

}

public bool FindConnection(int? id)

{

int idConnection = GetIDConnection(id);

bool found;

ConnectionEntityDL connectionFromDB = dbRepository.GetConnectionByID(idConnection);

if (connectionFromDB == null)

{

return found=false;

}

DisplayOldConnection = new ConnectionEntity(idConnection,

connectionFromDB.DepartureCityId,

connectionFromDB.DepartureTime,

connectionFromDB.ArrivalCityId,

connectionFromDB.ArrivalTime

);

return found = true;

}

public void ChangeConnection(AdminUpdateViewModel adminUpdateViewModel)

{

ConnectionEntityDL connectionToDB = new ConnectionEntityDL(GetIDConnection(adminUpdateViewModel.IdConnection),

adminUpdateViewModel.UpdateConnection.DepartureCityId,

UpdateConnection.GetDepartureTime(),

adminUpdateViewModel.UpdateConnection.ArrivalCityId,

UpdateConnection.GetArrivalTime()

);

dbRepository.UpdateConnection(connectionToDB);

}

}

public class ChangedConnection

{

[Display(Name ="Neue Abfahrtszeit")]

[Required(ErrorMessage = "Geben Sie bitte die neue Abfahrtszeit an")]

public TimeSpan? DepartureTime { get; set; }

[Display(Name = "Neue Anfahrtszeit")]

[Required(ErrorMessage = "Geben Sie bitte die neue Anfahrtszeit an")]

public TimeSpan? ArrivalTime { get; set; }

[Display(Name = "Neue Abfahrtsstadt")]

public int DepartureCityId { get; set; }

[Display(Name = "Neue Anfahrtsstadt")]

public int ArrivalCityId { get; set; }

public TimeSpan GetDepartureTime()

{

if (!DepartureTime.HasValue)

throw new InvalidOperationException($"{nameof(DepartureTime)} is empty.");

return DepartureTime.Value;

}

public TimeSpan GetArrivalTime()

{

if (!ArrivalTime.HasValue)

throw new InvalidOperationException($"{nameof(ArrivalTime)} is empty.");

return ArrivalTime.Value;

}

}

}

VIEW: AdminUpdate: Update

@model MVCTimetable.Models.AdminUpdateViewModel

@{

ViewBag.Title = "AdminUpdate";

}

<h2>Änderung gegründeter Verkehrsverbindung aufgrund ID</h2><br/>

@using (Html.BeginForm())

{

@Html.AntiForgeryToken()

@Html.ValidationSummary(true)

<div class="form-horizontal">

<div class="form-group">

@Html.LabelFor(model => model.IdConnection, new { @class = "control-label col-md-2"})

<div class="col-md-10">

@Html.TextBoxFor(model => model.IdConnection, new { @class = "form-control"})

@Html.ValidationMessageFor(model => model.IdConnection,"",new { @class = "text-danger"})

</div>

</div><br/>

<div class="form-group">

<div class="col-md-offset-2 col-md-10"¨>

<input type="submit" value="finden" class="btn btn-default" />

</div>

</div>

</div>

}

VIEW: AdminUpdate: ConnectionForUpdate

@model MVCTimetable.Models.AdminUpdateViewModel

@{

ViewBag.Title = "AdminUpdate";

}

<h2>Abbildung Verkehrsverbindung im Zwecke seiner Änderung</h2><br/>

@using (Html.BeginForm("UpdateConnection", "AdminUpdate",Model))

{

@Html.AntiForgeryToken()

@Html.ValidationSummary(true)

var oldDepartureCityName = Model.GetCityById(Model.DisplayOldConnection.DepartureCityId);

var oldArrivalCityName = Model.GetCityById(Model.DisplayOldConnection.ArrivalCityId);

<div class="row">

<div class="col-md-2">

@Html.LabelFor(model => model.DisplayOldConnection.ID, new { @class = "control-label col-md-2" })

@Html.TextBoxFor(model => model.DisplayOldConnection.ID, new { @class = "form-control", @readonly = "readonly" })

</div>

<div class="col-md-2">

@Html.LabelFor(model => model.DisplayOldConnection.DepartureCityId, new { @class = "control-label col-md-2" })

@Html.TextBoxFor(model => oldDepartureCityName, new { @class = "form-control", @readonly = "readonly" })

</div>

<div class="col-md-2">

@Html.LabelFor(model => model.DisplayOldConnection.DepartureTime, new { @class = "control-label col-md-2" })

@Html.TextBoxFor(model => model.DisplayOldConnection.DepartureTime,@"{0:hh\:mm}" ,new { @class = "form-control", @readonly = "readonly" })

</div>

<div class="col-md-2">

@Html.LabelFor(model => model.DisplayOldConnection.ArrivalCityId, new { @class = "control-label col-md-2" })

@Html.TextBoxFor(model => oldArrivalCityName, new { @class = "form-control", @readonly = "readonly" })

</div>

<div class="col-md-2">

@Html.LabelFor(model => model.DisplayOldConnection.ArrivalTime, new { @class = "control-label col-md-2" })

@Html.TextBoxFor(model => model.DisplayOldConnection.ArrivalTime,@"{0:hh\:mm}" ,new { @class = "form-control", @readonly = "readonly" })

</div>

</div><br/>

<div class="form-horizontal">

<div class="form-group">

@Html.LabelFor(model => model.UpdateConnection.DepartureCityId, new { @class="control-label col-md-2"})

<div class="col-md-10">

@Html.DropDownListFor(model => model.UpdateConnection.DepartureCityId, new SelectList(Model.Cities, "CityId", "City"))

@Html.ValidationMessageFor(model => model.UpdateConnection.DepartureCityId, "", new { @class = "text-danger" })

</div>

</div><br/>

<div class="form-group">

@Html.LabelFor(model => model.UpdateConnection.DepartureTime, new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.UpdateConnection.DepartureTime, new { @class = "form-control" })

@Html.ValidationMessageFor(model => model.UpdateConnection.DepartureTime, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.UpdateConnection.ArrivalCityId, new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.DropDownListFor(model => model.UpdateConnection.ArrivalCityId, new SelectList(Model.Cities, "CityId", "City"))

@Html.ValidationMessageFor(model => model.UpdateConnection.ArrivalCityId, "", new { @class = "text-danger" })

</div>

</div>

<br/>

<div class="form-group">

@Html.LabelFor(model => model.UpdateConnection.ArrivalTime, new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.UpdateConnection.ArrivalTime, new { @class = "form-control" })

@Html.ValidationMessageFor(model => model.UpdateConnection.ArrivalTime)

</div>

</div>

<div class="form-group">

<div class="col-md-offset-2 col-md-10">

<input type="submit" value="ändern" class="btn btn-default" />

</div>

</div></div>}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

***Výmaz produktu adminem (spojení)***

CONTROLLER: AdminDeleteController

using System.Web.Mvc;

using MVCTimetable.Models;

namespace MVCTimetable.Controllers

{

public class AdminDeleteController : Controller

{

[Authorize]

public ActionResult Delete()

{

AdminDeleteViewModel adminDeleteViewModel = new AdminDeleteViewModel();

return View(adminDeleteViewModel);

}

[Authorize]

[HttpPost]

public ActionResult Delete(AdminDeleteViewModel adminDeleteViewModel)

{

if (!ModelState.IsValid)

{

return View(adminDeleteViewModel);

}

bool found = adminDeleteViewModel.FindConnection(adminDeleteViewModel.IdConnection);

if (!found)

{

ModelState.AddModelError(nameof(adminDeleteViewModel.IdConnection),$"Die Identifizierungsnummer {adminDeleteViewModel.IdConnection} wurde nicht gefunden.");

return View(adminDeleteViewModel);

}

return View("ConnectionForDelete",adminDeleteViewModel);

}

[Authorize]

[HttpPost]

public ActionResult DeleteConnection(AdminDeleteViewModel adminDeleteViewModel)

{

adminDeleteViewModel.DeleteConnection(adminDeleteViewModel.IdConnection);

TempData["adminDeleteResult"] = $"Die Verkehrsverbindung {adminDeleteViewModel.IdConnection} wurde gelöscht";

return RedirectToAction("Option","AdminOption");

}

}

}

MODEL: AdminDeleteViewModel

using System;

using System.ComponentModel.DataAnnotations;

using CLTimeTableDB;

namespace MVCTimetable.Models

{

public class AdminDeleteViewModel

{

DbRepository dbRepository = new DbRepository();

CityCache cityCache = new CityCache();

[Display(Name = "Identifizierungsnummer")]

[Required(ErrorMessage = "Geben Sie bitte eine Identifizierungsnummer an")]

public int? IdConnection { get; set; }

public ConnectionEntity ConnectionToDelete { get; private set; }

public string GetCityById(int cityId)

{

CityEntityDL cityEntity = cityCache.GetCityById(cityId);

string cityName = cityEntity.CityName;

return cityName;

}

private int GetIdConnection(int? idConnection)

{

if (!idConnection.HasValue)

throw new InvalidOperationException($"{nameof(idConnection)} is empty.");

return idConnection.Value;

}

public bool FindConnection(int? id)

{

int idConnection = GetIdConnection(id);

bool found;

ConnectionEntityDL connectionFromDB = dbRepository.GetConnectionByID(idConnection);

if (connectionFromDB == null)

{

return found = false;

}

ConnectionToDelete = new ConnectionEntity(connectionFromDB.IdConnection,

connectionFromDB.DepartureCityId,

connectionFromDB.DepartureTime,

connectionFromDB.ArrivalCityId,

connectionFromDB.ArrivalTime

);

return found = true;

}

public void DeleteConnection(int? id)

{

int idConnection = GetIdConnection(id);

dbRepository.DeleteConnection(idConnection);

}

}

}

VIEW: AdminDelete: Delete

@model MVCTimetable.Models.AdminDeleteViewModel

@{

ViewBag.Title = "AdminDelete";

}

<h2>Löschung gegründeter Verkehrsverbindung aufgrund ID</h2><br/>

@using (Html.BeginForm())

{

@Html.AntiForgeryToken()

@Html.ValidationSummary()

<div class="form-horizontal">

<div class="form-group">

@Html.LabelFor(model => model.IdConnection, new { @class="control-label col-md-2"})

<div class="col-md-10">

@Html.EditorFor(model => model.IdConnection, new { @class="form-control"})

@Html.ValidationMessageFor(model => model.IdConnection,"",new { @class="text-danger"})

</div>

</div><br/>

<div class="form-group">

<div class="col-md-offset-2 col-md-10">

<input type="submit" value="finden" class="btn btn-default" />

</div>

</div>

</div>

}

VIEW: AdminDelete: ConnectionForDelete

@model MVCTimetable.Models.AdminDeleteViewModel

@{

ViewBag.Title = "AdminDelete";

}

<h2>Abbildung Verkehrsverbindung im Zwecke seiner Löschung</h2><br/>

@using (Html.BeginForm("DeleteConnection", "AdminDelete", Model))

{

Html.AntiForgeryToken();

Html.ValidationSummary(true);

string departureCityName = Model.GetCityById(Model.ConnectionToDelete.DepartureCityId);

string arrivalCityName = Model.GetCityById(Model.ConnectionToDelete.ArrivalCityId);

<div class="row">

<div class="col-md-2">

@Html.LabelFor(model => model.ConnectionToDelete.ID, new { @class = "control-label col-md-2" })

@Html.TextBoxFor(model => model.ConnectionToDelete.ID, new { @class = "form-control", @readonly = "readonly" })

</div>

<div class="col-md-2">

@Html.LabelFor(model => model.ConnectionToDelete.DepartureCityId, new { @class = "control-label col-md-2" })

@Html.TextBoxFor(model => departureCityName, new { @class = "form-control", @readonly = "readonly" })

</div>

<div class="col-md-2">

@Html.LabelFor(model => model.ConnectionToDelete.DepartureTime, new { @class = "control-label col-md-2" })

@Html.TextBoxFor(model => model.ConnectionToDelete.DepartureTime, @"{0:hh\:mm}", new { @class = "form-control", @readonly = "readonly" })

</div>

<div class="col-md-2">

@Html.LabelFor(model => model.ConnectionToDelete.ArrivalCityId, new { @class = "control-label col-md-2" })

@Html.TextBoxFor(model => arrivalCityName, new { @class = "form-control", @readonly = "readonly" })

</div>

<div class="col-md-2">

@Html.LabelFor(model => model.ConnectionToDelete.ArrivalTime, new { @class = "control-label col-md-2" })

@Html.TextBoxFor(model => model.ConnectionToDelete.ArrivalTime, @"{0:hh\:mm}", new { @class = "form-control", @readonly = "readonly" })

</div>

</div><br/>

<div class="row">

<div class="col-md-1 col-md-10">

<input type="submit" value="löschen" class="btn btn-default" />

@Html.ActionLink("abbrechen", "Option", "AdminOption", null, new { @class = "btn btn-default" }) </div></div>}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

***Zobrazení zpráv od web visitor adminem (feedback)***

CONTROLLER: AdminFeedbackDisplayController

using System.Web.Mvc;

using MVCTimetable.Models;

namespace MVCTimetable.Controllers

{

public class AdminFeedbackDisplayController : Controller

{

[Authorize]

public ActionResult FeedbackDisplay()

{

AdminFeedbackDisplayViewModel adminFeedbackDisplay = new AdminFeedbackDisplayViewModel();

return View(adminFeedbackDisplay);

}

}

}

MODEL: AdminFeedbackDisplayViewModel

using System.Collections.Generic;

using System.Linq;

using CLTimeTableDB;

namespace MVCTimetable.Models

{

public class AdminFeedbackDisplayViewModel

{

DbRepository dbRepository = new DbRepository();

public List<FeedbackEntity> DisplayMessages { get; private set; }

public AdminFeedbackDisplayViewModel()

{

DisplayMessages = new List<FeedbackEntity>();

List<FeedbackEntityDL> messagesFromDB = dbRepository.GetMessages();

DisplayMessages = messagesFromDB.Select(x => new FeedbackEntity(x.Id,

x.RequestType,

x.Message,

x.Email,

x.PhoneNumber

)).ToList();

}

}

}

VIEW: AdminFeedbackDisplay: FeedbackDisplay

@model MVCTimetable.Models.AdminFeedbackDisplayViewModel

@{

ViewBag.Title = "Abbildung der Berichten von Besuchern";

}

@using (Html.BeginForm())

{

Html.AntiForgeryToken();

Html.ValidationSummary(true);

foreach (var message in Model.DisplayMessages)

{

<div class="row">

<div class="col-md-2">

@Html.LabelFor(model => message.Id, new { @class="control-label"})

@Html.TextBoxFor(model => message.Id, new { @class="form-control", @readonly = "readonly" })

</div>

<div class="col-md-2">

@Html.LabelFor(model => message.RequestType, new { @class="control-label"})

@Html.TextBoxFor(model => message.RequestType, new { @class="form-control", @readonly = "readonly" })

</div>

<div class="col-md-2">

@Html.LabelFor(model => message.Message, new { @class="control-label"})

@Html.TextBoxFor(model => message.Message, new { @class="form-control", @readonly = "readonly" })

</div>

<div class="col-md-2">

@Html.LabelFor(model => message.Email, new { @class="control-label"})

@Html.TextBoxFor(model => message.Email, new { @class="form-control", @readonly = "readonly" })

</div>

<div class="col-md-2">

@Html.LabelFor(model => message.PhoneNumber, new { @class="control-label"})

@Html.TextBoxFor(model => message.PhoneNumber, new { @class="form-control", @readonly = "readonly" })

</div>

</div><br/>

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

***Výmaz zpráv od web visitor adminem (feedback)***

CONTROLLER: AdminFeedbackDeleteController

using System.Web.Mvc;

using MVCTimetable.Models;

namespace MVCTimetable.Controllers

{

public class AdminFeedbackDeleteController : Controller

{

[Authorize]

public ActionResult FeedbackDelete()

{

AdminFeedbackDeleteViewModel adminFeedbackDeleteViewModel = new AdminFeedbackDeleteViewModel();

return View(adminFeedbackDeleteViewModel);

}

[Authorize]

[HttpPost]

public ActionResult FeedbackDelete(AdminFeedbackDeleteViewModel adminFeedbackDeleteViewModel)

{

if (!ModelState.IsValid)

{

return View(adminFeedbackDeleteViewModel);

}

bool found = adminFeedbackDeleteViewModel.FindMessage(adminFeedbackDeleteViewModel.Id);

if (!found)

{

ModelState.AddModelError(nameof(adminFeedbackDeleteViewModel.Id), $"Die Identifizierungsnummer {adminFeedbackDeleteViewModel.Id} wurde nicht gefunden");

return View(adminFeedbackDeleteViewModel);

}

return View("MessageForDelete",adminFeedbackDeleteViewModel);

}

[Authorize]

[HttpPost]

public ActionResult MessageToDelete(AdminFeedbackDeleteViewModel adminFeedbackDeleteViewModel)

{

adminFeedbackDeleteViewModel.DeleteMessage(adminFeedbackDeleteViewModel.Id);

TempData["adminFeedbackDeleteResult"] = $"Der Bericht {adminFeedbackDeleteViewModel.Id} wurde gelöscht";

return RedirectToAction("Option","AdminOption");

}

}

}

MODEL: AdminFeedbackDeleteViewModel

using System;

using System.ComponentModel.DataAnnotations;

using CLTimeTableDB;

namespace MVCTimetable.Models

{

public class AdminDeleteViewModel

{

DbRepository dbRepository = new DbRepository();

CityCache cityCache = new CityCache();

[Display(Name = "Identifizierungsnummer")]

[Required(ErrorMessage = "Geben Sie bitte eine Identifizierungsnummer an")]

public int? IdConnection { get; set; }

public ConnectionEntity ConnectionToDelete { get; private set; }

public string GetCityById(int cityId)

{

CityEntityDL cityEntity = cityCache.GetCityById(cityId);

string cityName = cityEntity.CityName;

return cityName;

}

private int GetIdConnection(int? idConnection)

{

if (!idConnection.HasValue)

throw new InvalidOperationException($"{nameof(idConnection)} is empty.");

return idConnection.Value;

}

public bool FindConnection(int? id)

{

int idConnection = GetIdConnection(id);

bool found;

ConnectionEntityDL connectionFromDB = dbRepository.GetConnectionByID(idConnection);

if (connectionFromDB == null)

{

return found = false;

}

ConnectionToDelete = new ConnectionEntity(connectionFromDB.IdConnection,

connectionFromDB.DepartureCityId,

connectionFromDB.DepartureTime,

connectionFromDB.ArrivalCityId,

connectionFromDB.ArrivalTime

);

return found = true;

}

public void DeleteConnection(int? id)

{

int idConnection = GetIdConnection(id);

dbRepository.DeleteConnection(idConnection);

}

}

}

VIEW: AdminFeedbackDelete: FeedbackDelete

@model MVCTimetable.Models.AdminFeedbackDeleteViewModel

@{

ViewBag.Title = "AdminFeedbackDelete";

}

<h2>Löschung eines Berichts vom Besucher aufgrund ID</h2><br/>

@using (Html.BeginForm())

{

Html.AntiForgeryToken();

Html.ValidationSummary(true);

<div class="form-horizontal">

<div class="form-group">

@Html.LabelFor(model => model.Id, new { @class="control-label col-md-2"})

<div class="col-md-10">

@Html.EditorFor(model => model.Id, new { @class="form-control"})

@Html.ValidationMessageFor(model => model.Id,"",new { @class="text-danger"})

</div>

</div><br/>

<div class="form-group">

<div class="col-md-offset-2 col-md-10">

<input type="submit" value="finden" class="btn btn-default" />

</div>

</div>

</div>

}

VIEW: AdminFeedbackDelete: MessageForDelete

@model MVCTimetable.Models.AdminFeedbackDeleteViewModel

@{

ViewBag.Title = "AdminFeedbackDelete";

}

<h2>Abbildung des Berichts im Zwecke seiner Löschung</h2><br/>

@using (Html.BeginForm("MessageToDelete", "AdminFeedbackDelete",Model))

{

Html.AntiForgeryToken();

Html.ValidationSummary(true);

<div class="row">

<div class="col-md-2">

@Html.LabelFor(model => model.MessageToDelete.Id, new { @class="control-label col-md-2"})

@Html.TextBoxFor(model => model.MessageToDelete.Id, new { @class="form-control", @readonly = "readonly" })

</div>

<div class="col-md-2">

@Html.LabelFor(model => model.MessageToDelete.RequestType, new { @class="control-label col-md-2"})

@Html.TextBoxFor(model=> model.MessageToDelete.RequestType, new { @class="form-control", @readonly = "readonly" })

</div>

<div class="col-md-2">

@Html.LabelFor(model => model.MessageToDelete.Message, new { @class="control-label col-md-2"})

@Html.TextBoxFor(model => model.MessageToDelete.Message, new { @class= "form-control", @readonly = "readonly" })

</div>

<div class="col-md-2">

@Html.LabelFor(model => model.MessageToDelete.Email, new { @class="control-label col-md-2"})

@Html.TextBoxFor(model => model.MessageToDelete.Email, new { @class="form-control", @readonly = "readonly" })

</div>

<div class="col-md-2">

@Html.LabelFor(model => model.MessageToDelete.PhoneNumber, new { @class="control-label col-md-2"})

@Html.TextBoxFor(model => model.MessageToDelete.PhoneNumber, new { @class="form-control", @readonly = "readonly" })

</div>

</div><br/>

<div class="row">

<div class="col-md-1 col-md-10">

<input type="submit" value="löschen" class="btn btn-default" />

@Html.ActionLink("abbrechen","Option","AdminOption",null,new { @class="btn btn-default"})

</div>

</div>

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

***Zobrazení produktů web visitorovi***

CONTROLLER: ConnectionsController

using System.Web.Mvc;

using MVCTimetable.Models;

namespace MVCTimetable.Controllers

{

public class ConnectionsController : Controller

{

public ActionResult DisplayPossibilities()

{

ConnectionsViewModel connectionsViewModel = new ConnectionsViewModel();

if (connectionsViewModel.Cities.Count == 0 )

{

TempData["NoConnection"] = "Derzeit können wir Ihnen leider keine Verkehrsverbindugen anbieten";

return RedirectToAction("Index", "Home");

}

return View(connectionsViewModel);

}

public ActionResult DisplayConnections(ConnectionsViewModel connectionsViewModel)

{

if (!ModelState.IsValid)

{

return View("DisplayPossibilities",connectionsViewModel);

}

if (connectionsViewModel.DepartureCityId == connectionsViewModel.ArrivalCityId)

{

ModelState.AddModelError(nameof(connectionsViewModel.DepartureCityId), "Abfahrtsstad und Anfahrtsstadt sind identisch");

ModelState.AddModelError(nameof(connectionsViewModel.ArrivalCityId), "Abfahrtsstad und Anfahrtsstadt sind identisch");

return View("DisplayPossibilities",connectionsViewModel);

}

bool found = connectionsViewModel.FindConnections(connectionsViewModel);

if (!found)

{

TempData["NoConnectionThisWay"] = "Derzeit können wir Ihnen in diese Richtung keine VerkehrsVerbindungen anbieten";

return RedirectToAction("DisplayPossibilities");

}

return View(connectionsViewModel);

}

}

}

MODEL: ConnectionsViewModel

using System;

using System.Collections.Generic;

using System.Linq;

using CLTimeTableDB;

using System.ComponentModel.DataAnnotations;

namespace MVCTimetable.Models

{

public class ConnectionsViewModel

{

CityCache cityCache = new CityCache();

DbRepository dbRepository = new DbRepository();

[Display(Name = "Abfahrtsdatum")]

[Required(ErrorMessage = "Geben Sie bitte das Abfahrtsdatum an")]

[DataType(DataType.Date)]

public DateTime? DepartureDate { get; set; }

[Display(Name ="Abfahrtsstadt")]

public int DepartureCityId { get; set; }

[Display(Name ="Anfahrtsstadt")]

public int ArrivalCityId { get; set; }

public List<SelectListForWebVisitors> Cities { get; private set; }

public List<ConnectionEntity> Connections { get; private set; }

public ConnectionsViewModel()

{

Cities = new List<SelectListForWebVisitors>();

foreach (var city in cityCache.GetCities())

{

Cities.Add(new SelectListForWebVisitors(city.Key, city.Value.CityName));

}

}

public string GetCityName(int cityId)

{

CityEntityDL cityEntity = cityCache.GetCityById(cityId);

string cityName=cityEntity.CityName;

return cityName;

}

private DateTime GetDepartureDate(DateTime? departureDate)

{

if (!departureDate.HasValue)

throw new InvalidOperationException($"{nameof(departureDate)} is empty");

return departureDate.Value;

}

public bool FindConnections(ConnectionsViewModel connectionsViewModel)

{

bool found;

DateTime departureDate = GetDepartureDate(connectionsViewModel.DepartureDate);

List<ConnectionEntityDL> connectionsFromDB = dbRepository.GetConnectionsByDepartureCityIdArrivalCityId(connectionsViewModel.DepartureCityId, connectionsViewModel.ArrivalCityId).ToList();

if (connectionsFromDB.Count() == 0)

return found = false;

Connections = connectionsFromDB.Select(x => new ConnectionEntity(x.DepartureCityId,

x.DepartureTime,

x.ArrivalCityId,

x.ArrivalTime

)).ToList();

{ DepartureDate = departureDate; }

;

return found = true;

}

}

public class SelectListForWebVisitors

{

public int CityId { get; private set; }

public string CityName { get; private set; }

public SelectListForWebVisitors(int cityId,string cityName)

{

CityId = cityId;

CityName = cityName;

}

}

}

VIEW: Connections: DisplayPossibilities

@model MVCTimetable.Models.ConnectionsViewModel

@{

ViewBag.Title = "Possibilities";

}

<h2>Auswahl der Verkehrsverbindungen</h2><br/>

@using (Html.BeginForm("DisplayConnections", "Connections",FormMethod.Get))

{

Html.AntiForgeryToken();

Html.ValidationSummary(true);

<div class="form-horizontal">

<div class="form-group">

@Html.LabelFor(model => model.DepartureDate, new { @class="control-label col-md-2"})

<div class="col-md-10">

@Html.EditorFor(model => model.DepartureDate, new { @class = "form-control", })

@Html.ValidationMessageFor(model => model.DepartureDate, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.DepartureCityId, new { @class= "control-label col-md-2" })

<div class="col-md-10">

@Html.DropDownListFor(model => model.DepartureCityId, new SelectList(Model.Cities, "CityId", "CityName"))

@Html.ValidationMessageFor(model => model.DepartureCityId,"",new { @class="text-danger"})

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.ArrivalCityId, new { @class= "control-label col-md-2" })

<div class="col-md-10">

@Html.DropDownListFor(model => model.ArrivalCityId, new SelectList(Model.Cities, "CityId", "CityName"))

@Html.ValidationMessageFor(model => model.ArrivalCityId,"",new { @class= "text-danger" })

</div>

</div>

<div class="form-group">

<div class="col-md-offset-2 col-md-10">

<input type="submit" formmethod="get" value="finden" class="btn btn-default" />

</div>

</div>

</div>

}

VIEW: Connections: DisplayConnections

@model MVCTimetable.Models.ConnectionsViewModel

@{

ViewBag.Title = "Connections";

}

<h2>Ihre Verkehrsverbindungen</h2>

@using (Html.BeginForm())

{

Html.AntiForgeryToken();

Html.ValidationSummary(true);

string departureCityName=Model.GetCityName(Model.DepartureCityId);

string arrivalCityName=Model.GetCityName(Model.ArrivalCityId);

foreach (var connection in Model.Connections)

{

<div class="row">

<div class="col-md-2">

@Html.LabelFor(model => model.DepartureDate, new { @class= "control-label" })

@Html.TextBoxFor(model => model.DepartureDate, @"{0:dd/MM/yyyy}", new { @class="form-control", @readonly= "readonly" })

</div>

<div class="col-md-2">

@Html.LabelFor(model => connection.DepartureCityId, new { @class="control-label"})

@Html.TextBoxFor(model => departureCityName, new { @class="form-control", @readonly= "readonly" })

</div>

<div class="col-md-2">

@Html.LabelFor(model => connection.DepartureTime, new { @class="control-label"})

@Html.TextBoxFor(model => connection.DepartureTime, @"{0:hh\:mm}", new { @class="form-control", @readonly="readonly"})

</div>

<div class="col-md-2">

@Html.LabelFor(model => connection.ArrivalCityId, new { @class="control-label"})

@Html.TextBoxFor(model => arrivalCityName, new { @class="form-control", @readonly= "readonly" })

</div>

<div class="col-md-2">

@Html.LabelFor(model => connection.ArrivalTime, new { @class="control-label"})

@Html.TextBoxFor(model => connection.ArrivalTime, @"{0:hh\:mm}", new { @class="form-control", @readonly= "readonly" })

</div>

</div>

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

***Zaslání zprávy od web visitorova k admin stránek***

CONTROLLER: HomeController

using System.Web.Mvc;

using MVCTimetable.Models;

namespace MVCTimetable.Controllers

{

public class HomeController : Controller

{

public ActionResult Index()

{

FeedbackViewModel feedbackViewModel = new FeedbackViewModel();

return View(feedbackViewModel);

}

[HttpPost]

public ActionResult Index(FeedbackViewModel feedbackViewModel)

{

if (!ModelState.IsValid)

{

return View(feedbackViewModel);

}

feedbackViewModel.AddMessageToDatabase(feedbackViewModel);

TempData["FeedbackResult"] = "Wir danken für Ihre Meinung";

return RedirectToAction("Index");

}

}

}

MODEL: FeedbackViewModel

using System.Collections.Generic;

using System.Web.Mvc;

using System.ComponentModel.DataAnnotations;

using CLTimeTableDB;

namespace MVCTimetable.Models

{

public class FeedbackViewModel

{

DbRepository DbRepository = new DbRepository();

public List<SelectListItem> RequestTypes { get; private set; }

[Display(Name ="Ihr Email:")]

[Required(ErrorMessage ="Geben Sie bitte Ihr Email an")]

public string Email { get; set; }

[Display(Name ="Ihre Handynummer")]

public string PhoneNumber { get; set; }

[Display(Name ="Bericht")]

[MaxLength(1000,ErrorMessage ="Ihre Bericht ist zu lang. Maximum ist tausend Zeichen")]

public string Message { get; set; }

[Display(Name = "Anforderungstyp")]

public string RequestType { get; set; }

public FeedbackViewModel()

{

RequestTypes = new List<SelectListItem>();

RequestTypes.Add(new SelectListItem { Text = "Anforderungstyp", Value = "RequestTyp" });

RequestTypes.Add(new SelectListItem { Text = "Belobung", Value = "Compliment" });

RequestTypes.Add(new SelectListItem { Text = "Beschwerde", Value = "Complaint" });

RequestTypes.Add(new SelectListItem { Text = "Rufen Sie mich zurück", Value = "CallMeBack" });

RequestTypes.Add(new SelectListItem { Text = "Sonstige", Value = "Another" });

}

public void AddMessageToDatabase(FeedbackViewModel feedbackViewModel)

{

FeedbackEntityDL messageToDB = new FeedbackEntityDL(feedbackViewModel.Email, feedbackViewModel.PhoneNumber, feedbackViewModel.Message, feedbackViewModel.RequestType);

DbRepository.AddMessageFromWebSite(messageToDB);

}

}

}

VIEW: Home: Index

@model MVCTimetable.Models.FeedbackViewModel

@{

ViewBag.Title = "Reisebüro";

}

<div class="jumbotron">

<h1>R E I S E B Ü R O</h1>

</div>

<div class="row">

<p> <h4>Mit unserer Reisebüro können Sie komfortabel nach Prag reisen. </h4></p><br/>

<p> <h4>Schreiben Sie uns bitte Ihre Meinung:</h4></p>

</div>

@using (Html.BeginForm())

{

@Html.AntiForgeryToken()

@Html.ValidationSummary(true)

<div class="form-horizontal">

<div class="form-group">

@Html.LabelFor(m => m.Email, new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(m => m.Email, new { @class = "form-control" })

@Html.ValidationMessageFor(m => m.Email, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(m => m.PhoneNumber, new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(m => m.PhoneNumber, new { @class ="form-control"})

@Html.ValidationMessageFor(m => m.PhoneNumber,"", new { @class = "text-danger"})

</div>

</div>

<div class="form-group">

@Html.LabelFor(m => m.Message, new { @class="control-label col-md-2"})

<div class="col-md-10">

@Html.EditorFor(m => m.Message, new { @class = "form-control" })

@Html.ValidationMessageFor(m => m.Message,"", new { @class="text-danger"})

</div>

</div>

<div class="form-group">

@Html.LabelFor(m => m.RequestType, new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.DropDownListFor(m => m.RequestType, new SelectList(Model.RequestTypes,"Value","Text"))

@Html.ValidationMessageFor(m => m.RequestType)

</div>

</div>

<div class="form-group">

<div class="col-md-offset-2 col-md-10">

<input type="submit" value="Abschicken" class="btn btn-default" />

</div>

</div>

</div>

}

----------------------------CL\_TIME TABLE\_DL----data layer-----------------------------------------------

***Představení použitých entit a přístup k DB***

CityEntityDL:

namespace CLTimeTableDB

{

public class CityEntityDL

{

public int Id { get; set; }

public string CityName { get; set; }

public override string ToString()

{

return CityName;

}

}

}

ConnectionEntityDL:

using System;

namespace CLTimeTableDB

{

public class ConnectionEntityDL

{

public int IdConnection { get; private set; }

public int DepartureCityId { get; private set; }

public TimeSpan DepartureTime { get; private set; }

public int ArrivalCityId { get; private set; }

public TimeSpan ArrivalTime { get; private set; }

public ConnectionEntityDL(int id, int departureCityId, TimeSpan departureTime, int arrivalCityId, TimeSpan arrivalTime)

{

IdConnection = id;

DepartureCityId = departureCityId;

DepartureTime = departureTime;

ArrivalCityId = arrivalCityId;

ArrivalTime = arrivalTime;

}

public ConnectionEntityDL(int departureCityId, TimeSpan departureTime, int arrivalCityId, TimeSpan arrivalTime)

{

DepartureCityId = departureCityId;

DepartureTime = departureTime;

ArrivalCityId = arrivalCityId;

ArrivalTime = arrivalTime;

}

public ConnectionEntityDL()

{ }

public ConnectionEntityDL(int iD)

{

Id = iD;

}

}

}

FeedbackEntityDL:

namespace CLTimeTableDB

{

public class FeedbackEntityDL

{

public int Id { get; private set; }

public string RequestType { get; private set; }

public string Message { get; private set; }

public string Email { get; private set; }

public string PhoneNumber { get; private set; }

public FeedbackEntityDL(string email,string phoneNumber,string message,string requestType)

{

Email = email;

PhoneNumber = phoneNumber;

Message = message;

RequestType = requestType;

}

public FeedbackEntityDL(int id,string requestType,string message,string email,string phoneNumber)

{

Id = id;

RequestType = requestType;

Message = message;

Email = email;

PhoneNumber = phoneNumber;

}

public FeedbackEntityDL()

{ }

public FeedbackEntityDL(int iD)

{

Id = iD;

}

}

}

EntityFramework:

using System.Data.Entity;

namespace CLTimeTableDB

{

public class EntityFramework:DbContext

{

public DbSet<CityEntityDL> CityEntityDLTable { get; set; }

public DbSet<ConnectionEntityDL> ConnectionEntityDLTable { get; set; }

public DbSet<FeedbackEntityDL> FeedbackEntityDLTable { get; set; }

public EntityFramework():base("DefaultConnection")

{ }

}

}

***CityCache:***

using System.Collections.Generic;

using System.Linq;

namespace CLTimeTableDB

{

public class CityCache

{

private Dictionary<int, CityEntityDL> citycache;

DbRepository dbConnectionRepository = new DbRepository();

public Dictionary<int, CityEntityDL> GetCities()

{

EnsureCacheInited();

return citycache;

}

public CityEntityDL GetCityById(int cityId)

{

EnsureCacheInited();

CityEntityDL city;

return citycache.TryGetValue(cityId, out city) ? city : null;

}

private void EnsureCacheInited()

{

if (citycache == null)

{

citycache = dbConnectionRepository.GetCities()

.ToDictionary(c => c.Id);

}

}

}

}

***DbRepository***

using System;

using System.Collections.Generic;

using System.Data.Entity;

using System.Linq;

using System.Xml.Serialization;

using System.IO;

namespace CLTimeTableDB

{

public class DbRepository

{

private const string path = "C:/Users/David/Desktop/David/Programovani/MVCProjects/MVCTimetable/CLTimeTableDB/1\_ERRORS/Errors.txt";

//private const string ConnectionString = @"Data Source=.\sqlexpress;Initial Catalog=TimetableDB;Integrated Security=True";

//private ConnectionEntityDL GetDbConnectionEntity(SqlDataReader reader)

//{

// return new ConnectionEntityDL(reader.GetInt32(reader.GetOrdinal("id")),

// reader.GetInt32(reader.GetOrdinal("departureCityId")),

// reader.GetTimeSpan(reader.GetOrdinal("departureTime")),

// reader.GetInt32(reader.GetOrdinal("arrivalCityId")),

// reader.GetTimeSpan(reader.GetOrdinal("arrivalTime"))

// );

//}

private void ExceptionLog (string error)

{

try

{

XmlSerializer serializer = new XmlSerializer(typeof(string));

using (StreamWriter writer = new StreamWriter(path))

{

serializer.Serialize(writer, error);

}

}

catch (Exception ex)

{

string IOError = ex.Message.ToString();

}

}

// GET ALL CONNECTIONS

public List<ConnectionEntityDL> GetConnections()

{

try

{

using (EntityFramework accessToDB = new EntityFramework())

{

return accessToDB.ConnectionEntityDLTable.ToList();

}

}

catch (Exception ex)

{

ExceptionLog(ex.Message.ToString());

List<ConnectionEntityDL> nullObject = new List<ConnectionEntityDL>();

nullObject.Add(new ConnectionEntityDL());

return nullObject;

}

}

//public IEnumerable<ConnectionEntityDL> GetConnections()

//{

// using (SqlConnection connection = new SqlConnection(ConnectionString))

// {

// connection.Open();

// using (SqlCommand command = new SqlCommand("SELECT id,departureCityId,arrivalCityId,departureTime,arrivalTime FROM Departures2", connection))

// {

// using (SqlDataReader reader = command.ExecuteReader())

// {

// while (reader.Read())

// {

// yield return GetDbConnectionEntity(reader);

// }

// }

// }

// }

//}

// GET ONE CONNECTION BY ID

public ConnectionEntityDL GetConnectionByID(int idConnection)

{

try

{

using (EntityFramework accessToDB = new EntityFramework())

{

return accessToDB.ConnectionEntityDLTable.Where(x => x.Id == idConnection).SingleOrDefault();

}

}

catch (Exception ex)

{

ExceptionLog(ex.Message.ToString());

ConnectionEntityDL nullObject = new ConnectionEntityDL();string errorMessage = $"{ex.Message.ToString()}";

return nullObject; }

//using (SqlConnection connection = new SqlConnection(ConnectionString))

//{

// connection.Open();

// using (SqlCommand command = new SqlCommand("SELECT id,departureCityId,arrivalCityId,departureTime,arrivalTime FROM Departures2 WHERE id=@idConnection ", connection))

// {

// command.Parameters.AddWithValue("@idConnection", idConnection);

// using (SqlDataReader reader = command.ExecuteReader())

// {

// if (!reader.Read())

// {

// return null;

// }

// return GetDbConnectionEntity(reader);

// }

// }

//}

}

// GET CONNECTIONS BY DEPARTURE CITY ID AND ARRIVAL CITY ID

public List<ConnectionEntityDL> GetConnectionsByDepartureCityIdArrivalCityId(int departureCityId, int arrivalCityId)

{

try

{

using (EntityFramework accessToDB = new EntityFramework())

{

return accessToDB.ConnectionEntityDLTable.Where(x => x.DepartureCityId == departureCityId)

.Where(x => x.ArrivalCityId == arrivalCityId)

.ToList();

}

}

catch (Exception ex)

{

ExceptionLog(ex.Message.ToString());

List<ConnectionEntityDL> nullObject = new List<ConnectionEntityDL>();

nullObject.Add(new ConnectionEntityDL()); string errorMessage = $"{ex.Message.ToString()}";

return nullObject; }

}

//public IEnumerable<ConnectionEntityDL> GetConnectionsByDepartureCityIdArrivalCityId(int departureCityId, int arrivalCityId)

//{

// using (SqlConnection connection = new SqlConnection(ConnectionString))

// {

// connection.Open();

// using (SqlCommand command = new SqlCommand("SELECT id,departureCityId,arrivalCityId,departureTime,arrivalTime FROM Departures2 WHERE departureCityId = @departureCityId AND arrivalCityId = @arrivalCityId", connection))

// {

// command.Parameters.AddWithValue("@departureCityId", departureCityId);

// command.Parameters.AddWithValue("@arrivalCityId", arrivalCityId);

// using (SqlDataReader reader = command.ExecuteReader())

// {

// while (reader.Read())

// {

// yield return GetDbConnectionEntity(reader);

// }

// }

// }

// }

//}

// INSERT ONE CONNECTION

public string InsertConnection(ConnectionEntityDL connectionToDB)

{

string result = "";

try

{

using (EntityFramework accessToDB = new EntityFramework())

{

accessToDB.ConnectionEntityDLTable.Add(connectionToDB);

accessToDB.SaveChanges();

return result = "Die Verbindung wurde gegründet";

}

}

catch (Exception ex)

{

return result = $"Die Verbindung wurde nicht gegründet! Fehlerbeschreibung:\n {ex.Message.ToString()}";

}

//using (SqlConnection connection = new SqlConnection(ConnectionString))

//{

// connection.Open();

// using (SqlCommand command = new SqlCommand("INSERT INTO Departures2(departureCityId,arrivalCityId,departureTime,arrivalTime) VALUES (@departureCity,@arrivalCity,@departureTime,@arrivalTime)", connection))

// {

// command.Parameters.AddWithValue("@departureCity", connectionToDB.DepartureCityId);

// command.Parameters.AddWithValue("@arrivalCity", connectionToDB.ArrivalCityId);

// command.Parameters.AddWithValue("@departureTime", connectionToDB.DepartureTime);

// command.Parameters.AddWithValue("@arrivalTime", connectionToDB.ArrivalTime);

// command.ExecuteNonQuery();

// }

//}

}

// UPDATE ONE CONNECTION

public string UpdateConnection(ConnectionEntityDL connectionToDB)

{

string result = "";

try

{

using (EntityFramework accessToDB = new EntityFramework())

{

accessToDB.Entry(connectionToDB).State = EntityState.Modified;

accessToDB.SaveChanges();

return result = "Die Verkehresverbindung wurde korrigiert.";

}

}

catch (Exception ex)

{

return result = $"Die Verkehresverbindung wurde nicht korrigiert. Fehlerbeschreibung: {ex.Message.ToString()}";

}

//using (SqlConnection connection = new SqlConnection(ConnectionString))

//{

// connection.Open();

// using (SqlCommand command = new SqlCommand("UPDATE Departures2 SET departureCityId=@departureCityId, arrivalCityId=@arrivalCityId, departureTime=@departureTime, arrivalTime=@arrivalTime WHERE id=@IdConnection", connection))

// {

// command.Parameters.AddWithValue("@departureCityId", connectionToDB.DepartureCityId);

// command.Parameters.AddWithValue("@arrivalCityId", connectionToDB.ArrivalCityId);

// command.Parameters.AddWithValue("@departureTime", connectionToDB.DepartureTime);

// command.Parameters.AddWithValue("@arrivalTime", connectionToDB.ArrivalTime);

// command.Parameters.AddWithValue("@IdConnection", connectionToDB.Id);

// command.ExecuteNonQuery();

// }

//}

}

// DELETE ONE CONNECTION

public string DeleteConnection(int idConnection)

{

string result = "";

try

{

using (EntityFramework accessToDB = new EntityFramework())

{

accessToDB.Entry(new ConnectionEntityDL(idConnection)).State = EntityState.Deleted;

accessToDB.SaveChanges();

return result = $"Die Verkehrsverbindung {idConnection} wurde gelöscht";

}

}

catch (Exception ex)

{

return result = $"Die Verkehrsverbindung {idConnection} wurde nicht gelöscht. Fehlerbeschreibung: {ex.Message.ToString()}";

}

//using (SqlConnection connection = new SqlConnection(ConnectionString))

//{

// connection.Open();

// using (SqlCommand command = new SqlCommand("DELETE FROM Departures2 WHERE id=@idConnection", connection))

// {

// command.Parameters.AddWithValue("@idConnection", idConnection);

// command.ExecuteNonQuery();

// }

//}

}

// ADD ONE MESSAGE FROM WEB SITE

public string AddMessageFromWebSite(FeedbackEntityDL messageToDB)

{

string result = "";

try

{

using (EntityFramework accessToDB = new EntityFramework())

{

accessToDB.FeedbackEntityDLTable.Add(messageToDB);

accessToDB.SaveChanges();

return result = "Wir danken für Ihre Meinung";

}

}

catch (Exception ex)

{

return result = $"Die Bericht wurde nicht gespeichert. Fehlerbeschreibung: {ex.Message.ToString()}";

}

//using (SqlConnection connection = new SqlConnection(ConnectionString))

//{

// connection.Open();

// using (SqlCommand command = new SqlCommand("INSERT INTO Messages (email, phoneNumber, message, requestType) VALUES (@email, @phoneNumber, @message, @requestType)", connection))

// {

// command.Parameters.AddWithValue("@email", messageToDB.Email);

// command.Parameters.AddWithValue("@phoneNumber", messageToDB.PhoneNumber ?? string.Empty);

// command.Parameters.AddWithValue("@message", messageToDB.Message ?? string.Empty);

// command.Parameters.AddWithValue("@requestType", messageToDB.RequestType);

// command.ExecuteNonQuery();

// }

//}

}

//private FeedbackEntityDL GetFeedbackEntity(SqlDataReader reader)

//{

// return new FeedbackEntityDL(reader.GetInt32(reader.GetOrdinal("id")),

// reader.GetString(reader.GetOrdinal("requestType")),

// reader.GetString(reader.GetOrdinal("message")) ?? string.Empty,

// reader.GetString(reader.GetOrdinal("email")),

// reader.GetString(reader.GetOrdinal("phoneNumber")) ?? string.Empty

// );

//}

// GET ALL MESSAGES

public List<FeedbackEntityDL> GetMessages()

{

try

{

using (EntityFramework accessToDB = new EntityFramework())

{

return accessToDB.FeedbackEntityDLTable.ToList();

}

}

catch (Exception ex)

{

List<FeedbackEntityDL> error = new List<FeedbackEntityDL>();

error.Add(new FeedbackEntityDL("Keine Bericht", "wurde eingeselen", "Fehlerbeschreibung:", $"{ex.Message.ToString()}"));

return error;

}

//public IEnumerable<FeedbackEntityDL> GetMessages()

// {

//using (SqlConnection connection = new SqlConnection(ConnectionString))

//{

// connection.Open();

// using (SqlCommand command = new SqlCommand("SELECT id,requestType,message,email,phoneNumber FROM Messages", connection))

// {

// using (SqlDataReader reader = command.ExecuteReader())

// {

// while (reader.Read())

// {

// yield return GetFeedbackEntity(reader);

// }

// }

// }

//}

//}

}

//GET ONE MESSAGE

public FeedbackEntityDL GetMessage(int idMessage)

{

try

{

using (EntityFramework accessToDB = new EntityFramework())

{

return accessToDB.FeedbackEntityDLTable.Where(x => x.Id == idMessage).SingleOrDefault();

}

}

catch(Exception ex)

{

FeedbackEntityDL error = new FeedbackEntityDL("Keine Bericht", "wurde eingelesen", "Fehlerbeschreibung:",$"{ex.Message.ToString()}");

return error;

}

//using (SqlConnection connection = new SqlConnection(ConnectionString))

//{

// connection.Open();

// using (SqlCommand command = new SqlCommand("SELECT id,requestType,message,email,phoneNumber FROM Messages WHERE id=@idMessage", connection))

// {

// command.Parameters.AddWithValue("@idMessage", idMessage);

// using (SqlDataReader reader = command.ExecuteReader())

// {

// if(!reader.Read())

// {

// return null;

// }

// return GetFeedbackEntity(reader);

// }

// }

//}

}

//DELETE ONE MESSAGE

public string DeleteMessage(int idMessage)

{

string result = "";

try

{

using (EntityFramework accessToDB = new EntityFramework())

{

accessToDB.Entry(new FeedbackEntityDL(idMessage)).State = EntityState.Deleted;

accessToDB.SaveChanges();

return result = $"Der Bericht {idMessage} wurde gelöscht";

}

}

catch (Exception ex)

{

return result = $"Der Bericht {idMessage} wurde nicht gelöscht. Fehlerbeschreibung: {ex.Message.ToString()}";

}

//using (SqlConnection connection = new SqlConnection(ConnectionString))

//{

// connection.Open();

// using (SqlCommand command = new SqlCommand("DELETE FROM Messages WHERE id=@idMessage", connection))

// {

// command.Parameters.AddWithValue("@idMessage", idMessage);

// command.ExecuteNonQuery();

// }

//}

}

// GET CITIES FROM DB TO CITYCACHE

public Dictionary<int,CityEntityDL> GetCities()

{

try

{

using (EntityFramework accessToDB = new EntityFramework())

{

return accessToDB.CityEntityDLTable.ToDictionary(c => c.Id);

}

}

catch (Exception ex)

{

Dictionary<int, CityEntityDL> error = new Dictionary<int, CityEntityDL>();

error.Add(0, new CityEntityDL { CityName = $"Die Angaben wurden nicht durch eingelesen: Fehlerbeschreibung:{ex.Message.ToString()} " });

return error;

}

}

//public IEnumerable<CityEntityDL> GetCities()

//{

// using (SqlConnection connection = new SqlConnection(ConnectionString))

// {

// connection.Open();

// using (SqlCommand command = new SqlCommand("SELECT id,city FROM Cities", connection))

// {

// using (SqlDataReader reader = command.ExecuteReader())

// {

// while (reader.Read())

// {

// var city = new CityEntityDL();

// city.Id = reader.GetInt32(reader.GetOrdinal("id"));

// city.CityName = reader.GetString(reader.GetOrdinal("city"));

// yield return city;

// }

// }

// }

// }

//}

// INSERT ONE CITY TO DB

public string AddCity(string cityName)

{

string result = "";

try

{

using (EntityFramework accessToDB = new EntityFramework())

{

accessToDB.CityEntityDLTable.Add(new CityEntityDL { CityName = cityName });

accessToDB.SaveChanges();

return result = "Anforderung wurde durchgeführt";

}

}

catch(Exception ex)

{

return result = $"Die Angaben wurden nicht eingespeichert! Fehlerbeschreibung:\n {ex.Message.ToString()} ";

}

//using (SqlConnection connection = new SqlConnection(ConnectionString))

//{

// connection.Open();

// using (SqlCommand command = new SqlCommand("INSERT INTO Cities (city) VALUES (@city)", connection))

// {

// command.Parameters.AddWithValue("@city", cityName);

// command.ExecuteNonQuery();

// }

//}

}

// DELETE ONE CITY IN DB

public string DeleteCity(int idCity)

{

string result = "";

try

{

using (EntityFramework accessToDB = new EntityFramework())

{

accessToDB.Entry(new CityEntityDL { Id = idCity }).State = EntityState.Deleted;

accessToDB.SaveChanges();

return result = "Anforderung wurde durchgeführt";

}

}

catch (Exception ex)

{

return result = $"Die Angaben wurden nicht gelöscht! Fehlerbeschreibung:\n {ex.Message.ToString()}";

}

//using (SqlConnection connection = new SqlConnection(ConnectionString))

//{

// connection.Open();

// using (SqlCommand command = new SqlCommand("DELETE FROM Cities WHERE id=@idCity", connection))

// {

// command.Parameters.AddWithValue("@idCity", idCity);

// command.ExecuteNonQuery();

// }

//}

}

}

}