**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

**ЛАБОРАТОРНАЯ РАБОТА №6**

по дисциплине: «Разработка приложений баз данных для информационных систем»

на тему: «Создание *Web*-сервиса с использованием *WEB API ASP.NET MVC* (*ASP.NET Core MVC*) для работы с информацией из базы данных»

Выполнил: студент гр. ИТП-31

Демиденко Д. В.

Принял: доцент

Асенчик О. Д.

Гомель 2023

**Цель работы:** ознакомиться с возможностями .*NET Core* по получению, отображению и изменение данных при помощи технологии *WEB API*.

**Задание:**

Создать с использованием технологии WEB API Web-приложение, в котором организовать получение, отображение и изменение данных.

Для этого необходимо:

1. Создать (использовать ранее разработанные в предыдущих лабораторных работах):

a. Классы, моделирующие не менее чем три таблицы базы данных согласно вашему варианту. Одна из таблиц обязательно должна находиться на стороне отношения «многие» связи с другой таблицей в схеме базы данных.

b. Класс контекста данных для доступа к базе данных.

2. Создать контроллер для операций с данными таблицы, стоящей на стороне отношения «многие».

3. Задокументировать WEB API ASP.NET Core с помощью инструментов Swagger.

4. Клиентское приложение в виде HTML страницы для визуального выполнения всех операций с данными выбранной в п.2 таблицы, использующих запросы к методам разработанного контроллера.

**Ход работы**

В ходе выполнения лабораторной работы были разработан три модели классов, где одна из моделей стоит на стороне отношения многие. Далее был разработан класс контекста для доступа к данным из СУБД. Пример этих классов указан в приложении А.

Далее был разработан контроллер, который будет представлять интерфейс для взаимодействия с *API*. Вся обработка в контроллере была построена по принципу *REST API,* где метод GET отвечает за операцию получения все коллекции данных или конкретного элемента коллекции. Метод *DELETE* отвечает за операцию удаления элемента. Метод *POST* отвечает за операцию добавления элемента. И метод *PUT* отвечает за операцию обновления данных. На основе этих шаблонов был разработан контролер для работы с одно из таблиц базы данных, которая стоит на стороне отношения многие. Пример данного контроллера указан в приложении А.

Далее для более удобно отладки и документации *API* был использован механизм *Swagger*. Пример операции добавления при помощи *Swagger* указан на рисунке 1.

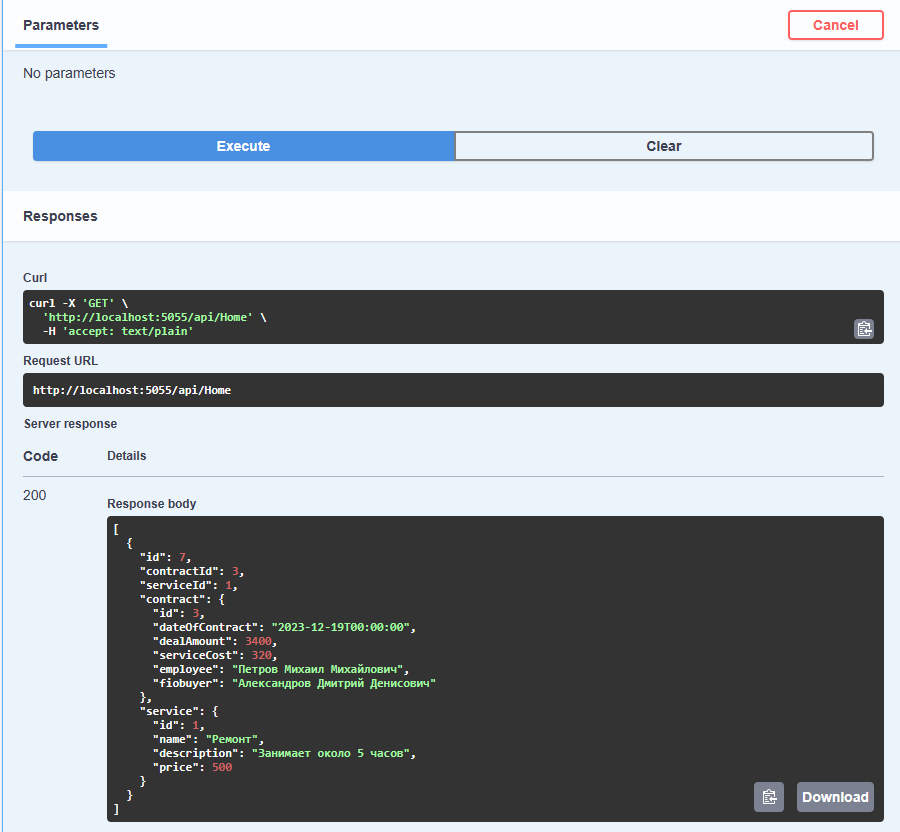


Рисунок 1 – Пример *GET* запроса в *Swagger*

Далее был протестирован *POST* запрос для операции добавления данных. Пример *POST* запроса в *Swagger* указан на рисунке 2.

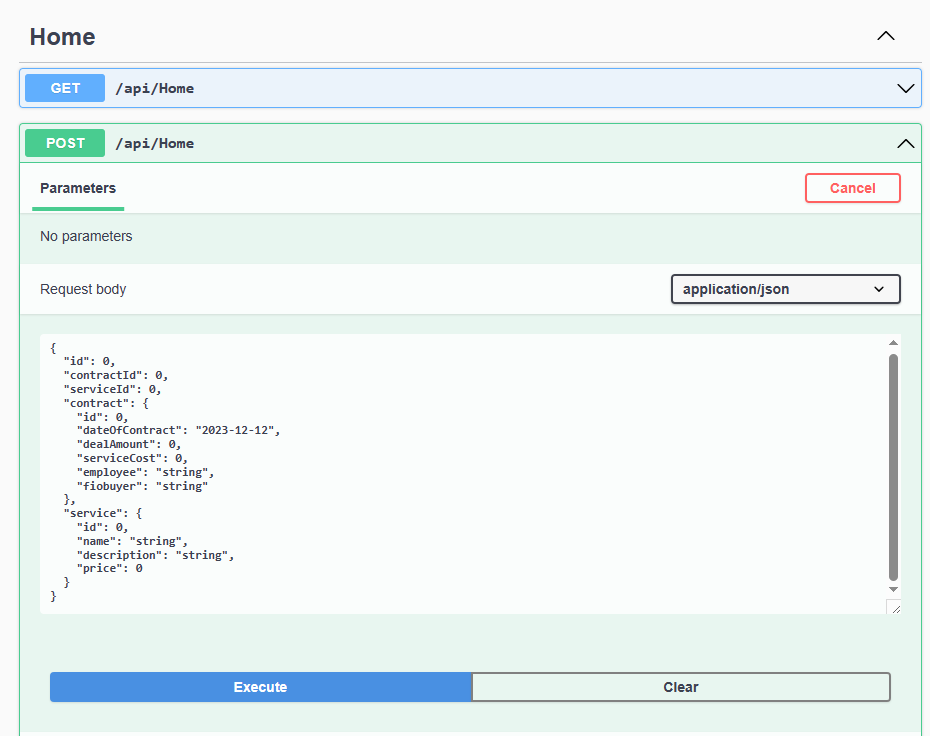


Рисунок 2 – Пример *POST* запроса в *Swagger*

Далее был протестирован *PUT* запрос для операции обновления данных. Пример *PUT* запроса в *Swagger* указан на рисунке 3.

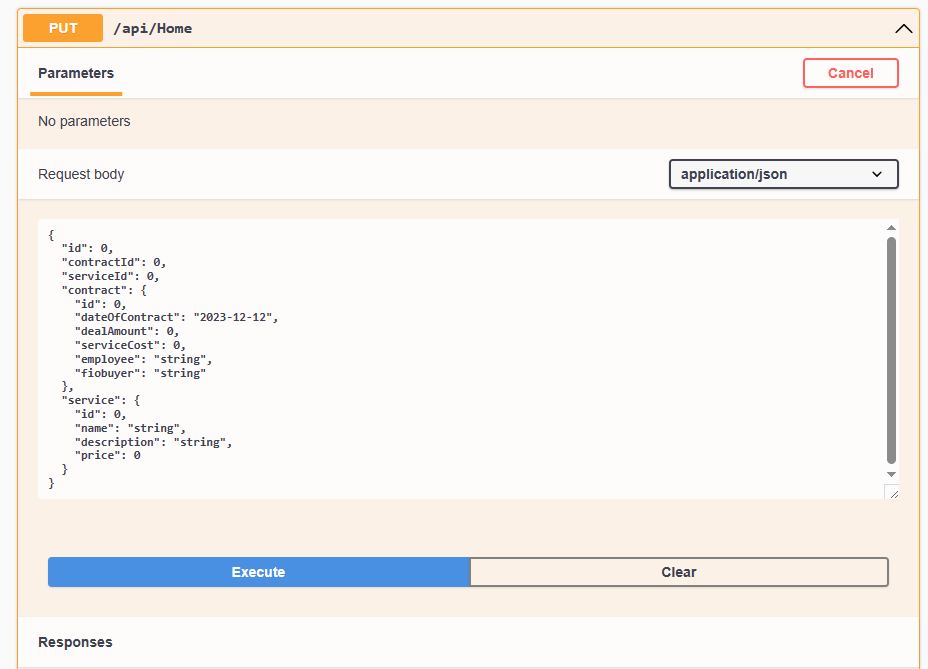


Рисунок 3 – Пример *PUT* запроса в *Swagger*

Далее был протестирован *GET* запрос для операции получения конкретного элемента из базы данных. Пример этого *GET* запроса в *Swagger* указан на рисунке 4.

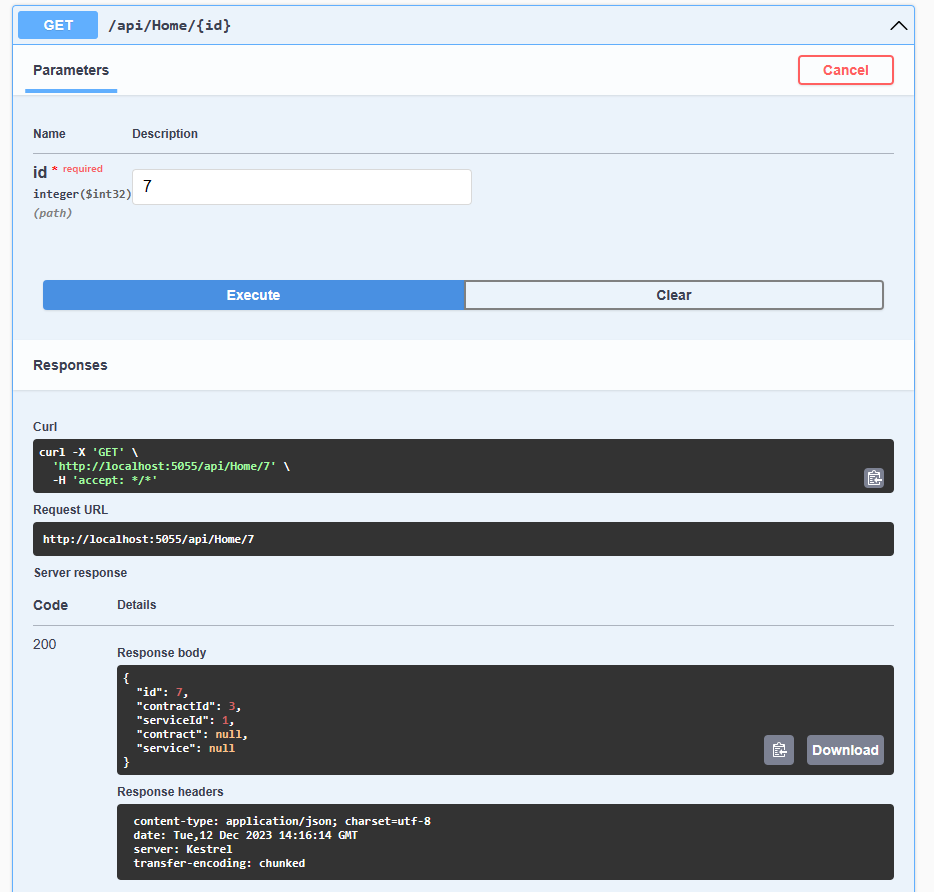


Рисунок 4 – Пример *GET* запроса конкретного элемента по *id* в *Swagger*

Далее был протестирован *DELETE* запрос для операции удаления элемента из базы данных. Пример этого *DELETE* запроса в *Swagger* указан на рисунке 5.

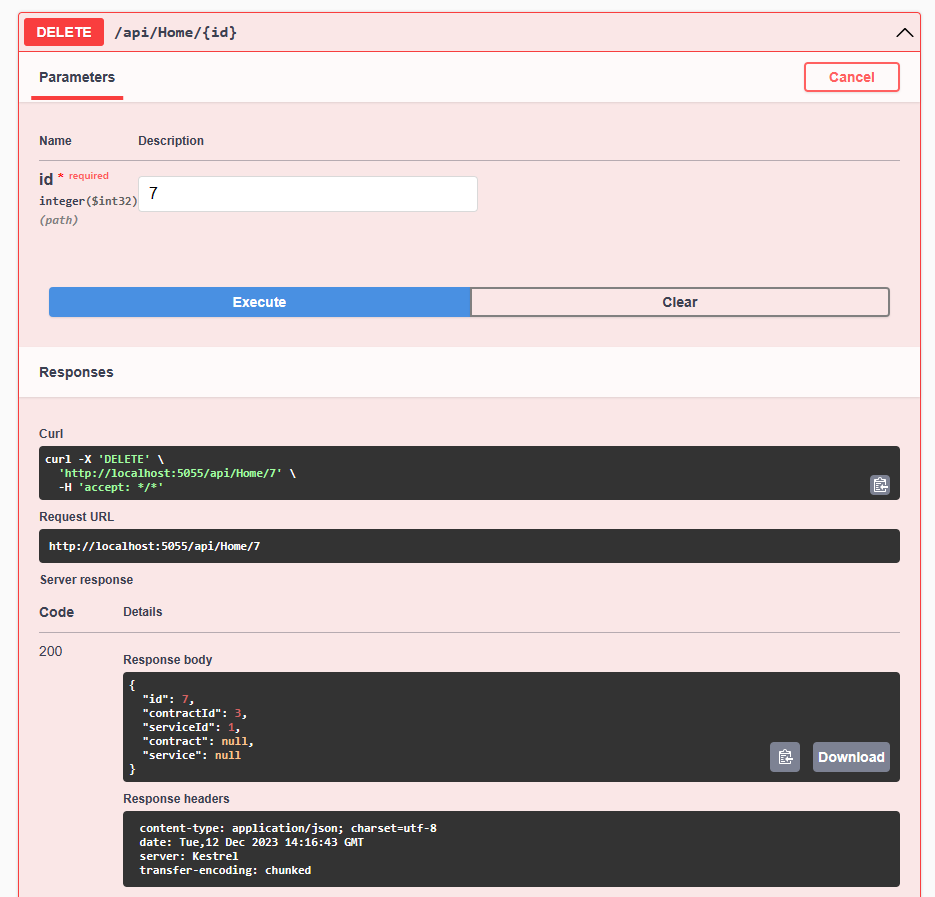


Рисунок 5 – Пример *DELETE* запроса в *Swagger*

Далее было разработана клиентская часть приложения, которая использует написанное *API* для доступа к данным. Клиентское приложение было написано на *HTML* и *JavaScript.* При помощи *js* клиент получает полный доступ к написанному *API*. Пример кода клиентского приложении указан в приложении А. Пример главного меню приложения указан на рисунке 6.

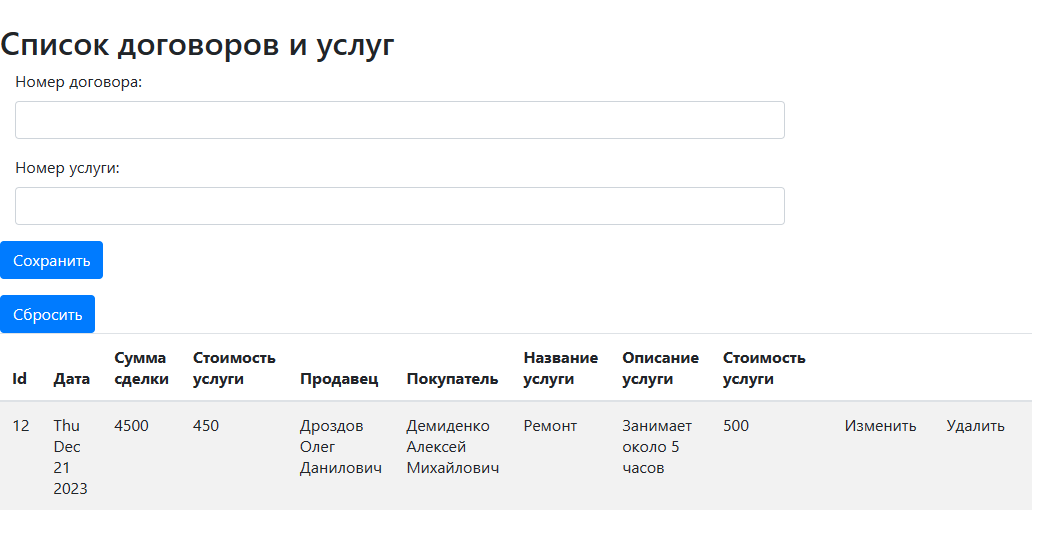


Рисунок 6 – Пример меню *web*-приложения

При загрузке сайта клиент вызывает метод *GET* для получения всех данных и занесения их в таблицу на сайте. При нажатии на кнопку удалить срабатывает *DELETE* запрос на удаление данных. При нажатии на кнопку изменить срабатывает *GET* запрос для элемента по определенному *id* и занесению этих значений в поля выше. Далее при нажатии на кнопку сохранить в зависимости от параметров выполняется *POST* или *PUT* запрос, следовательно происходит добавление или обновления данных.

После выполнения лабораторной работы созданные проект был добавлен в локальный *git* репозиторий а потом перенесен в *GitHub* репозиторий своего аккаунта. Чтобы ознакомится с созданным проектом можно по ссылке [ddemidenko/RPBDIS: Лабораторные работы РПБДТС (github.com)](https://github.com/ddemidenko/RPBDIS)*.*

**Вывод**: в ходе выполнения лабораторной работы был разработан контроллер, который выступает в виде обработчика запросов, которые поступают от клиентов. Был изучен механизм Swagger для удобного документирования и тестирования *WEB API*. Было разработано *web*-приложение, которое получает доступ к данным через *WEB API.*

**ПРИЛОЖНИЕ А**

Листинг класса *Program*

using Microsoft.EntityFrameworkCore;

namespace Lab6

{

public class Program

{

public static void Main(string[] args)

{

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddDbContext<Lab6Context>(e => e.UseSqlServer(builder.Configuration.GetConnectionString("SqlServerConnection")));

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

var app = builder.Build();

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

app.UseStaticFiles();

app.UseAuthorization();

app.MapControllers();

app.Run();

}

}

}

Листинг класса *Contract*

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

namespace Lab6.Models;

public partial class Contract

{

public int Id { get; set; }

[DataType(DataType.Date)]

[Display(Name = "Дата")]

public DateTime DateOfContract { get; set; }

[Display(Name = "Сумма сделки")]

public decimal DealAmount { get; set; }

[Display(Name = "Стоимость услуги")]

public decimal ServiceCost { get; set; }

[Display(Name = "Сотрудник")]

public string Employee { get; set; }

[Display(Name = "Покупатель")]

public string Fiobuyer { get; set; }

}

Листинг класса *Service*

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

namespace Lab6.Models;

public partial class Service

{

public int Id { get; set; }

[Display(Name = "Название")]

public string Name { get; set; }

[Display(Name = "Описание")]

public string Description { get; set; }

[Display(Name = "Стоимость")]

public decimal Price { get; set; }

}

Листинг класса *ContractService*

using System;

using System.Collections.Generic;

namespace Lab6.Models;

public partial class ContractService

{

public int Id { get; set; }

public int ContractId { get; set; }

public int ServiceId { get; set; }

public virtual Contract? Contract { get; set; }

public virtual Service? Service { get; set; }

}

Листинг класса *Lab6Context*

using Lab6.Models;

using Microsoft.EntityFrameworkCore;

namespace Lab6

{

public partial class Lab6Context : DbContext

{

public Lab6Context()

{

}

public Lab6Context(DbContextOptions<Lab6Context> options)

: base(options)

{

}

public virtual DbSet<Contract> Contracts { get; set; }

public virtual DbSet<ContractService> ContractServices { get; set; }

public virtual DbSet<Service> Services { get; set; }

}

}

Листинг класса *HomeController*

using Lab6;

using Lab6.Models;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using System.Diagnostics.Contracts;

namespace lab6.Controllers

{

[ApiController]

[Route("api/[controller]")]

public class HomeController : ControllerBase

{

private readonly Lab6Context \_context;

public HomeController(Lab6Context context)

{

\_context = context;

}

[HttpGet]

public IEnumerable<ContractService> Get()

{

var data = \_context.ContractServices

.Include(e => e.Contract)

.Include(e => e.Service)

.ToList();

return data;

}

[HttpGet("{id}")]

public IActionResult Get(int id)

{

ContractService insuranceCase = \_context.ContractServices.FirstOrDefault(e => e.Id == id);

if (insuranceCase == null)

return NotFound();

return new ObjectResult(insuranceCase);

}

[HttpPost]

public IActionResult Post([FromBody] ContractService insuranceCase)

{

if (insuranceCase == null)

{

return BadRequest();

}

\_context.ContractServices.Add(insuranceCase);

\_context.SaveChanges();

return Ok(insuranceCase);

}

[HttpPut]

public IActionResult Put([FromBody] ContractService insuranceCase)

{

if (insuranceCase == null)

{

return BadRequest();

}

if (!\_context.ContractServices.Any(e => e.Id == insuranceCase.Id))

{

return NotFound();

}

\_context.Update(insuranceCase);

\_context.SaveChanges();

return Ok(insuranceCase);

}

[HttpDelete("{id}")]

public IActionResult Delete(int id)

{

ContractService insuranceCase = \_context.ContractServices.FirstOrDefault(e => e.Id == id);

if (insuranceCase == null)

{

return NotFound();

}

\_context.ContractServices.Remove(insuranceCase);

\_context.SaveChanges();

return Ok(insuranceCase);

}

}

}

Листинг файла *Lab6.html*

﻿

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8" />

<meta name="viewport" content="width=device-width" />

<title>Список договоров и услуг</title>

<link href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.0/css/bootstrap.min.css" rel="stylesheet" />

</head>

<body>

<h2>Список договоров и услуг</h2>

<form name="insuranceCaseForm">

<input type="hidden" name="id" value="0" />

<div class="form-group col-md-5">

<label for="contractId">Номер договора:</label>

<input class="form-control" name="contractId" type="number" />

</div>

<div class="form-group col-md-5">

<label for="serviceId">Номер услуги:</label>

<input class="form-control" name="serviceId" type="number" />

</div>

<div class="panel-body">

<button type="submit" id="submit" class="btn btn-primary">Сохранить</button>

</div>

</form>

<button id="reset" onclick="reset()" class="btn btn-primary">Сбросить</button>

<table class="table table-condensed table-striped col-md-6">

<thead><tr><th>Id</th><th>Дата</th><th>Сумма сделки</th><th>Стоимость услуги</th><th>Продавец</th><th>Покупатель</th><th>Название услуги</th><th>Описание услуги</th><th>Стоимость услуги</th><th></th></tr></thead>

<tbody>

</tbody>

</table>

<script>

async function GetContractServices() {

const response = await fetch("/api/Home", {

method: "GET",

headers: { "Accept": "application/json" }

});

if (response.ok === true) {

const insuranceCases = await response.json();

let rows = document.querySelector("tbody");

insuranceCases.forEach(insuranceCase => {

rows.append(row(insuranceCase));

});

}

}

async function GetContractService(id) {

const response = await fetch("/api/Home/" + id, {

method: "GET",

headers: { "Accept": "application/json" }

});

if (response.ok === true) {

const insuranceCase = await response.json();

const form = document.forms["insuranceCaseForm"];

form.elements["id"].value = insuranceCase.id;

form.elements["contractId"].value = insuranceCase.contractId;

form.elements["serviceId"].value = insuranceCase.serviceId;

}

}

async function CreateContractService(serviceId, contractId) {

const response = await fetch("api/Home", {

method: "POST",

headers: { "Accept": "application/json", "Content-Type": "application/json" },

body: JSON.stringify({

serviceId: serviceId,

contractId: contractId

})

});

if (response.ok === true) {

await response.json();

reset();

}

}

async function EditContractService(id, serviceId, contractId) {

const response = await fetch("api/Home", {

method: "PUT",

headers: { "Accept": "application/json", "Content-Type": "application/json" },

body: JSON.stringify({

id: id,

serviceId: serviceId,

contractId: contractId

})

});

if (response.ok === true) {

await response.json();

reset();

}

}

async function DeleteContractService(id) {

const response = await fetch("/api/Home/" + id, {

method: "DELETE",

headers: { "Accept": "application/json" }

});

if (response.ok === true) {

const insuranceCase = await response.json();

document.querySelector("tr[data-rowid='" + insuranceCase.id + "']").remove();

}

}

function reset() {

document.location.reload();

}

function row(insuranceCase) {

const tr = document.createElement("tr");

tr.setAttribute("data-rowid", insuranceCase.id);

const idTd = document.createElement("td");

idTd.append(insuranceCase.id);

tr.append(idTd);

const dateOfContractTd = document.createElement("td");

dateOfContractTd.append(new Date(insuranceCase.contract.dateOfContract).toDateString());

tr.append(dateOfContractTd);

const dealAmountTd = document.createElement("td");

dealAmountTd.append(insuranceCase.contract.dealAmount);

tr.append(dealAmountTd);

const serviceCostTd = document.createElement("td");

serviceCostTd.append(insuranceCase.contract.serviceCost);

tr.append(serviceCostTd);

const employeeTd = document.createElement("td");

employeeTd.append(insuranceCase.contract.employee);

tr.append(employeeTd);

const fiobuyerTd = document.createElement("td");

fiobuyerTd.append(insuranceCase.contract.fiobuyer);

tr.append(fiobuyerTd);

const nameTd = document.createElement("td");

nameTd.append(insuranceCase.service.name);

tr.append(nameTd);

const descriptionTd = document.createElement("td");

descriptionTd.append(insuranceCase.service.description);

tr.append(descriptionTd);

const priceTd = document.createElement("td");

priceTd.append(insuranceCase.service.price);

tr.append(priceTd);

const linksTd = document.createElement("td");

const editLink = document.createElement("a");

editLink.setAttribute("data-id", insuranceCase.id);

editLink.setAttribute("style", "cursor:pointer;padding:15px;");

editLink.append("Изменить");

editLink.addEventListener("click", e => {

e.preventDefault();

GetContractService(insuranceCase.id);

});

linksTd.append(editLink);

const removeLink = document.createElement("a");

removeLink.setAttribute("data-id", insuranceCase.id);

removeLink.setAttribute("style", "cursor:pointer;padding:15px;");

removeLink.append("Удалить");

removeLink.addEventListener("click", e => {

e.preventDefault();

DeleteContractService(insuranceCase.id);

});

linksTd.append(removeLink);

tr.appendChild(linksTd);

return tr;

}

document.forms["insuranceCaseForm"].addEventListener("submit", e => {

e.preventDefault();

const form = document.forms["insuranceCaseForm"];

const id = form.elements["id"].value;

const serviceId = form.elements["serviceId"].value;

const contractId = form.elements["contractId"].value;

if (id == 0)

CreateContractService(serviceId, contractId);

else

EditContractService(id, serviceId, contractId);

});

GetContractServices();

</script>

</body>

</html>