ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук
Департамент программной инженерии

СОГЛАСОВАНО
Доцент департамента
программной инженерии
факультета компьютерных наук
канд.техн.наук

_____ В. А. Дударев
«_13_» ___мая___ 2022 г.

УТВЕРЖДАЮ
Академический руководитель
образовательной программы
«Программная инженерия»
профессор департамента программной
инженерии, канд. техн. наук

_____ В. В. Шилов
«_13_» ___мая_____2022 г.

Подп. и дата	
Инв. № дубл.	
Взам. Инв. №	
Подп. и дата	
Инв. № подл.	RU.17701729.05.15-01 12 01-1-ЛУ

ВЕБ-СЕРВИС ДЛЯ РАБОТЫ С БАЗОЙ ДАННЫХ ПО СВОЙСТВАМ ХИМИЧЕСКИХ ЭЛЕМЕНТОВ

Текст программы

ЛИСТ УТВЕРЖДЕНИЯ

RU.17701729.05.15-01 12 01-1-ЛУ

Исполнитель
Студент группы БПИ 208
_____ Родионов А. А.
«_13_» ___мая_____2022 г.

Москва 2022

УТВЕРЖДЕН
RU.17701729.05.15-01 12 01-1-ЛУ

ВЕБ-СЕРВИС ДЛЯ РАБОТЫ С БАЗОЙ ДАННЫХ ПО СВОЙСТВАМ ХИМИЧЕСКИХ ЭЛЕМЕНТОВ

ТЕКСТ ПРОГРАММЫ

RU.17701729.05.15-01 12 01-1-ЛУ

Листов: 122

Подп. и дата	
Инв. № дубл.	
Взам. Инв. №	
Подп. и дата	
Инв. № подл.	RU.17701729.05.15-01 12 01-1-ЛУ

Москва 2022

Оглавление

1. УДАЛЕННЫЙ РЕПОЗИТОРИЙ

Основной код программы находится в репозитории: https://github.com/AlexiRod/Elements-Web-Service

2. ТЕКСТ ПРОГРАММЫ

2.1. Element.cs

```
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ElementsClassLibrary
{
   public class Element
   {
      [JsonProperty]
      public int Id { get; private set; }
      [JsonProperty]
      public string Symbol { get; set; }
      [JsonProperty]
      public float AtomicWeight { get; private set; }
      public float Density { get; private set; }
      [JsonProperty]
      public float MeltingTemp { get; private set; }
      [JsonProperty]
      public float BoilingTemp { get; private set; }
   }
}
```

2.2. IProperty.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ElementsClassLibrary
{
   public interface IProperty
   {
      public string Id { get; set; }
      public string ElementSymbol { get; set; }
      public string Name { get; set; }

      public decimal Value { get; set; }
      public string Unit { get; }
      public string? Comments { get; }
      public string? Reference { get; }

      public string GetFormattedValue();
   }
}
```

2.3. LiteratureReference.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
namespace ElementsClassLibrary
{
   public class LiteratureReference
   {
      //[Article], [Source], [Vol], [Num], [DOI], [Page1], [Page2], [Year1], [Year2]
      public string Article { get; set; }
      public string Source { get; set; }
      public string Vol { get; set; }
      public string Num { get; set; }
      public string DOI { get; set; }
      public string Page1 { get; set; }
      public string Page2 { get; set; }
      public string Year1 { get; set; }
      public string Year2 { get; set; }
      public List<string> Authors { get; set; }

      public override string ToString()
      {
         string authors = (Authors == null ? "" : string.Join(" ", Authors));
         string article = (string.IsNullOrEmpty(Article) ? "" : Article);
         string source = (string.IsNullOrEmpty(Source) ? "" : $" / {Source}");
         string vol = (string.IsNullOrEmpty(Vol) ? "" : $", {Vol}");
         string num = (string.IsNullOrEmpty(Num) ? "" : $", {Num}");
         string doi = (string.IsNullOrEmpty(DOI) ? "" : $", {DOI}");
         string page1 = (string.IsNullOrEmpty(Page1) ? "" : $", {Page1}");
         string page2 = (string.IsNullOrEmpty(Page2) ? "" : $"-{Page2}");
         string year1 = (string.IsNullOrEmpty(Year1) ? "" : $", {Year1}");
         string year2 = (string.IsNullOrEmpty(Year2) ? "" : $"-{Year2}");
```

```
            return $"{authors}\n{article}{source}{vol}{num}{doi}{page1}{page2}{year1}{year2}";
        }
    }
}
```

2.4. Property.cs

```
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading.Tasks;

namespace ElementsClassLibrary
{
    public class Property : IProperty
    {
        private string name, unit, elementSymbol;


        [JsonProperty]
        public string Id { get; set; }


        [JsonProperty]
        public string ElementSymbol { get => elementSymbol; set => elementSymbol = value?.Trim(); }


        [JsonProperty]
        public string Name { get => name; set => name = value?.Replace("[&delta;B]",
"&Delta;").Replace("[&gamma;]", "&gamma;").Replace("[&alpha;]", "&alpha;"); }
```

```
        [JsonProperty]
        //[DisplayFormat(DataFormatString = "{0:######,##0.00}", NullDisplayText = "-")]
        public decimal Value { get; set; }

        [JsonProperty]
        public string Unit { get => unit; set => unit = value?.Trim().Replace('*', '\0'); }

        [JsonProperty]
        public string? Comments { get; set; }

        [JsonProperty]
        public string? Reference { get; set; }
        public string GetFormattedValue()
        {
            return Value.ToString("0.###############");
        }
    }
}
```

2.5. RIProperty.cs

```
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ElementsClassLibrary
{
    public class RIProperty : Property
    {
```

```
        [JsonProperty]
        public int Charge { get; private set; }


        [JsonProperty]
        public int? CN { get; private set; }
    }
}
```

2.6. Unit.cs

```
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ElementsClassLibrary
{
    public class Unit
    {
        private string unit1, unit2;
        [JsonProperty]
        public string Unit1 { get => unit1; private set => unit1 = value.Trim(); }
        [JsonProperty]
        public string Unit2 { get => unit2; private set => unit2 = value.Trim(); }
        [JsonProperty]
        public string Equation { get; private set; }



        /// <summary>
        /// Перестановка единиц измерения.
```

```
/// </summary>
public void ChangeUnits()
{
   string temp = Unit1;
   Unit1 = Unit2;
   Unit2 = temp;
}

// From Uint1 to Unit2
/// <summary>
/// Конвертация значения value из ЕИ1 в ЕИ2 по заданному условию.
/// </summary>
/// <param name="value">Значение для конвертации.</param>
/// <returns>Значение после конвертации.</returns>
public decimal Convert(decimal value) {
   decimal[] vals = Equation.Replace('.', ',').Split(new char[] { '*', '/', '+', '-' })
      .Where(x => decimal.TryParse(x, out decimal res)).Select(decimal.Parse).ToArray();

   int i = 0;
   foreach (var symb in Equation)
      switch (symb)
      {
         case '+':
            value += vals[i++];
            break;
         case '-':
            value -= vals[i++];
            break;
         case '*':
            value *= vals[i++];
```

```
                break;
            case '/':
                value /= vals[i++];
                break;
        }
      return value;
    }
  }
}
```

2.7. Program.cs (1)

```
var builder = WebApplication.CreateBuilder(args);

// Add services to the container.
//builder.Services.AddRazorPages().AddRazorRuntimeCompilation();
builder.Services.AddControllersWithViews().AddSessionStateTempDataProvider();
builder.Services.AddDistributedMemoryCache();

builder.Services.AddSession(options =>
{
   options.Cookie.Name = ".AdventureWorks.Session";
   //options.IdleTimeout = TimeSpan.FromMinutes(30);
   options.Cookie.IsEssential = true;
});

var app = builder.Build();

// Configure the HTTP request pipeline.
if (!app.Environment.IsDevelopment())
{
   app.UseExceptionHandler("/Home/Error");
```

```
    // The default HSTS value is 30 days. You may want to change this for production scenarios, see
https://aka.ms/aspnetcore-hsts.
    app.UseHsts();
}

app.UseHttpsRedirection();
app.UseStaticFiles();

app.UseRouting();

app.UseAuthorization();

app.UseSession();

app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Home}/{action=Index}/{id?}");

app.Run();
```

2.8. PropertiesTest.cs

```
using ElementsClassLibrary;
using ElementsWebAPI.Entities;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using System.Collections.Generic;
using System;
using System.Linq;

namespace ElementsTest
{
    [TestClass]
```

```
public class PropertiesTest
{
    private static readonly PropertyRepository rep = new PropertyRepository(
        new DapperContext("server=.; database=phases; Integrated Security=true; TrustServerCertificate=True"));



    [TestMethod]
    public void TestGetLiteratureReference()
    {
        string exp = @"Bakshi B. Berndt M. Brandenburg K. Chen P. Igelnik B. Iwata S. Jackson A. LeClair S. Oxley M.
Pao Y.-H. Villars P.";
        string res = rep.GetLiteratureReference("13823").Result;
        Assert.AreEqual(exp, res.Split('\n')[0]);
    }



    [TestMethod]
    public void TestGetUnitsAndPairs()
    {
        Tuple<Dictionary<string, Unit>, Dictionary<string, string>> res = rep.GetUnitsAndPairs().Result;
        Dictionary<string, Unit> units = res.Item1;


        Assert.AreEqual(units.Count, 7);
        Assert.AreEqual(units["(K)"].Unit1, "(K)");
        Assert.AreEqual(units["(K)"].Unit2, "(C)");
    }



    [TestMethod]
    public void TesGetAllElements()
    {
```

```
    List<Element> elems = rep.GetAllElements().Result.ToList();

    Assert.AreEqual(elems.Count, 100);
    Assert.AreEqual(elems[0].Symbol, "H ");
    Assert.AreEqual(elems[54].Symbol, "Cs");
}

[TestMethod]
public void TestGetAllPropertiesNames()
{
    List<Property> props = rep.GetAllPropertiesNames().Result.ToList();

    Assert.AreEqual(props.Count, 90);
    Assert.AreEqual(props[0].Id, "I26");
    Assert.AreEqual(props[54].Id, "M2");
}

[TestMethod]
public void TestGetAllPropertiesOfElementById()
{
    List<IProperty> props = rep.GetAllPropertiesOfElementById(1).Result.ToList();

    Assert.AreEqual(props.Count, 95);
    Assert.AreEqual(props[0].Value, 0);
    Assert.AreEqual(props[54].Value, 97);
}
```

```
[TestMethod]
public void TestGetGivenPropertiesOfElementById()
{
   List<IProperty> props = rep.GetGivenPropertiesOfElementById(1, new List<string> { "C1", "C2", "I25",
"S14"}).Result.ToList();

   Assert.AreEqual(props.Count, 4);
   Assert.AreEqual(props[0].Value, 14);
   Assert.AreEqual(props[1].Value, 21.15m);
   Assert.AreEqual(props[2].Value, 20.28m);
   Assert.AreEqual(props[3].Value, 0);
}

[TestMethod]
public void TestGetGivenPropertiesValues()
{
   Dictionary<string, IEnumerable<IProperty>> props =
      rep.GetGivenPropertiesValues(new List<string> { "C1", "S14" }, false).Result;

   Assert.AreEqual(props.Count, 2);
   Assert.AreEqual(props["C1"].Count(), 163);
   Assert.AreEqual(props["S14"].Count(), 167);
}

[TestMethod]
public void TestGetGivenPropertiesValuesRec()
{
   Dictionary<string, IEnumerable<IProperty>> props =
      rep.GetGivenPropertiesValues(new List<string> { "C1", "S14" }, true).Result;
```

```
    Assert.AreEqual(props.Count, 2);
    Assert.AreEqual(props["C1"].Count(), 100);
    Assert.AreEqual(props["S14"].Count(), 167);
}


[TestMethod]
public void TestGetGivenPropertiesValuesQuery()
{
    List<IProperty> props = rep.GetGivenPropertyValuesById("C2", true, -1000.31m, 2000.54m).Result.ToList();

    Assert.AreEqual(props.Count, 40);
    Assert.AreEqual(props[0].Value, 20.28m);
}


[TestMethod]
public void TestGetGivenPropertiesValuesWithQuery()
{
    Dictionary<string, IEnumerable<IProperty>> props =
        rep.GetGivenPropertiesValuesWithQuery(new List<string> { "C2", "S15" }, true,
        new List<decimal> { 10.2m, 0.4m }, new List<decimal> { 100, 1.5m }).Result;

    Assert.AreEqual(props.Keys.Count, 3);
    Assert.AreEqual(string.Join(" ", props.Keys), "N O F");
    Assert.AreEqual(props["N"].Count(), 2);
    Assert.AreEqual(props["O"].Count(), 6);
    Assert.AreEqual(props["F"].Count(), 5);
}
```

```
    }
}
```

2.9. Program.cs (2)

```
using ElementsWebAPI.Entities;
using ElementsWebAPI.Interfaces;
using Microsoft.OpenApi.Models;
using System.Reflection;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.
builder.Services.AddSingleton<DapperContext>();
builder.Services.AddScoped<IPropertyRepository, PropertyRepository>();
builder.Services.AddControllers();

// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen(options =>
{
    options.SwaggerDoc("v1", new OpenApiInfo
    {
        Version = "v1",
        Title = "Web Service \"Elements\" API",
        Description = "API ??? ?????? ? ???-???????? ?? ????????? ?????????? ?????????.",
        Contact = new OpenApiContact
        {
            Name = "???????? ???????",
            Url = new Uri("mailto:aarodionov_2@edu.hse.ru")
        },
```

```
        License = new OpenApiLicense
        {
            Name = "???????? ???-??????????",
            Url = new Uri("https://phase.imet-db.ru/elements/main.aspx")
        }
    });
    var xmlFilename = $"{Assembly.GetExecutingAssembly().GetName().Name}.xml";
    options.IncludeXmlComments(Path.Combine(AppContext.BaseDirectory, xmlFilename));
});

var app = builder.Build();
// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}

app.UseHttpsRedirection();
app.UseAuthorization();
app.MapControllers();
app.Run();
```

2.10. HomeController.cs

```
using ElementsClassLibrary;
using ElementsMVCApp.Models;
using Microsoft.AspNetCore.Mvc;
using Newtonsoft.Json;
using System.Diagnostics;
using System.Globalization;
```

```
using System.Net.Http.Headers;
using System.Web;
using Tavis.UriTemplates;
using Microsoft.AspNetCore.Http;

namespace ElementsMVCApp.Controllers
{
   public class HomeController : Controller
   {
      private readonly ILogger<HomeController> _logger;

      public HomeController(ILogger<HomeController> logger)
      {
         _logger = logger;
      }

      public IActionResult Index()
      {
         return View();
      }

      [ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]
      public IActionResult Error()
      {
         return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });
      }
   }
}
```

2.11. PropertiesController (1)

```
using ElementsClassLibrary;
using ElementsMVCApp.Models;
using ElementsMVCApp.Utilities;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Newtonsoft.Json;
using System.Globalization;
using System.Net.Http.Headers;
using Tavis.UriTemplates;
using CsvHelper;
using CsvHelper.Configuration;
using System.Text;

#pragma warning disable CS8600 // Преобразование литерала, допускающего значение NULL или возможного значения NULL в тип, не допускающий значение NULL.

namespace ElementsMVCApp.Controllers
{
    /// <summary>
    /// Контроллер для работы с запросами по свойствам.
    /// </summary>
    public class PropertiesController : Controller
    {

        /// <summary>
        /// Логгер для отлавливания ошибок сервиса.
        /// </summary>
        private readonly ILogger<HomeController> _logger;
        /// <summary>
```

```
/// Http клиент для взаимодействия с API.
/// </summary>
private readonly HttpClient client = new HttpClient();
/// <summary>
/// Настройки сериализации
/// </summary>
private readonly JsonSerializerSettings settings = new JsonSerializerSettings { TypeNameHandling = TypeNameHandling.Auto };
/// <summary>
/// Список всех химических элементов.
/// </summary>
private static List<Element> elements;
/// <summary>
/// Построенная на основе информации из БД таблица Менделеева.
/// </summary>
private static List<List<Element>> table;
/// <summary>
/// Список Id, названий и единиц измерения всех свойств.
/// </summary>
private static List<Property> propertiesNames;
/// <summary>
/// Список пар Id-Название свойства.
/// </summary>
private static List<Tuple<string, string>> propertiesPairs;



/// <summary>
/// Конструктор контроллера с параметрами конфигурации и логгера.
/// </summary>
/// <param name="logger">Экземпляр ILogger.</param>
/// <param name="configuration">Экземпляр IConfiguration.</param>
```

```
public PropertiesController(ILogger<HomeController> logger, IConfiguration configuration)
{
   _logger = logger;
   client.BaseAddress = new Uri(configuration.GetConnectionString("DefaultHost"));
   client.DefaultRequestHeaders.Accept.Clear();
   client.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));
   propertiesNames ??= GetAllPropertiesNames().Result;
   propertiesPairs ??= propertiesNames.Select(x => new Tuple<string, string>(x.Name, x.Id)).ToList();
   elements ??= GetAllElements().Result;
   table ??= TableWorker.MakeTable(elements);
}



/// <summary>
/// Установка настроек единиц измерения по-умолчанию в сессию клиента.
/// </summary>
public async Task<IActionResult> SetSession()
{
   try
   {
      if (HttpContext.Session.Get<Dictionary<string, Unit>>("units") == default ||
            HttpContext.Session.Get<Dictionary<string, string>>("unitpairs") == default)
      {

         HttpResponseMessage response = await client.GetAsync("api/properties/units");
         if (response.IsSuccessStatusCode)
         {
            string stringDeserialized = await response.Content.ReadAsAsync<string>();
            Tuple<Dictionary<string, Unit>, Dictionary<string, string>> pair =
```

```
                JsonConvert.DeserializeObject<Tuple<Dictionary<string, Unit>, Dictionary<string, string>>>
                (stringDeserialized, new JsonSerializerSettings { TypeNameHandling = TypeNameHandling.Auto });
            var units = pair.Item1;
            var unitPairs = pair.Item2;
            HttpContext.Session.Set("units", units);
            HttpContext.Session.Set("unitpairs", unitPairs);
          }
        }
        return Ok();
      }
      catch (Exception ex)
      {
        _logger.LogError(ex.Message + "\n" + ex.StackTrace);
        return StatusCode(500, ex.Message + "\n" + ex.StackTrace);
      }
    }


    /// <summary>
    /// Метод установки настроек при первом обращении к приложению.
    /// </summary>
    /// <returns></returns>
    public IActionResult SetParams()
    {
      return SetSession().Result;
    }


    /// <summary>
    /// Главная страница контроллера
```

```
/// </summary>
/// <returns>Переадресация на главную HomeController</returns>
public IActionResult Index()
{
   return RedirectToAction("Index", "Home");
}



/// <summary>
/// Получение из сессии выбранных настроек пользователя и отображение страницы с их изменением.
/// </summary>
[HttpGet]
public IActionResult Settings()
{
   try
   {
      Dictionary<string, string> unitPairs = HttpContext.Session.Get<Dictionary<string, string>>("unitpairs");
      Dictionary<string, Unit> units = HttpContext.Session.Get<Dictionary<string, Unit>>("units");


      if (units == null || unitPairs == null)
         return NoContent(); // this won't happen


      List<Tuple<UnitViewModel, UnitViewModel>> settingPairs = new List<Tuple<UnitViewModel,
UnitViewModel>>();
      foreach (var pair in unitPairs)
      {
         bool isFirstSelected = units[pair.Key].Unit1.Trim() == pair.Key.Trim();
         Tuple<UnitViewModel, UnitViewModel> unitViewModel = new Tuple<UnitViewModel,
UnitViewModel>(
            new UnitViewModel(pair.Key, isFirstSelected), new UnitViewModel(pair.Value, !isFirstSelected));
```

```
            settingPairs.Add(unitViewModel);
        }

        return View(new SettingsViewModel() { Units = settingPairs });
    }
    catch (Exception ex)
    {
        _logger.LogError(ex.Message + "\n" + ex.StackTrace);
        return StatusCode(500, ex.Message + "\n" + ex.StackTrace);
    }
}


/// <summary>
/// Подтверждение выбранных пользователем настроек и изменение их в сессии.
/// </summary>
/// <param name="model">Выбранные настройки.</param>
public IActionResult Settings(SettingsViewModel model)
{
    UpdateUnits(model.SelectedUnits);
    return View("SettingsChanged");
}


/// <summary>
/// Отображение информации об источнике по заданной ссылке.
/// </summary>
/// <param name="id">Ссылка на источник.</param>
public IActionResult LitRef(string? id)
{
```

```
    try
    {
        if (string.IsNullOrEmpty(id))
            return NoContent();

        string res = GetLiteratureReference(id).Result;
        ViewBag.Lit = res;
        return View();
    }
    catch (Exception ex)
    {
        _logger.LogError(ex.Message + "\n" + ex.StackTrace);
        return StatusCode(500, ex.Message + "\n" + ex.StackTrace);
    }
}



/// <summary>
/// Отображение страницы с интерактивной таблицей Менделеева.
/// </summary>
/// <param name="mode">Режим работы пользователя.</param>
public IActionResult Mendel(int? mode)
{
    try
    {
        ViewBag.Mode = mode;
        return View(table);
    }
    catch (Exception ex)
```

```
    {
        _logger.LogError(ex.Message + "\n" + ex.StackTrace);
        return StatusCode(500, ex.Message + "\n" + ex.StackTrace);
    }
}



/// <summary>
/// Отображение интерактивного списка для выбора свойств.
/// </summary>
/// <param name="mode">Режим работы пользователя.</param>
/// <param name="elemId">Id выбранного элемента (Опционально)</param>
[HttpGet("Properties/PropsList/{mode}/{elemId?}")]
public IActionResult PropsList(int? mode, int? elemId)
{
    try
    {
        List<SelectListItem> selectedList = propertiesPairs.Select(
            x => new SelectListItem() { Text = x.Item1, Value = x.Item2 }).ToList();


        ViewBag.Mode = mode;
        ViewBag.ElemId = elemId;
        return View(new PropertyListViewModel() { AvailableProps = selectedList });
    }
    catch (Exception ex)
    {
        _logger.LogError(ex.Message + "\n" + ex.StackTrace);
        return StatusCode(500, ex.Message + "\n" + ex.StackTrace);
    }
}
```

```
/// <summary>
/// Переадресация на нужный метод после выбора свойств из списка.
/// </summary>
/// <param name="mode">Режим работы пользователя.</param>
/// <param name="elemId">Id выбранного элемента (Опционально)</param>
/// <param name="propsList">Список выбранных свойств.</param>
[HttpPost("Properties/PropsList/{mode}/{elemId?}")]
public IActionResult PropsList(int? mode, int? elemId, PropertyListViewModel propsList)
{
    try
    {
        if (ModelState.IsValid && mode != null)
        {
            switch (mode.Value)
            {
                case 2:
                    return GivenProps(elemId, propsList);
                case 3:
                    return PropsValues(propsList, 3);
                case 4:
                    return RecProps(propsList);
                case 5:
                    return QueryProps(propsList);
            }
        }
        return View();
    }
    catch (Exception ex)
```

```
        {
            _logger.LogError(ex.Message + "\n" + ex.StackTrace);
            return StatusCode(500, ex.Message + "\n" + ex.StackTrace);
        }
    }


    /// <summary>
    /// Отображение списка свойств заданного элемента.
    /// </summary>
    /// <param name="props">Список выбранных свойств.</param>
    /// <param name="elemId">Id выбранного элемента.</param>
    /// <param name="element">Символ выбранного элемента.</param>
    /// <param name="mode">Режим работы пользователя.</param>
    public IActionResult ElementPropsList(List<IProperty> props, int elemId, string element, int mode)
    {
        try
        {
            List<Property> properties = new List<Property>();
            List<RIProperty> riproperties = new List<RIProperty>();
            props.ForEach(x => { if (x is RIProperty) riproperties.Add((RIProperty)x); else properties.Add((Property)x);
});
            ViewBag.Properties = properties;
            ViewBag.RIProperties = riproperties;
            ViewBag.ElemId = elemId;
            ViewBag.Element = element;
            ViewBag.Mode = mode;

            return View("ElementPropsList");
        }
        catch (Exception ex)
```

```
        {
            _logger.LogError(ex.Message + "\n" + ex.StackTrace);
            return StatusCode(500, ex.Message + "\n" + ex.StackTrace);
        }
    }


    /// <summary>
    /// Отображение частичного представления при выборе свойства из интерактивного выпадающего списка.
    /// </summary>
    /// <param name="id">Id выбранного свойства.</param>
    /// <param name="mode">Режим работы пользователя.</param>
    public ActionResult Partial(string id, int? mode)
    {
        try
        {
            if (id == null || mode == null)
                return NotFound();


            List<IProperty> model = mode == 3 ? GetGivenPropValuesById(id).Result : GetGivenPropOfGivenElemRec(id).Result;
            if (model == null || model.Count == 0)
                return NotFound();


            ViewBag.Mode = mode.Value;
            return PartialView("Partial/PartialValsTable", model);
        }
        catch (Exception ex)
        {
```

```
        _logger.LogError(ex.Message + "\n" + ex.StackTrace);
        return StatusCode(500, ex.Message + "\n" + ex.StackTrace);
    }
}



/// <summary>
/// Получение всех свойств выбранного элемента (Режим 1).
/// </summary>
/// <param name="id">Id элемента.</param>
public IActionResult AllProps(int? id)
{
    try
    {
        ViewBag.Mode = 1;
        if (id == null)
            return View("ElementNotFound");

        var props = GetAllPropsOfGivenElem(id.Value).Result;
        if (props == null || props.Count == 0)
            return View("ElementNotFound");

        return ElementPropsList(props, id.Value, props[0].ElementSymbol, 1);
    }
    catch (Exception ex)
    {
        _logger.LogError(ex.Message + "\n" + ex.StackTrace);
        return StatusCode(500, ex.Message + "\n" + ex.StackTrace);
    }
```

```
}


/// <summary>
/// Получение значений выбранных из списка свойств выбранного элемента (Режим 2).
/// </summary>
/// <param name="elemId">Id элемента.</param>
/// <param name="propsList">Список выбранных свойств/</param>
public IActionResult GivenProps(int? elemId, PropertyListViewModel propsList)
{
   try
   {
      ViewBag.Mode = 2;
      if (!elemId.HasValue || elemId.Value < 1 || elemId.Value > elements.Count || propsList == null)
         return View("ElementNotFound");

      List<string> propIds = propsList.SelectedProps.ToList();
      var props = GetGivenPropsOfGivenElem(elemId.Value, propIds).Result;
      return ElementPropsList(props, elemId.Value, elements[elemId.Value - 1].Symbol, 2);
   }
   catch (Exception ex)
   {
      _logger.LogError(ex.Message + "\n" + ex.StackTrace);
      return StatusCode(500, ex.Message + "\n" + ex.StackTrace);
   }
}


/// <summary>
/// Получение значений выбранных из списка свойств для всех элементов таблицы Менделеева
(Режим 3).
```

```
/// </summary>
/// <param name="propsList">Список выбранных свойств.</param>
/// <param name="mode">Режим работы пользователя.</param>
public IActionResult PropsValues(PropertyListViewModel propsList, int mode)
{
    try
    {
        if (propsList == null)
            return NotFound();

        string firstId = "";
        Dictionary<string, string> dict = new Dictionary<string, string>();
        foreach (var id in propsList.SelectedProps)
        {
            firstId = firstId == "" ? id : firstId;
            dict[id] = propertiesNames.FirstOrDefault(x => x.Id == id)?.Name;
        }

        ViewBag.Dict = dict;
        ViewBag.FirstId = firstId;
        ViewBag.Mode = mode;
        return View("PropsValues");
    }
    catch (Exception ex)
    {
        _logger.LogError(ex.Message + "\n" + ex.StackTrace);
        return StatusCode(500, ex.Message + "\n" + ex.StackTrace);
    }
}
```

```
/// <summary>
/// Отображение таблицы рекомендованных значений выбранных свойств (Режим 4).
/// </summary>
/// <param name="propsList">Список выбранных свйоств.</param>
[HttpGet]
public IActionResult RecProps(PropertyListViewModel propsList)
{
   try
   {
      if (propsList == null)
         return NotFound();

      Dictionary<string, Unit> units = HttpContext.Session.Get<Dictionary<string, Unit>>("units");
      Dictionary<string, IEnumerable<IProperty>> dict =
GetGivenPropsOfGivenElemRec(propsList.SelectedProps.ToList()).Result;

      var props = propertiesNames.Where(y => propsList.SelectedProps.Contains(y.Id)).
         Select(x => x.Name + (string.IsNullOrEmpty(x.Unit) ? "" :
         $" {(units?.ContainsKey(x.Unit) == true ? units?[x.Unit].Unit1 : "")}")).ToList();

      var propNames = propertiesNames.Where(y => propsList.SelectedProps.Contains(y.Id))
         .Select(x => x.Name).ToList();

      PropertyTableViewModel table = new PropertyTableViewModel(
         elements.Select(x => x.Symbol.Trim()).ToList(), props, propNames);

      foreach (var list in dict.Values)
         list.DistinctBy(x => x.ElementSymbol).ToList().
            ForEach(x => table.AddValue(x.ElementSymbol.Trim(), x.Name, x.Value));
```

```
        TempData["table"] = JsonConvert.SerializeObject(table);
        return View("RecProps", table);
    }
    catch (Exception ex)
    {
        _logger.LogError(ex.Message + "\n" + ex.StackTrace);
        return StatusCode(500, ex.Message + "\n" + ex.StackTrace);
    }
}


/// <summary>
/// Отображение элементов со свойствами, значения которых попадают в заданные диапазоны.
/// </summary>
/// <param name="query"></param>
/// <returns></returns>
[HttpPost]
[ValidateAntiForgeryToken]
public IActionResult Query(QueryListViewModel query)
{
    try
    {
        if (query == null)
            return NotFound();

        List<string> propIds = new List<string>();
        List<decimal> lefts = new List<decimal>();
        List<decimal> rights = new List<decimal>();

        query.Queries.ForEach(x => { propIds.Add(x.PropId); lefts.Add(x.Left); rights.Add(x.Right); });
```

```
        var dict = GetGivenPropsOfGivenElemQuery(propIds, lefts, rights).Result;

        return View("QueryResult", dict);
    }
    catch (Exception ex)
    {
        _logger.LogError(ex.Message + "\n" + ex.StackTrace);
        return StatusCode(500, ex.Message + "\n" + ex.StackTrace);
    }
}

/// <summary>
/// Выбор диапазонов значений для выбранных свойств.
/// </summary>
/// <param name="propsList">Список выбранных свойств.</param>
public IActionResult QueryProps(PropertyListViewModel propsList)
{
    try
    {
        if (propsList == null)
            return NotFound();

        List<string> units = new List<string>();
        var props = propertiesNames.Where(y => propsList.SelectedProps.Contains(y.Id)).ToList();
        List<QueryViewModel> queries = new List<QueryViewModel>();
        foreach (var item in props)
        {
            queries.Add(new QueryViewModel(item.Id, item.Name));
            units.Add(item.Unit?.Trim('(', ')', '[', ']'));
```

```
            }

            ViewBag.Units = units;
            return View("Query", new QueryListViewModel() { Queries = queries });
        }
        catch (Exception ex)
        {
            _logger.LogError(ex.Message + "\n" + ex.StackTrace);
            return StatusCode(500, ex.Message + "\n" + ex.StackTrace);
        }
    }


    /// <summary>
    /// Экспортирование данных в формат CSV
    /// </summary>
    /// <returns>CSV файл для скачивания.</returns>
    public IActionResult Export()
    {
        try
        {
            if (!TempData.ContainsKey("table"))
                return NotFound();

            PropertyTableViewModel table =
JsonConvert.DeserializeObject<PropertyTableViewModel>(TempData["table"].ToString());

            if (table == null)
                return NotFound();

            using (var ms = new MemoryStream())
```

```
        {
            using (var sw = new StreamWriter(stream: ms, encoding: new UTF8Encoding(true)))
            {
                using (var cw = new CsvWriter(sw, CultureInfo.InvariantCulture))
                {
                    cw.WriteField("Element");
                    table.Properties.ForEach(x => cw.WriteField(x));
                    cw.NextRecord();
                    for (int i = 0; i < table.Elements.Count; i++)
                    {
                        cw.WriteField(table.Elements[i]);
                        table.Values[i].ForEach(x => cw.WriteField(x.ToString("0.###############",
CultureInfo.InvariantCulture)));
                        cw.NextRecord();
                    }
                }
                return File(ms.ToArray(), "text/csv", $"props_{DateTime.Now.Ticks}.csv");
            }
        }
    }
    catch (Exception ex)
    {
        _logger.LogError(ex.Message + "\n" + ex.StackTrace);
        return StatusCode(500, ex.Message + "\n" + ex.StackTrace);
    }
}
```

```
/// <summary>
```

```
/// Изменение настроек единиц измерения.
/// </summary>
/// <param name="newUnits">Новые единицы измерения.</param>
private void UpdateUnits(List<string> newUnits)
{
   Dictionary<string, Unit> units = HttpContext.Session.Get<Dictionary<string, Unit>>("units");
   if (units == default) // This won't happen...
   {
      var res = SetSession().Result;
      units = HttpContext.Session.Get<Dictionary<string, Unit>>("units");
   }

   foreach (string unit in newUnits)
   {
      var found = units.FirstOrDefault(x => x.Value.Unit2 == unit.Trim());
      if (/*unitPairs.ContainsKey(unit) ||*/ found.Value == null)
         continue;

      found.Value.ChangeUnits();
   }

   HttpContext.Session.Set("units", units);
}


/// <summary>
/// Получение библиотечной информации по заданной литературной ссылке.
/// </summary>
/// <param name="id">Id ссылки.</param>
/// <returns>Строка с информацией по данному ресурсу.</returns>
```

```
private async Task<string> GetLiteratureReference(string id)
{
   string res = "No data";
   HttpResponseMessage response = await client.GetAsync($"api/properties/refs/{id}");
   if (response.IsSuccessStatusCode)
      res = await response.Content.ReadAsAsync<string>();
   return res;
}



/// <summary>
/// Получение всех элементов с основыми свойствами.
/// </summary>
/// <returns>Список всех элементов таблицы Менделеева.</returns>
private async Task<List<Element>> GetAllElements()
{
   List<Element> elems = new List<Element>();
   HttpResponseMessage response = await client.GetAsync("api/properties/elems");
   if (response.IsSuccessStatusCode)
   {
      string stringDeserialized = await response.Content.ReadAsAsync<string>();
      elems = JsonConvert.DeserializeObject<List<Element>>(stringDeserialized,
         new JsonSerializerSettings { TypeNameHandling = TypeNameHandling.Auto });
   }
   return elems;
}



/// <summary>
/// Получение всех свойств с их Id, названием и типом.
```

```
/// </summary>
/// <returns>Список всех свойств в формате Id, Название, Тип.</returns>
private async Task<List<Property>> GetAllPropertiesNames()
{
   List<Property> props = new List<Property>();
   HttpResponseMessage response = await client.GetAsync("api/properties/propsnames");
   if (response.IsSuccessStatusCode)
   {
      string stringDeserialized = await response.Content.ReadAsAsync<string>();
      props = JsonConvert.DeserializeObject<List<Property>>(stringDeserialized,
         new JsonSerializerSettings { TypeNameHandling = TypeNameHandling.Auto });
   }
   return props;
}



/// <summary>
/// Взаимодействие с API (Режим 1).
/// </summary>
private async Task<List<IProperty>> GetAllPropsOfGivenElem(int id)
{
   return await GetPropsFromApi($"api/properties/allprops/{id}");
}



/// <summary>
/// Взаимодействие с API (Подрежим 2).
/// </summary>
private async Task<List<IProperty>> GetGivenPropOfGivenElem(int elemId, string propId)
{
```

```
    var url = new UriTemplate("api/properties/givenprops/{elemId},{propId}")
    .AddParameter("elemId", elemId).AddParameter("propId", propId).Resolve();

    return await GetPropsFromApi(url);
}

/// <summary>
/// Взаимодействие с API (Режим 2).
/// </summary>
private async Task<List<IProperty>> GetGivenPropsOfGivenElem(int elemId, List<string> propsId)
{
    var url = new UriTemplate($"api/properties/givenprops/{elemId}?" +
        $"{string.Join('&', propsId.Select(x => "propsId=" + x))}").Resolve();

    return await GetPropsFromApi(url);
}


/// <summary>
/// Взаимодействие с API (Подрежим 3).
/// </summary>
private async Task<List<IProperty>> GetGivenPropValuesById(string propId)
{
    return await GetPropValuesByPath(propId, "propsvalues");
}


/// <summary>
/// Взаимодействие с API (Режим 3).
/// </summary>
```

```
    private async Task<Dictionary<string, IEnumerable<IProperty>>> GetGivenPropsValues(List<string> propsId)
    {
       return await GetGivenPropsValuesByPath(propsId, "propsvalues");
    }


    /// <summary>
    /// Взаимодействие с API (Подрежим 4).
    /// </summary>
    private async Task<List<IProperty>> GetGivenPropOfGivenElemRec(string propId)
    {
       return await GetPropValuesByPath(propId, "recprops");
    }


    /// <summary>
    /// Взаимодействие с API (Режим 4).
    /// </summary>
    private async Task<Dictionary<string, IEnumerable<IProperty>>> GetGivenPropsOfGivenElemRec(List<string>
propsId)
    {
       return await GetGivenPropsValuesByPath(propsId, "recprops");
    }


    /// <summary>
    /// Взаимодействие с API (Подрежим 5).
    /// </summary>
    private async Task<List<IProperty>> GetGivenPropOfGivenElemQuery(string propId, decimal left, decimal
right)
    {
```

```
        var url = $"api/properties/queryprops/{propId}/{left.ToString().Replace(',', '.')}/{right.ToString().Replace(',',
'.')}/";


        return await GetPropsFromApi(url);
    }



    /// <summary>
    /// Взаимодействие с API (Режим 5).
    /// </summary>
    private async Task<Dictionary<string, IEnumerable<IProperty>>> GetGivenPropsOfGivenElemQuery(
        List<string> propsId, List<decimal> lefts, List<decimal> rights)
    {
        Dictionary<string, IEnumerable<IProperty>> propsValues = new Dictionary<string,
IEnumerable<IProperty>>();
        List<string> leftsStr = lefts.Select(x => x.ToString(CultureInfo.InvariantCulture)).ToList();
        List<string> rightsStr = rights.Select(x => x.ToString(CultureInfo.InvariantCulture)).ToList();

        var url = new UriTemplate("api/properties/queryprops/{?propsId*,lefts*,rights*}")
            .AddParameters(new { propsId = propsId, lefts = leftsStr, rights = rightsStr }).Resolve();

        HttpResponseMessage response = await client.GetAsync(url);
        if (response.IsSuccessStatusCode)
        {
            string stringDeserialized = await response.Content.ReadAsAsync<string>();
            propsValues = JsonConvert.DeserializeObject<Dictionary<string,
IEnumerable<IProperty>>>(stringDeserialized,
                new JsonSerializerSettings { TypeNameHandling = TypeNameHandling.Auto });
            foreach (IEnumerable<IProperty> list in propsValues.Values)
                ConvertPropertiesUnits(list.ToList());
        }
```

```
    return propsValues;
}



/// <summary>
/// Вспомогательный метод для вызова API.
/// </summary>
/// <param name="url"></param>
/// <returns></returns>
private async Task<List<IProperty>> GetPropsFromApi(string url)
{
    List<IProperty> props = new List<IProperty>();
    HttpResponseMessage response = await client.GetAsync(url);
    if (response.IsSuccessStatusCode)
    {
        string stringDeserialized = await response.Content.ReadAsAsync<string>();
        props = JsonConvert.DeserializeObject<List<IProperty>>(stringDeserialized,
            new JsonSerializerSettings { TypeNameHandling = TypeNameHandling.Auto });
        ConvertPropertiesUnits(props);
    }
    return props;
}



/// <summary>
/// Вспомогательный метод для подрежимов 3-4.
/// </summary>
private async Task<List<IProperty>> GetPropValuesByPath(string propId, string path)
{
```

```
        var url = new UriTemplate($"api/properties/{path}/{{propId}}").AddParameter("propId", propId).Resolve();


        return await GetPropsFromApi(url);
    }



    /// <summary>
    /// Вспомогательный метод для режимов 3-4.
    /// </summary>
    private async Task<Dictionary<string, IEnumerable<IProperty>>> GetGivenPropsValuesByPath(List<string> propsId, string path)
    {
        Dictionary<string, IEnumerable<IProperty>> propsValues = new Dictionary<string, IEnumerable<IProperty>>();


        var url = new UriTemplate($"api/properties/{path}/{{?propsId*}}").AddParameter("propsId", propsId).Resolve();


        HttpResponseMessage response = await client.GetAsync(url);
        if (response.IsSuccessStatusCode)
        {
            string stringDeserialized = await response.Content.ReadAsAsync<string>();
            propsValues = JsonConvert.DeserializeObject<Dictionary<string, IEnumerable<IProperty>>>(stringDeserialized,
                new JsonSerializerSettings { TypeNameHandling = TypeNameHandling.Auto });
            foreach (IEnumerable<IProperty> list in propsValues.Values)
                ConvertPropertiesUnits(list.ToList());
        }
        return propsValues;
    }
```

```
    /// <summary>
    /// Вспомогательный метод конвертации величин свойств.
    /// </summary>
    private void ConvertPropertiesUnits(List<IProperty> props)
    {
      Dictionary<string, Unit> units = HttpContext.Session.Get<Dictionary<string, Unit>>("units");
      if (units == default)
        return;

      foreach (Property prop in props)
      {
        if (prop.Unit == null)
          continue;

        var unit = units.GetValueOrDefault(prop.Unit);
        if (unit != null && unit.Unit2 == prop.Unit)
        {
          prop.Value = unit.Convert(prop.Value);
          prop.Unit = unit.Unit1;
        }
      }
    }
  }
}
```

#pragma warning restore CS8600 // Преобразование литерала, допускающего значение NULL или возможного значения NULL в тип, не допускающий значение NULL.

2.12. ErrorViewModel.cs

```
namespace ElementsMVCApp.Models
{
```

```
    public class ErrorViewModel
    {
        public string? RequestId { get; set; }

        public bool ShowRequestId => !string.IsNullOrEmpty(RequestId);
    }
}
```

2.13. PropertyListViewModel.cs

```
using Microsoft.AspNetCore.Mvc.Rendering;

namespace ElementsMVCApp.Models
{
    public class PropertyListViewModel
    {
        public IList<string> SelectedProps { get; set; }
        public IList<SelectListItem> AvailableProps { get; set; }

        public PropertyListViewModel()
        {
            SelectedProps = new List<string>();
            AvailableProps = new List<SelectListItem>();
        }
    }
}
```

2.14. PropertyTableViewModel.cs

```
namespace ElementsMVCApp.Models
{
    public class PropertyTableViewModel
    {
```

```
        public List<string> Elements { get; set; }
        public List<string> Properties { get; set; }
        public List<string> PropertiesNames { get; set; }
        public List<List<decimal>> Values { get; set; }

        public PropertyTableViewModel(List<string> elements, List<string> properties, List<string> propsNames)
        {
            Elements = elements;
            Properties = properties;
            PropertiesNames = propsNames;
            Values = new List<List<decimal>>();
            foreach (var item in elements)
                Values.Add(new List<decimal>(new decimal[properties.Count]));
        }
        public PropertyTableViewModel() { }

        public void AddValue(string elem, string prop, decimal value)
        {
            int i = Elements.IndexOf(elem);
            int j = PropertiesNames.IndexOf(prop);
            Values[i][j] = value;
        }
    }
}
```

2.15. QueryListViewModel.cs

```
namespace ElementsMVCApp.Models
{
    public class QueryListViewModel
    {
        public  List<QueryViewModel> Queries { get; set; }
```

```
    }
}
```

2.16. QueryViewModel.cs

```
using System.ComponentModel.DataAnnotations;

namespace ElementsMVCApp.Models
{
    public class QueryViewModel
    {
        public string PropId { get; set; }
        public string Property { get; set; }

        [RegularExpression(@"^-?\d+([,]\d{1,})?$",
            ErrorMessage = "Left border should be a numver separated by comma")]
        public decimal Left { get; set; }

        [RegularExpression(@"-?\d+([,]\d{1,})?",
            ErrorMessage = "Right border should be a numver separated by comma")]
        [Range(int.MinValue, int.MaxValue, ErrorMessage = "")]
        public decimal Right { get; set; }

        public QueryViewModel(string propId, string prop)
        {
            Property = prop;
            PropId = propId;
            Left = decimal.MinValue;
            Right = decimal.MaxValue;
        }
        public QueryViewModel()
```

```
        {
            Left = decimal.MinValue;
            Right = decimal.MaxValue;
        }
    }
}
```

2.17. SettingsViewModel.cs

```
namespace ElementsMVCApp.Models
{
    public class SettingsViewModel
    {
        public List<Tuple<UnitViewModel, UnitViewModel>> Units { set; get; }
        public List<string> SelectedUnits { get; set; }
    }
}
```

2.18. UnitViewModel.cs

```
using ElementsClassLibrary;


namespace ElementsMVCApp.Models
{
    public class UnitViewModel
    {
        public string Id { get; set; }
        public bool IsSelected { get; set; }


        public UnitViewModel(string id, bool isSelected)
        {
            Id = id;
            IsSelected = isSelected;
        }
```

```
    }
}
```

2.19. SessionExtensions.cs

```
using Newtonsoft.Json;


namespace ElementsMVCApp.Utilities
{
    public static class SessionExtensions
    {
        private static readonly JsonSerializerSettings settings = new JsonSerializerSettings
        {
            TypeNameHandling = TypeNameHandling.Auto,
            Formatting = Formatting.Indented
        };


        public static void Set<T>(this ISession session, string key, T value)
        {
            var t = JsonConvert.SerializeObject(value, settings);
            session.SetString(key, t);
        }


        public static T? Get<T>(this ISession session, string key)
        {
            string? value = session.GetString(key);
            var d = value == null ? default : JsonConvert.DeserializeObject<T>(value,
                new JsonSerializerSettings { TypeNameHandling = TypeNameHandling.Auto });
            return value == null ? default : d;
        }
    }
}
```

2.20. TableWorker.cs

```
using ElementsClassLibrary;


namespace ElementsMVCApp.Utilities
{
   public class TableWorker
   {
      // Ac Ku ?
      private static string[][] table = new string[][] {
         new string[] { "H", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "He" },
         new string[] { "Li","Be", "", "", "", "", "", "", "", "", "", "", "B", "C","N","O","F","Ne" },
         new string[] { "Na", "Mg", "", "", "", "", "", "", "", "", "", "", "Al","Si","P","S","Cl","Ar" },
         new string[] { "K","Ca","Sc","Ti","V","Cr","Mn","Fe","Co","Ni", "Cu","Zn","Ga","Ge","As","Se","Br","Kr" },
         new string[] { "Rb","Sr","Y","Zr","Nb","Mo","Tc","Ru","Rh","Pd", "Ag","Cd","In","Sn","Sb","Te","I","Xe" },
         new string[] { "Cs", "Ba", "", "Hf", "Ta", "W", "Re", "Os", "Ir", "Pt", "Au", "Hg", "Tl", "Pb", "Bi", "Po", "At",
"Rn" },
         new string[] { "Fr", "Ra", "", "Rf", "Db", "Sg", "Bh", "Hs", "Mt", "Ds", "Rg", "Cn", "Uut", "Fl", "Uup", "Lv",
"Uus", "Uuo"},
         new string[] { },
         new string[] { "", "", "", "La", "Ce","Pr","Nd","Pm","Sm","Eu","Gd","Tb","Dy","Ho","Er","Tm","Yb", "Lu" },
         new string[] { "", "", "", "Ac","Th","Pa","U","Np","Pu","Am","Cm","Bk","Cf","Es","Fm","Md","No","Lr" }
      };

      public static List<List<Element>> MakeTable(List<Element> list)
      {
         List<List<Element>> elements = new List<List<Element>>();
         for (int i = 0; i < table.Length; i++)
         {
            List<Element> row = new List<Element>();
            for (int j = 0; j < table[i].Length; j++)
            {
```

```
            if (string.IsNullOrEmpty(table[i][j]))
            {
               row.Add(null);
               continue;
            }
            var el = list.FirstOrDefault(x => x.Symbol.Trim() == table[i][j]);
            if (el == null)
               el = new Element() { Symbol = table[i][j] };
            row.Add(el);
         }
         elements.Add(row);
      }

      return elements;
    }
  }
}
```

2.21. PropertiesController (2)

```
using Microsoft.AspNetCore.Mvc;
using ElementsWebAPI.Interfaces;
using ElementsWebAPI.Entities;
using ElementsClassLibrary;
using Newtonsoft.Json;

namespace ElementsWebAPI.Controllers
{
  //
  // Tests:
  //
```

```
  //
https://localhost:7011/api/properties/queryprops?propsId=S12&propsId=S11&propsId=S15&lefts=0&lefts=0&left
s=0.1&rights=2&rights=2&rights=1
  //
https://localhost:7011/api/properties/queryprops?propsId=A6&propsId=C1&propsId=C2&lefts=10&lefts=50&left
s=50&rights=100&rights=2000&rights=3000



  [ApiController]
  [Route("api/properties")]
  //[Produces("application/json")]
  public class PropertiesController : Controller
  {
    #region Глобальные переменные и конструктор

    /// <summary>
    /// Логгер для отлавливания ошибок сервера.
    /// </summary>
    private readonly ILogger _logger;
    /// <summary>
    /// Репозиторий для работы с базой данных.
    /// </summary>
    private readonly IPropertyRepository _propertyRepository;

    /// <summary>
    /// Настройки сериализатора с автоматическим TypeNameHandling
    /// </summary>
    private readonly JsonSerializerSettings settings = new JsonSerializerSettings
    {
      TypeNameHandling = TypeNameHandling.Auto,
      Formatting = Formatting.Indented
    };
```

```
/// <summary>
/// Настройки сериализатора с объектным TypeNameHandling
/// </summary>
private readonly JsonSerializerSettings objectSettings = new JsonSerializerSettings
{
   TypeNameHandling = TypeNameHandling.Objects,
   TypeNameAssemblyFormat = System.Runtime.Serialization.Formatters.FormatterAssemblyStyle.Simple,
   Formatting = Formatting.Indented
};



public PropertiesController(ILogger<PropertiesController> logger, IPropertyRepository propertyRepository)
{
   _logger = logger;
   _propertyRepository = propertyRepository;
}

#endregion



#region Вспомогательные методы контроллера


/// <summary>
/// Вспомогательный метод для получения всех значений выбранного свойства с опцией рекомендуемых и диапозона.
/// </summary>
/// <param name="propId">Id свойства.</param>
/// <param name="isRecomended">Флаг, показывающий, необходимо ли взять все значения или только рекомендуемые.</param>
```

```
    /// <param name="left">Левая граница диапозона (опционально).</param>
    /// <param name="right">Правая граница диапозона (опционально).</param>
    /// <returns>Список (в зависимости от флага) значений свойства.</returns>
    private async Task<IActionResult> GivenPropValuesByIdOrRecQuery(
      string propId, bool isRecomended, decimal? left = null, decimal? right = null)
    {
      try
      {
        var prop = await _propertyRepository.GetGivenPropertyValuesById(propId, isRecomended, left, right);
        var res = JsonConvert.SerializeObject(prop, objectSettings);
        return Ok(res);
      }
      catch (Exception ex)
      {
        _logger.LogError(ex.Message);
        return StatusCode(500, ex.Message);
      }

    }

    /// <summary>
    /// Вспомогательный метод для получения всех значений выбранных из списка свойств с опцией рекомендуемых.
    /// </summary>
    /// <param name="propsId">Список Id выбранных свойств.</param>
    /// <param name="isRecomended">Флаг, показывающий, необходимо ли взять все значения или только рекомендуемые.</param>
    /// <returns>Словарь Id - List<Property> (в зависимости от флага).</returns>
    private async Task<IActionResult> GivenPropsValuesOrRecQuery(
      List<string> propsId, bool isRecomended)
    {
```

```
        try
        {
            if (propsId == null)
                return BadRequest("Список Id свойств обязательно должен быть указан.");
            var dict = await _propertyRepository.GetGivenPropertiesValues(propsId, isRecomended);


            var res = JsonConvert.SerializeObject(dict, new JsonSerializerSettings
            {
                TypeNameHandling = TypeNameHandling.All,
                TypeNameAssemblyFormat =
System.Runtime.Serialization.Formatters.FormatterAssemblyStyle.Simple,
                Formatting = Formatting.Indented
            }).Replace("[[ElementsClassLibrary.Property, ElementsClassLibrary]]",
            "[[ElementsClassLibrary.IProperty, ElementsClassLibrary]]");


            return Ok(res);
        }
        catch (Exception ex)
        {
            _logger.LogError(ex.Message);
            return StatusCode(500, ex.Message);
        }
    }


    #endregion




    #region Get-запросы на получение данных
```

```
/// <summary>
/// Получение библиотечной информации по заданной литературной ссылке.
/// </summary>
/// <param name="id">Id ссылки.</param>
/// <returns>Строка с информацией по данному ресурсу.</returns>
/// <response code="200">Успешное выполнение.</response>
/// <response code="500">Ошибка на стороне сервиса.</response>
[HttpGet("refs/{id}")]
[ProducesResponseType(200)]
[ProducesResponseType(500)]
public async Task<IActionResult> GetLiteratureReference(string id)
{
   try
   {
      string res = await _propertyRepository.GetLiteratureReference(id);
      return Ok(res);
   }
   catch (Exception ex)
   {
      _logger.LogError(ex.Message + "\n" + ex.StackTrace);
      return StatusCode(500, ex.Message + "\n" + ex.StackTrace);
   }
}



/// <summary>
/// Получение единиц измерения.
/// </summary>
/// <returns>Два словаря - пары ЕИ1-ЕИ2 и пары Выбранная_ЕИ-Unit.</returns>
/// <response code="200">Успешное выполнение.</response>
```

```
/// <response code="500">Ошибка на стороне сервиса.</response>
[HttpGet("units/")]
[ProducesResponseType(200)]
[ProducesResponseType(500)]
public async Task<IActionResult> GetUnits()
{
   try
   {
      var pair = await _propertyRepository.GetUnitsAndPairs();
      var res = JsonConvert.SerializeObject(pair, settings);
      return Ok(res);
   }
   catch (Exception ex)
   {
      _logger.LogError(ex.Message + "\n" + ex.StackTrace);
      return StatusCode(500, ex.Message + "\n" + ex.StackTrace);
   }
}



/// <summary>
/// Получение всех элементов с основыми свойствами.
/// </summary>
/// <returns>Список всех элементов таблицы Менделеева.</returns>
/// <response code="200">Успешное выполнение.</response>
/// <response code="500">Ошибка на стороне сервиса.</response>
[HttpGet("elems/")]
[ProducesResponseType(200)]
[ProducesResponseType(500)]
public async Task<IActionResult> GetAllElements()
```

```
{
    try
    {
        var elems = await _propertyRepository.GetAllElements();
        var res = JsonConvert.SerializeObject(elems, settings);
        return Ok(res);
    }
    catch (Exception ex)
    {
        _logger.LogError(ex.Message + "\n" + ex.StackTrace);
        return StatusCode(500, ex.Message + "\n" + ex.StackTrace);
    }
}



/// <summary>
/// Получение всех свойств с их Id, названием и типом.
/// </summary>
/// <returns>Список всех свойств в формате Id, Название, Тип.</returns>
/// <response code="200">Успешное выполнение.</response>
/// <response code="500">Ошибка на стороне сервиса.</response>
[HttpGet("propsnames/")]
[ProducesResponseType(200)]
[ProducesResponseType(500)]
public async Task<IActionResult> GetAllPropertiesNames()
{
    try
    {
        var props = await _propertyRepository.GetAllPropertiesNames();
        var res = JsonConvert.SerializeObject(props, settings);
```

```
            return Ok(res);
        }
        catch (Exception ex)
        {
            _logger.LogError(ex.Message + "\n" + ex.StackTrace);
            return StatusCode(500, ex.Message + "\n" + ex.StackTrace);
        }
    }
```

```
    /// <summary>
    /// Получение всех свойств выбранного элемента (Режим 1).
    /// </summary>
    /// <param name="id">Id элемента.</param>
    /// <returns>Список всех значений всех свойств выбранного элемента.</returns>
    /// <response code="200">Успешное выполнение.</response>
    /// <response code="500">Ошибка на стороне сервиса.</response>
    [HttpGet("allprops/{id}")]
    [ProducesResponseType(200)]
    [ProducesResponseType(500)]
    public async Task<IActionResult> GetAllPropsOfGivenElem(int id)
    {
        try
        {
            var props = await _propertyRepository.GetAllPropertiesOfElementById(id);
            var res = JsonConvert.SerializeObject(props, settings);
            return Ok(res);
        }
```

```
        catch (Exception ex)
        {
            _logger.LogError(ex.Message + "\n" + ex.StackTrace);
            return StatusCode(500, ex.Message + "\n" + ex.StackTrace);
        }
    }


    /// <summary>
    /// Получение значения выбранного свойства выбранного элемента (Подрежим 2).
    /// </summary>
    /// <remarks>
    /// Важно: Возвращается список, так как для одного Id свойства могут быть разные значения. Например:
    /// Melting temperature (разные значения из разных источников) Radii Ionic ... (разные значения для разных зарядов)
    /// </remarks>
    /// <param name="elemId">Id элемента.</param>
    /// <param name="propId">Id свойства.</param>
    /// <returns>Список всех значений выбранного свойства для выбранного элемента.</returns>
    /// <response code="200">Успешное выполнение.</response>
    /// <response code="500">Ошибка на стороне сервиса.</response>
    [HttpGet("givenprops/{elemId},{propId}")]
    [ProducesResponseType(200)]
    [ProducesResponseType(500)]
    public async Task<IActionResult> GetGivenPropOfGivenElem(int elemId, string propId)
    {
        try
        {
```

```
            var prop = await _propertyRepository.GetGivenPropertyOfElementByIds(elemId, propId);
            return Ok(JsonConvert.SerializeObject(prop, objectSettings));
        }
        catch (Exception ex)
        {
            _logger.LogError(ex.Message + "\n" + ex.StackTrace);
            return StatusCode(500, ex.Message + "\n" + ex.StackTrace);
        }
    }

    /// <summary>
    /// Получение значений выбранных из списка свойств выбранного элемента (Режим 2).
    /// </summary>
    /// <param name="elemId">Id элемента.</param>
    /// <param name="propsId">Список Id выбранных свойств</param>
    /// <returns>Список выбранных свойств выбранного элемента.</returns>
    /// <response code="200">Успешное выполнение.</response>
    /// <response code="400">Список Id свойств не указан.</response>
    /// <response code="500">Ошибка на стороне сервиса.</response>
    [HttpGet("givenprops/{elemId}")]
    [ProducesResponseType(200)]
    [ProducesResponseType(400)]
    [ProducesResponseType(500)]
    public async Task<IActionResult> GetGivenPropsOfGivenElem(int elemId, [FromQuery] List<string> propsId)
    {
        try
        {
            if (propsId == null)
                return BadRequest("Список Id свойств обязательно должен быть указан.");
            var props = await _propertyRepository.GetGivenPropertiesOfElementById(elemId, propsId);
```

```
            return Ok(JsonConvert.SerializeObject(props, settings));
        }
        catch (Exception ex)
        {
            _logger.LogError(ex.Message + "\n" + ex.StackTrace);
            return StatusCode(500, ex.Message + "\n" + ex.StackTrace);
        }
    }

    /// <summary>
    /// Получение значений выбранного свойства для всех элементов таблицы Менделеева (Подрежим 3).
    /// </summary>
    /// <param name="propId">Id свойства</param>
    /// <returns>Список всех значений свойства.</returns>
    /// <response code="200">Успешное выполнение.</response>
    /// <response code="500">Ошибка на стороне сервиса.</response>
    [HttpGet("propsvalues/{propId}")]
    [ProducesResponseType(200)]
    [ProducesResponseType(500)]
    public async Task<IActionResult> GetGivenPropValuesById(string propId)
    {
        return await GivenPropValuesByIdOrRecQuery(propId, false);
    }

    /// <summary>
    /// Получение значений выбранных из списка свойств для всех элементов таблицы Менделеева (Режим 3).
```

```
/// </summary>
/// <param name="propsId">Список Id выбранных свойств</param>
/// <returns>Словарь значений свойств.</returns>
/// <response code="200">Успешное выполнение.</response>
/// <response code="400">Список Id свойств не указан.</response>
/// <response code="500">Ошибка на стороне сервиса.</response>
[HttpGet("propsvalues/")]
[ProducesResponseType(200)]
[ProducesResponseType(400)]
[ProducesResponseType(500)]

public async Task<IActionResult> GetGivenPropsValues([FromQuery] List<string> propsId)
{
   return await GivenPropsValuesOrRecQuery(propsId, false);
}
```

```
/// <summary>
/// Получение рекомендуемых значений выбранного свойства для всех элементов таблицы Менделеева (Подрежим 4).
/// </summary>
/// <param name="propId">Id свойства.</param>
/// <returns>Список рекомендуемых значений свойства.</returns>
/// <response code="200">Успешное выполнение.</response>
/// <response code="500">Ошибка на стороне сервиса.</response>
[HttpGet("recprops/{propId}")]
[ProducesResponseType(200)]
[ProducesResponseType(500)]
public async Task<IActionResult> GetGivenPropOfGivenElemRec(string propId)
```

```
{
   return await GivenPropValuesByIdOrRecQuery(propId, true);
}

/// <summary>
/// Получение рекомендуемых значений выбранных из списка свойств для всех элементов таблицы Менделеева (Режим 4).
/// </summary>
/// <param name="propsId">Список Id выбранных свойств</param>
/// <returns>Словарь рекомендуемых значений.</returns>
/// <response code="200">Успешное выполнение.</response>
/// <response code="400">Список Id свойств не указан.</response>
/// <response code="500">Ошибка на стороне сервиса.</response>
[HttpGet("recprops/")]
[ProducesResponseType(200)]
[ProducesResponseType(400)]
[ProducesResponseType(500)]
public async Task<IActionResult> GetGivenPropsOfGivenElemRec([FromQuery] List<string> propsId)
{
   return await GivenPropsValuesOrRecQuery(propsId, true);
}

/// <summary>
/// Получение значений выбранного свойства из заданного диапозона для всех элементов таблицы Менделеева (Подрежим 5).
/// </summary>
/// <param name="propId">Id свойства.</param>
/// <param name="left">Левая граница диапозона (опционально).</param>
```

```
/// <param name="right">Правая граница диапозона (опционально).</param>
/// <returns>Список рекомендуемых значений свойства.</returns>
/// <response code="200">Успешное выполнение.</response>
/// <response code="500">Ошибка на стороне сервиса.</response>
[HttpGet("queryprops/{propId}/{left}/{right}")]
[ProducesResponseType(200)]
[ProducesResponseType(500)]
public async Task<IActionResult> GetGivenPropOfGivenElemQuery(string propId, decimal left, decimal right)
{
   return await GivenPropValuesByIdOrRecQuery(propId, true, left, right);
}

/// <summary>
/// Получение значений выбранных из списка свойств из заданного диапозона для всех элементов таблицы Менделеева (Режим 5).
/// </summary>
/// <param name="propsId">Список Id выбранных свойств</param>
/// <param name="lefts">Левая граница диапозона (опционально).</param>
/// <param name="rights">Правая граница диапозона (опционально).</param>
/// <returns>Словарь рекомендуемых значений с учетом диапозона.</returns>
/// <response code="200">Успешное выполнение.</response>
/// <response code="400">Параметры списков для запроса заданы неверно.</response>
/// <response code="500">Ошибка на стороне сервиса.</response>
[HttpGet("queryprops")]
[ProducesResponseType(200)]
[ProducesResponseType(400)]
[ProducesResponseType(500)]
public async Task<IActionResult> GetGivenPropsOfGivenElemQuery(
   [FromQuery] List<string> propsId, [FromQuery] List<decimal> lefts, [FromQuery] List<decimal> rights)
{
   try
```

```
        {
            if (propsId.Count != lefts.Count || lefts.Count != rights.Count)
                return BadRequest("Длины списков Id свойств и границ диапозонов не равны.");
            var dict = await _propertyRepository.GetGivenPropertiesValuesWithQuery(propsId, true, lefts, rights);
            return Ok(JsonConvert.SerializeObject(dict, settings));
        }
        catch (Exception ex)
        {
            _logger.LogError(ex.Message + "\n" + ex.StackTrace);
            return StatusCode(500, ex.Message + "\n" + ex.StackTrace);
        }

    }

    #endregion
  }
}
```

2.22. DapperContext.cs

```
using Microsoft.Data.SqlClient;
using System.Data;

namespace ElementsWebAPI.Entities
{
  /// <summary>
  /// Контекст базы данных для работы Dapper
  /// </summary>
  public class DapperContext
  {
    private readonly IConfiguration _configuration;
    private readonly string _connectionString;
```

```
    /// <summary>
    /// Конструктор для задания конфигурации сервиса
    /// </summary>
    /// <param name="configuration">Конфигурация сервиса</param>
    public DapperContext(IConfiguration configuration)
    {
        _configuration = configuration;
        _connectionString = _configuration.GetConnectionString("SqlConnection");
    }

    /// <summary>
    /// Конструктор для тестирования.
    /// </summary>
    /// <param name="connection">Строка подключения к БД</param>
    public DapperContext(string connection)
    {
        _connectionString = connection;
    }

    /// <summary>
    /// Создание подключения к БД через данные из ConnectionString
    /// </summary>
    /// <returns>Соединение с БД</returns>
    public IDbConnection CreateConnection() => new SqlConnection(_connectionString);

  }
}
```

2.23. PropertyRepository.cs

```
using Dapper;
using ElementsClassLibrary;
```

```
using ElementsWebAPI.Interfaces;
using System.Globalization;

namespace ElementsWebAPI.Entities
{
    /// <summary>
    /// Класс репозитория, работающего с БД
    /// </summary>
    public class PropertyRepository : IPropertyRepository
    {
        #region Глобальные переменные и конструктор

        /// <summary>
        /// Контекст для работы с базой данных.
        /// </summary>
        private readonly DapperContext _context;

        /// <summary>
        /// Список Id свойств ионного радиуса, имеющими дополнительные значения.
        /// </summary>
        private static readonly List<string> riPropIds = new List<string>() { "S11", "S12", "S13", "S14", "S15" };
        /// <summary>
        /// Список Id таблиц для организации запросов к БД.
        /// </summary>
        private static readonly List<string> propIds = new List<string>() { "Properties", "S11", "S12", "S13", "S14",
"S15" };

        /// <summary>
        /// Конструктор с определением контекста, списка таблиц и конвертируемых единиц измерения.
        /// </summary>
        /// <param name="context"></param>
```

```
public PropertyRepository(DapperContext context)
{
   _context = context;
}

#endregion

#region Get-методы для получения данных из БД

/// <summary>
/// Метод получения информации об источнике по заданной ссылке
/// </summary>
/// <param name="id">Номер ссылки</param>
/// <returns>Информация об источнике по ссылке</returns>
public async Task<string> GetLiteratureReference(string id)
{
   using (var connection = _context.CreateConnection())
   {

      string query = @$"SELECT * FROM ((SELECT [Article], [Source], [Vol], [Num], [Page1], [Page2] FROM
[phases].[dbo].[bData] WHERE N = {id}) dat
            LEFT JOIN (SELECT TOP 1 N, [Year1], [Year2] FROM [phases].[dbo].[bYears] WHERE N = {id}) yer
ON 1=1
            LEFT JOIN (SELECT TOP 1 N, [DOI] FROM [phases].[dbo].bPdf WHERE N = {id}) doi ON 1=1)";
      LiteratureReference litref = connection.Query<LiteratureReference>(query).FirstOrDefault();
      if (litref == null)
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.15-01 12 01-1-ЛУ				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```
            return "No data";

        query = $"SELECT DISTINCT [Author] FROM [phases].[dbo].[bAuthors] WHERE N = {id}";
        List<string> authors = connection.Query<string>(query).ToList();
        litref.Authors = authors;



        return litref.ToString();

    }
}

/// <summary>
/// Метод чтения из БД и установки пар конвертируемых ЕИ а также уравнения конвертации.
/// </summary>
/// /// <remarks>
/// Важно: Так как готовые таблицы были сочтены некорректными и неподходящими для работы, было принято решение
/// о создании двух вспомогательных таблиц rUnits и rUnitConv для правильной и удобной работой с ЕИ.
/// Одна хранит Id конвертируемой величины, первую ЕИ, вторую ЕИ, уравнение конвертации.
/// Вторая хранит Id свойства, его название, Id конвертируемой величины, в которой измеряется свойство.
/// </remarks>
public async Task<Tuple<Dictionary<string, Unit>, Dictionary<string, string>>> GetUnitsAndPairs()
{
    Dictionary<string, Unit> units = new Dictionary<string, Unit>();
    Dictionary<string, string> unitPairs = new Dictionary<string, string>();

    using (var connection = _context.CreateConnection())
    {
```

```
        string query = @"SELECT Unit1, Unit2, Equation FROM [phases].[dbo].[rUnits]
                    WHERE Equation IS NOT NULL ORDER BY Id";
        List<Unit> list = connection.Query<Unit>(query).ToList();
        foreach (var unit in list)
        {
           units.TryAdd(unit.Unit1, unit);
           unitPairs.TryAdd(unit.Unit1, unit.Unit2);
        }
        return new Tuple<Dictionary<string, Unit>, Dictionary<string, string>>(units, unitPairs);
     }
  }


  /// <summary>
  /// Все элементы таблицы Менделеева с основными свойствами.
  /// </summary>
  /// <returns>Список всех элементов.</returns>
  public async Task<IEnumerable<Element>> GetAllElements()
  {
     using (var connection = _context.CreateConnection())
     {
        var elems = new List<Element>();
        string query = @"SELECT el.Id, el.Elem as Symbol, aw.AtomicWeight, den.Density, melt.MeltingTemp,
boil.BoilingTemp FROM [phases].[dbo].[xElements] el
                    INNER JOIN (SELECT * FROM (SELECT a.Symbol, a.Value as AtomicWeight, (ROW_NUMBER()
OVER (PARTITION BY Symbol ORDER BY Symbol DESC)) row
                    FROM [phases].[dbo].[A6] a) un WHERE row = 1) aw ON aw.Symbol = el.Elem
                    INNER JOIN (SELECT * FROM (SELECT a.Symbol, a.Value as Density, (ROW_NUMBER() OVER
(PARTITION BY Symbol ORDER BY Symbol DESC)) row
                    FROM [phases].[dbo].[I5] a) un WHERE row = 1) den ON den.Symbol = el.Elem
                    INNER JOIN (SELECT * FROM (SELECT a.Symbol, a.Value as MeltingTemp, (ROW_NUMBER()
OVER (PARTITION BY Symbol ORDER BY Symbol DESC)) row
```

```
                FROM [phases].[dbo].[C1] a) un WHERE row = 1) melt ON melt.Symbol = el.Elem
                INNER JOIN (SELECT * FROM (SELECT a.Symbol, a.Value as BoilingTemp, (ROW_NUMBER() OVER (PARTITION BY Symbol ORDER BY Symbol DESC)) row
                FROM [phases].[dbo].[C2] a) un WHERE row = 1) boil ON boil.Symbol = el.Elem
                ORDER BY el.Id";
            elems.AddRange(await connection.QueryAsync<Element>(query));
            return elems;
        }
    }


    /// <summary>
    /// Все свойства из БД в формате Id, Название, Тип
    /// </summary>
    /// <returns>Список всех свойств, где Id, Name и Value - значения Id, Названия и Типа свойства.</returns>
    public async Task<IEnumerable<Property>> GetAllPropertiesNames()
    {
        using (var connection = _context.CreateConnection())
        {
            string query = @"SELECT PropertyNumber as Id, PropertyName as Name, PropertyType as Value, Unit
                FROM[phases].[dbo].[PropertiesNames]
                    INNER JOIN (SELECT NProp, Unit1 as Unit from [phases].[dbo].[rPropUnit]
                    INNER JOIN [phases].[dbo].[rUnits] ON UnitId = Id) as u ON PropertyNumber = u.NProp  ORDER BY Name";
            return await connection.QueryAsync<Property>(query);
        }
    }


    /// <summary>
    /// Все свойства выбранного из таблицы элемента.
```

```
    /// </summary>
    /// <param name="id">Id элемента.</param>
    /// <returns>Список всех значений всех свойств выбранного элемента.</returns>
    public async Task<IEnumerable<IProperty>> GetAllPropertiesOfElementById(int id)
    {
      using (var connection = _context.CreateConnection())
      {
        var allProps = new List<IProperty>();
        foreach (string propId in propIds)
        {
          string RIField = riPropIds.Contains(propId) ? ", Charge" : "";
          string RIShanonField = propId == "S15" ? ", CN" : "";
          string query = $@"SELECT pr.NProp as Id, Symbol as ElementSymbol, PropertyName as Name, Value,
Unit, Comments,
                Nref as Reference{RIField}{RIShanonField} FROM [phases].[dbo].[{propId}] pr
                INNER JOIN [phases].[dbo].[PropertiesNames] pn ON pr.Nprop = pn.PropertyNumber
                INNER JOIN (SELECT NProp, Unit1 as Unit from [phases].[dbo].[rPropUnit]
                                INNER JOIN [phases].[dbo].[rUnits] ON UnitId = Id) as u ON pr.Nprop =
u.NProp
                WHERE Symbol = (SELECT Elem FROM xElements el WHERE el.Id = @Id) ORDER BY Name";
          allProps.AddRange(riPropIds.Contains(propId) ?
            (await connection.QueryAsync<RIProperty>(query, new { id })).OrderBy(x => x.Id) :
            (await connection.QueryAsync<Property>(query, new { id })));
        }
        return allProps;
      }
    }
```

```
    /// <summary>
```

```
/// Конкретное свойство по Id для выбранного элемента по Id.
/// </summary> ///
/// <param name="elemId">Id элемента.</param>
/// <param name="propId">Id свойства.</param>
/// <returns>Список всех значений конкретного свойства для выбранного элемента.</returns>
public async Task<IEnumerable<IProperty>> GetGivenPropertyOfElementByIds(int elemId, string propId)
{
   string tableFrom = "Properties";
   string RIField = "";
   string RIShanonField = propId == "S15" ? ", CN" : "";
   if (riPropIds.Contains(propId))
   {
      tableFrom = propId;
      RIField = ", Charge";
   }

   string query = $@"SELECT pr.NProp as Id, Symbol as ElementSymbol, PropertyName as Name, Value, Unit,
Comments,
        Nref as Reference{RIField}{RIShanonField} FROM [phases].[dbo].[{tableFrom}] pr
        INNER JOIN [phases].[dbo].[PropertiesNames] pn ON pr.Nprop = pn.PropertyNumber
        INNER JOIN (SELECT NProp, Unit1 as Unit from [phases].[dbo].[rPropUnit]
                INNER JOIN [phases].[dbo].[rUnits] ON UnitId = Id) as u ON pr.Nprop = u.NProp
        WHERE pr.NProp = @propId AND Symbol = (SELECT Elem FROM xElements el WHERE el.Id =
@elemId)";

   using (var connection = _context.CreateConnection())
   {
      var props = riPropIds.Contains(propId) ?
          (await connection.QueryAsync<RIProperty>(query, new { propId, elemId })).OrderBy(x => x.Id) :
          (await connection.QueryAsync<Property>(query, new { propId, elemId }));
      return props;
```

```
        }
    }

    /// <summary>
    /// Выбранные свойства выбранного элемента.
    /// </summary>
    /// <param name="elemId">Id элемента.</param>
    /// <param name="propsId">Список Id выбранных свойств</param>
    /// <returns>Список выбранных свойств выбранного элемента.</returns>
    public async Task<IEnumerable<IProperty>> GetGivenPropertiesOfElementById(int elemId,
IEnumerable<string> propsId)
    {
        List<IProperty> props = new List<IProperty>();
        foreach (string propId in propsId)
        {
            var prop = await GetGivenPropertyOfElementByIds(elemId, propId);
            props.AddRange(prop);
        }
        return props;
    }



    /// <summary>
    /// Значения конкретного свойства для всех элементов с возможностью выбора рекомендованного
значения и диапозона значений.
    /// </summary>
    /// <param name="propId">Id свойства.</param>
    /// <param name="isRecomended">Флаг: все значения | рекомендованные значения.</param>
    /// <param name="left">Левая граница диапозона (опционально).</param>
    /// <param name="right">Правая граница диапозона (опционально).</param>
```

```
    /// <returns>Список всех значений свойства.</returns>
    public async Task<IEnumerable<IProperty>> GetGivenPropertyValuesById(
      string propId, bool isRecomended, decimal? left, decimal? right)
    {
      string tableFrom = "Properties";
      string RIField = "";
      string RIShanonField = propId == "S15" ? ", CN" : "";
      string queryRec = isRecomended ? " AND Rec = 1" : "";
      string queryLeft = left.HasValue ? $" AND Value >= {left.Value.ToString(CultureInfo.InvariantCulture)}" : "";
      string queryRight = right.HasValue ? $" AND Value <= {right.Value.ToString(CultureInfo.InvariantCulture)}" :
"";
      if (riPropIds.Contains(propId))
      {
        tableFrom = propId;
        RIField = ", Charge";
      }

      string query = $@"SELECT pr.NProp as Id, Symbol as ElementSymbol, PropertyName as Name, Value, Unit,
Comments,
          Nref as Reference, el.Id as ElId{RIField}{RIShanonField} FROM [phases].[dbo].[{tableFrom}] pr
          INNER JOIN [phases].[dbo].[PropertiesNames] pn ON pr.Nprop = pn.PropertyNumber
          INNER JOIN (SELECT NProp, Unit1 as Unit from [phases].[dbo].[rPropUnit]
                  INNER JOIN [phases].[dbo].[rUnits] ON UnitId = Id) as u ON pr.Nprop = u.NProp
          INNER JOIN [phases].[dbo].[xElements] el ON el.Elem = pr.Symbol
          WHERE (pr.NProp = @propId){queryRec}{queryLeft}{queryRight} order by ElId";

      using (var connection = _context.CreateConnection())
      {
        var props = riPropIds.Contains(propId) ?
          (await connection.QueryAsync<RIProperty>(query, new { propId })).OrderBy(x => x.Id) :
          (await connection.QueryAsync<Property>(query, new { propId }));
```

```
            return props.ToList();
        }
    }

    /// <summary>
    /// Значения выбранных свойств для всех элементов с возможностью выбора рекомендованного значения.
    /// </summary>
    /// <param name="propsId">Список Id выбранных свойств</param>
    /// <param name="isRecomended">Флаг: все значения | рекомендованные значения.</param>
    /// <returns>Словарь Id - List<Property> (все).</returns>
    public async Task<Dictionary<string, IEnumerable<IProperty>>> GetGivenPropertiesValues(
        IEnumerable<string> propsId, bool isRecomended)
    {
        Dictionary<string, IEnumerable<IProperty>> dict = new Dictionary<string, IEnumerable<IProperty>>();
        foreach (var prop in propsId)
        {
            var vals = await GetGivenPropertyValuesById(prop, isRecomended, null, null);
            dict.Add(prop, vals);
        }
        return dict;
    }

    /// <summary>
    /// Выбранные свойства со значениями из заданного диапозона значений с возможностью выбора рекомендованного значения.
    /// </summary>
    /// <remarks>
    /// Важно: Предполагается, что длины списков заданы корректно, то есть равны.
```

```
    /// </remarks>
    /// <param name="propsId">Список Id выбранных свойств</param>
    /// <param name="isRecomended">Флаг: все значения | рекомендованные значения.</param>
    /// <param name="lefts">Список левых границ диапозона для каждого свойства из списка (опционально).</param>
    /// <param name="rights">Список правых границ диапозона для каждого свойства из списка (опционально).</param>
    /// <returns>Словарь Id - List<Property> (все).</returns>
    public async Task<Dictionary<string, IEnumerable<IProperty>>> GetGivenPropertiesValuesWithQuery(
      IEnumerable<string> propsId, bool isRecomended, IEnumerable<decimal>? lefts, IEnumerable<decimal>? rights)
    {
      if (lefts != null && rights != null && (propsId.Count() != lefts.Count() || lefts.Count() != rights.Count()))
        throw new ArgumentException("Длины списков Id свойств и границ диапозонов не равны.");

      List<IProperty> res = new List<IProperty>();
      for (int i = 0; i < propsId.Count(); i++)
      {
        List<IProperty> vals = (await GetGivenPropertyValuesById(
          propsId.ElementAt(i), isRecomended, lefts?.ElementAt(i), rights?.ElementAt(i))).AsList();

        if (i == 0)
        {
          res = vals;
          continue;
        }

        List<IProperty> intersected = new List<IProperty>();
        List<string> addedElemSymbols = new List<string>();
        foreach (var val in vals)
        {
```

```
            if (addedElemSymbols.Contains(val.ElementSymbol))
            {
               intersected.Add(val);
               continue;
            }

            var a = res.FindAll(y => y.ElementSymbol == val.ElementSymbol);
            if (a.Count != 0)
            {
               intersected.AddRange(a);
               intersected.Add(val);
               addedElemSymbols.Add(val.ElementSymbol);
            }
         }
         res = intersected;
      }

      Dictionary<string, IEnumerable<IProperty>> dict = new Dictionary<string, IEnumerable<IProperty>>();
      foreach (var prop in res)
         if (dict.ContainsKey(prop.ElementSymbol))
            dict[prop.ElementSymbol].AsList().Add(prop);
         else dict.Add(prop.ElementSymbol, new List<IProperty>() { prop });
      return dict;
   }

   #endregion
  }
}
```

2.24. IPropertyRepository.cs

```
using ElementsClassLibrary;
```

```
using System.Collections.Generic;


namespace ElementsWebAPI.Interfaces
{
   public interface IPropertyRepository
   {
      public Task<string> GetLiteratureReference(string id);
      public Task<Tuple<Dictionary<string, Unit>, Dictionary<string, string>>> GetUnitsAndPairs();


      public Task<IEnumerable<Element>> GetAllElements();
      public Task<IEnumerable<Property>> GetAllPropertiesNames();


      public Task<IEnumerable<IProperty>> GetAllPropertiesOfElementById(int id);



      public Task<IEnumerable<IProperty>> GetGivenPropertyOfElementByIds(int elemId, string propId);
      public Task<IEnumerable<IProperty>> GetGivenPropertiesOfElementById(int elemId, IEnumerable<string>
propsId);



      public Task<IEnumerable<IProperty>> GetGivenPropertyValuesById(
         string propId, bool isRecomended, decimal? left, decimal? right);
      public Task<Dictionary<string, IEnumerable<IProperty>>> GetGivenPropertiesValues(
         IEnumerable<string> propsId, bool isRecomended);



      public Task<Dictionary<string, IEnumerable<IProperty>>> GetGivenPropertiesValuesWithQuery(
         IEnumerable<string> propsId, bool isRecomended, IEnumerable<decimal> lefts, IEnumerable<decimal>
rights);
   }
}
```

2.25. Index.cshtml

```
<div class="container marketing">
  <div class="row featurette">
    <div class="col-md-7 order-md-2">
      <h2 class="featurette-heading">A.A.BAIKOV INSTITUTE OF METALLURGY AND MATERIALS SCIENCE</h2>
      <p class="lead">Web service for database on chemical elements properties</p>
    </div>
    <div class="col-md-5 order-md-1">
      <img src="https://phase.imet-db.ru/images/imet2.gif" class="img-fluid" alt="Responsive image">
    </div>
  </div>
</div>
```

2.26. ElementNotFound.cshtml

```
<link rel="stylesheet" href="~/css/notfound.css"/>
<link href="https://fonts.googleapis.com/css2?family=Nunito+Sans:wght@600;900&display=swap"
rel="stylesheet">
<script src="https://kit.fontawesome.com/4b9ba14b0f.js" crossorigin="anonymous"></script>

<div class="mainbox">
  <div class="err">4</div>
  <i class="far fa-question-circle fa-spin"></i>
  <div class="err2">4</div>
  <div class="msg">Ooups, there is no data for this element
    <p>You can <a asp-action="Mendel" asp-route-mode="@ViewBag.Mode">change it</a> unitl database is
updated.</p>
  </div>

</div>
```

2.27. ElementPropsList.cshtml

```
@using System.Web
```

```
<div class="container">
    <div class="row justify-content-center">
        <h2>Properties of @ViewBag.Element.Trim():</h2>

        @if (ViewBag.Properties.Count != 0)
        {
            <table class="table table-striped">
                <thead class="thead-dark">
                    <tr>
                        <th>Property Name</th>
                        <th>Value</th>
                        <th>Unit</th>
                        <th>Comments</th>
                        <th>Reference</th>
                    </tr>
                </thead>
                <tbody>
                    @for (int i = 0; i < ViewBag.Properties.Count; i++)
                    {
                        <tr>
                            <td>@Html.Raw(ViewBag.Properties[i].Name)</td>
                            <td>@Html.Raw(ViewBag.Properties[i].GetFormattedValue())</td>
                            <td>@Html.Raw(ViewBag.Properties[i].Unit)</td>
                            <td>@Html.Raw(ViewBag.Properties[i].Comments)</td>
                            <td><a asp-action="LitRef" asp-route-id="@ViewBag.Properties[i].Reference"
target="_blank">@ViewBag.Properties[i].Reference</a></td>
                        </tr>
                    }
                </tbody>
            </table>
```

```
}
else
{
   <h4>No basic properties</h4>
}

<h2>Radii Ionic properties of @ViewBag.Element:</h2>

@if (ViewBag.RIProperties.Count != 0)
{
   <table class="table table-striped">
      <thead class="thead-dark">
         <tr>
            <th>Property Name</th>
            <th>Charge</th>
            <th>CN</th>
            <th>Value</th>
            <th>Unit</th>
            <th>Comments</th>
            <th>Reference</th>
         </tr>
      </thead>
      <tbody>
         @for (int i = 0; i < ViewBag.RIProperties.Count; i++)
         {
            <tr>
               <td>@Html.Raw(@ViewBag.RIProperties[i].Name)</td>
               <td>@Html.Raw(@ViewBag.RIProperties[i].Charge)</td>
               <td>@Html.Raw(@ViewBag.RIProperties[i].CN)</td>
               <td>@Html.Raw(@ViewBag.RIProperties[i].GetFormattedValue())</td>
```

```
                <td>@Html.Raw(@ViewBag.RIProperties[i].Unit)</td>
                <td>@Html.Raw(ViewBag.RIProperties[i].Comments)</td>
                <td><a asp-action="LitRef" asp-route-id="@ViewBag.RIProperties[i].Reference"
target="_blank">@ViewBag.RIProperties[i].Reference</a></td>
              </tr>
            }
          </tbody>
        </table>
      }
      else
      {
        <h4>No Radii Ionic properties</h4>
      }
    </div>
    <div class="row justify-content-center">
      <div class="d-grid gap-2 col-4 mx-auto mt-3">
        <a asp-action="Mendel" asp-route-mode="@ViewBag.Mode" class="btn btn-primary"
type="button">Change element</a>

        @if (ViewBag.Mode == 2)
        {
          <a asp-action="PropsList" asp-route-mode="2" asp-route-elemId="@ViewBag.ElemId"
          class="btn btn-primary" type="button">Change properties</a>
        }
      </div>
    </div>
</div>
```

2.28. LitRef.cshtml

```
<div class="container">
    <h2>Literature reference:</h2>
    <h4>@ViewBag.Lit</h4>
```

```
</div>
```

2.29. Mendel.cshtml

```
@using System.Globalization

@model List<List<Element>>


<link rel="stylesheet" href="~/css/mendel.css" asp-append-version="true" />


<div class="row">
   <div class="periodic">
      @for (int i = 0; i < Model.Count; i++)
      {
         <div class="periodic-row">
            @for (int j = 0; j < Model[i].Count; j++)
            {
               if (Model[i][j] == null)
               {
                  <div class="cell"></div>
                  continue;
               }
               string path = ViewBag.Mode == 1 ? $"/Properties/AllProps/{@Model[i][j].Id}" :
$"/Properties/PropsList/2/{Model[i][j].Id}";


               <a href="@path" class="cell">
                  <div class="element">
                     <div class="at_num">@Model[i][j].Id</div>
                     <div class="symbol">@Model[i][j].Symbol</div>
                     <div
class="at_details"><br/>@Model[i][j].AtomicWeight.ToString(CultureInfo.InvariantCulture)</div>
                  </div>
               </a>
            }
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.15-01 12 01-1-ЛУ				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```
        </div>
      }
    </div>
</div>
```

2.30. PropsList.cshtml

```
<div class="row form-group justify-content-center">
   <h2 class="text-center">Choose properties:</h2>
   <div class="d-flex gap-5 justify-content-center  mt-2">
      <form asp-action="PropsList" asp-route-mode="@ViewBag.Mode" asp-route-elemId="@ViewBag.ElemId"
asp-controller="Properties" method="post">
         @foreach (var item in Model.AvailableProps)
         {
            <div class="checkbox">
               <label>
                  <input type="checkbox"
                     name="SelectedProps"
                     value="@item.Value" /> @Html.Raw(item.Text)
               </label>
            </div>
         }

         <div class="d-grid gap-2 mx-auto mt-3">
            <input type="submit" class="btn btn-primary" value="Choose" />
         </div>
      </form>
   </div>
</div>
```

2.31. PropsValues.cshtml

```
<script type="text/javascript">
   window.onload = function()
```

```
    {
        $.ajax({
            url: '/Properties/Partial?id=' + '@Html.Raw(ViewBag.FirstId)' + '&mode=' + @ViewBag.Mode,
            dataType: 'html',
            success: function(data) {
                $('#loading-div-background').hide();
                $('#tableContainer').html(data);
            }
        });
    }

    function propChanged(value)
    {
        $('#loading-div-background').show();
        $('#tableContainer').hide();
        $('#list').prop('disabled',true);
        $.ajax({
            url: '/Properties/Partial?id=' + value + '&mode=' + @ViewBag.Mode,
            dataType: 'html',
            success: function(data) {
                $('#loading-div-background').hide();
                $('#tableContainer').show();
                $('#list').prop('disabled',false);
                $('#tableContainer').html(data);
            }
        });
    }
</script>
```

```
<div class="row">
  <div class="col-8">
    <select id="list" class="form-select" onmousedown="this.value='';" onchange="propChanged(this.value);">
      @foreach (var item in ViewBag.Dict)
      {
        <option value='@item.Key'>@item.Value</option>
      }
    </select>
  </div>

  <div class="col-4 mx-auto">
    <a asp-action="PropsList" asp-route-mode="3" class="btn btn-primary" type="button">Change properties</a>
  </div>
</div>
<div id="tableContainer"></div>
```

2.32. Query.cshtml

```
@model QueryListViewModel

<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery-validate/1.19.1/jquery.validate.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery-validation-unobtrusive/3.2.11/jquery.validate.unobtrusive.min.js"></script>

<div class="row form-group justify-content-center">
  <div class="row"></div>

  @if (Model.Queries != null && Model.Queries.Count != 0)
  {
      <h2 class="text-center">Enter ranges:</h2>
      <div class="col-6 justify-content-center mt-2">
        <div asp-validation-summary="All" class="text-danger"></div>
```

```
<form asp-action="Query" method="post">
@for (int i = 0; i < Model.Queries.Count; i++)
{
   @Html.Hidden($"Queries[{i}].PropId", Model.Queries[i].PropId)
   @Html.Hidden($"Queries[{i}].Property", Model.Queries[i].Property)
        <div class="card">
           <div class="card-header text-center">@Html.Raw(Model.Queries[i].Property +
(string.IsNullOrEmpty(ViewBag.Units[i]) ? "" : $" ({ViewBag.Units[i]})"))</div>
           <div class="card-body">
        @{
           string left = $"Queries[{i}].Left";
           string right = $"Queries[{i}].Right";
        }
            <div class="row">
              <div class="col-4">
                 <input class="form-control text-box single-line valid" data-val="true"
              data-val-regex-pattern="^-?\d+([,]\d{1,})?$"
              data-val-regex="Left border should be a numver separated by comma"
              id="@left" name="@left" type="text"
              aria-describedby="@left-error" aria-invalid="false" placeholder="0,0">
                 <span class="text-danger field-validation-valid" data-valmsg-for="@left" data-valmsg-
replace="true"></span>
              </div>
              <div class="col-4">
                 <blockquote class="blockquote text-center">
                    <p>? Value ?</p>
                 </blockquote>
              </div>
              <div class="col-4">
                 <input class="form-control text-box single-line valid" data-val="true"
              data-val-regex-pattern="^-?\d+([,]\d{1,})?$"
```

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.05.15-01 12 01-1-ЛУ				
Инв. № подл.	Подп. и дата	Взам. Инв. №	Инв. № дубл.	Подп. и дата

```
                    data-val-regex="Right border should be a numver separated by comma"
                    id="@right" name="@right" type="text"
                    aria-describedby="@right-error" aria-invalid="false" placeholder="0,0">
                        <span class="text-danger field-validation-valid" data-valmsg-for="@right" data-valmsg-replace="true"></span>
                    </div>
                </div>
            </div>
        </div>
    }
        <div class="d-grid gap-2 mx-auto mt-3">
            <input type="submit" class="btn btn-primary" value="Submit" />
        </div>
    </form>
</div>
}
else
{
    <h4 class="text-center">There is no selected properties for query</h4>
    <div class="d-grid gap-2 col-4 mx-auto mt-3">
        <a asp-controller="Properties" asp-action="PropsList" asp-route-mode="5" class="btn btn-primary">Select properties</a>
    </div>
}
</div>


@section Scripts {
    @{
    <partial name="_ValidationScriptsPartial" />
    }
}
```

2.33. QueryResult.cshtml

```
@model Dictionary<string, IEnumerable<IProperty>>


<div class="container">
  <div class="row justify-content-center">
    @if (Model == null || Model.Count == 0 || Model.Values.Any(x => x == null || x.Count() == 0))
    {
      <h4>No elements was found for this query.</h4>
    }
    else
    {
      foreach (var pair in Model)
      {
        List<IProperty> list = pair.Value.ToList();
        <div class="card mt-4">
          <div class="card-header text-center">
            <h3>@pair.Key</h3>
          </div>
          <div class="card-body">
            <table class="table table-striped">
              <thead class="thead-dark">
                <tr>
                  <th></th>
                  <th>Value</th>
                  <th>Unit</th>
                </tr>
              </thead>
              <tbody>
                @foreach (var prop in list)
                {
```

```
                    <tr>
                      @{
                        bool isRadiiIonic = prop is RIProperty;
                        if (prop is RIProperty)
                        {
                          RIProperty riprop = ((RIProperty)prop);
                          string charge = $" [Charge = {riprop.Charge}";
                          string cn = riprop.CN.HasValue ? $", CN = {riprop.CN}]" : "]";
                          <th>@Html.Raw(prop.Name)<span>@(charge + cn)</span></th>
                        }
                        else
                        {
                          <th>@Html.Raw(prop.Name)</th>
                        }
                        <th>@Html.Raw(prop.GetFormattedValue())</th>
                        <td>@Html.Raw(prop.Unit)</td>
                      }
                    </tr>
                  }
                </tbody>
              </table>
            </div>
          </div>
        }
      }
      <div class="d-grid gap-2 col-4 mx-auto mt-3">
        <a asp-controller="Properties" asp-action="PropsList" asp-route-mode="5" class="btn btn-
primary">Change properties</a>
      </div>
    </div>
  </div>
```

2.34. RecProps.cshtml

```
@model PropertyTableViewModel



<div class="row justify-content-center mt-3">
  <h3>Recomended values:</h3>


  @if (Model.Properties.Count != 0 && Model.Elements.Count != 0)
  {
    <div class="d-grid gap-2 mx-auto mt-3 justify-content-end">
      <a asp-controller="Properties" asp-action="Export" class="btn btn-primary" >Export as CSV</a>
    </div>
    <table class="table table-striped mt-2">
      <thead class="thead-dark">
        <tr>
          <th>Element</th>
          @for (int j = 0; j < Model.Properties.Count; j++)
          {
            <th>@Html.Raw(Model.Properties[j])</th>
          }
        </tr>
      </thead>
      <tbody>
        @for (int i = 0; i < Model.Elements.Count; i++)
        {
          <tr>
            <td>@Model.Elements[i]</td>
            @for (int j = 0; j < Model.Properties.Count; j++)
            {
              <td>@Model.Values[i][j].ToString("0.###############")</td>
```

```
                }
            </tr>
        }
    </tbody>
</table>
}
else
{
    <h4 class="text-center">There is no recomended properties/values for selected properties</h4>
    <div class="d-grid col-4 gap-2 mx-auto mt-3">
        <a asp-controller="Properties" asp-action="PropsList" asp-route-mode="4" class="btn btn-primary" >Select properties</a>
    </div>
}
</div>
```

2.35. Settings.cshtml

```
@model SettingsViewModel


<div class="row form-group justify-content-center">
    <h2 class="text-center">Choose settings:</h2>
    <div class="d-flex gap-5 justify-content-center mt-2">
        <form asp-action="Settings" method="post">
            @for (int i = 0; i < Model.Units.Count; i++)
            {
                <div class="container p-3 my-3 border">
                    @{
                        string first = @Model.Units[i].Item1.IsSelected ? "checked=\"checked\"" : "";
                        string second = @Model.Units[i].Item2.IsSelected ? "checked=\"checked\"" : "";
                        string name = "SelectedUnits[" + i + "]";
                    }
```

```
            <div class="form-check">
              <input class="form-check-input" name="@name" type="radio" value="@Model.Units[i].Item1.Id"
@first>
              <label class="form-check-label" for="@name">@Html.Raw(Model.Units[i].Item1.Id)</label>
            </div>
            <div class="form-check">
              <input class="form-check-input" name="@name" type="radio" value="@Model.Units[i].Item2.Id"
@second>
              <label class="form-check-label" for="@name">@Html.Raw(Model.Units[i].Item2.Id)</label>
            </div>
          </div>
        }

        <div class="d-grid gap-2 mx-auto mt-3">
          <input type="submit" class="btn btn-primary" value="Choose" />
        </div>
      </form>
    </div>
</div>
```

2.36. SettingsChanged.cshtml

```
<div class="row form-group justify-content-center">
  <h2 class="text-center">Settings successfully changed.</h2>
  <div class="d-flex gap-5 justify-content-center mt-2">
    <div class="d-grid gap-2 col-4 mx-auto mt-3">
        <a asp-action="Index" class="btn btn-primary" type="button">Home</a>
    </div>
  </div>
</div>
```

2.37. PartialValsTable.cshtml

```
@using System.Web
```

```
@model List<IProperty>


<div class="row justify-content-center mt-3">
   @{
      string header = ViewBag.Mode == 3 ? "Values" : "Recomended values";
      string unit = string.IsNullOrEmpty(Model[0].Unit) ? "" : $"({Model[0].Unit.Trim('(', ')', '[', ']')})";
   }
   <h3>@Html.Raw($"{header} of {Model[0].Name} {unit}:")</h3>



   @if (Model.Count != 0)
   {
      bool isRadiiIonic = Model[0] is RIProperty;
      <table class="table table-striped" >
         <thead class="thead-dark">
            <tr>
               <th>Element</th>
               <th>Value</th>
               @if (isRadiiIonic)
               {
                  <th>Charge</th>
                  <th>CN</th>
               }
               <th>Comments</th>
               <th>Reference</th>
            </tr>
         </thead>
         <tbody>
            @for (int i = 0; i < Model.Count; i++)
```

```
        {
          <tr>
            <td>@Model[i].ElementSymbol</td>
            <td>@Html.Raw(Model[i].GetFormattedValue())</td>
            @if (isRadiiIonic)
            {
              <td>@(((RIProperty)Model[i]).Charge)</td>
              <td>@(((RIProperty)Model[i]).CN)</td>
            }
            <td>@Html.Raw(Model[i].Comments)</td>
            <td><a asp-action="LitRef" asp-route-id="@Model[i].Reference"
target="_blank">@Model[i].Reference</a></td>
          </tr>
        }
      </tbody>
    </table>
  }
  else
  {
    <h4>No values for this property</h4>
  }
</div>
```

2.38. Error.cshtml

```
@model ErrorViewModel
@{
  ViewData["Title"] = "Error";
}

<h1 class="text-danger">Error.</h1>
<h2 class="text-danger">An error occurred while processing your request.</h2>
```

```
@if (Model.ShowRequestId)
{
    <p>
        <strong>Request ID:</strong> <code>@Model.RequestId</code>
    </p>
}

<h3>Development Mode</h3>
<p>
    Swapping to <strong>Development</strong> environment will display more detailed information about the error that occurred.
</p>
<p>
    <strong>The Development environment shouldn't be enabled for deployed applications.</strong>
    It can result in displaying sensitive information from exceptions to end users.
    For local debugging, enable the <strong>Development</strong> environment by setting the <strong>ASPNETCORE_ENVIRONMENT</strong> environment variable to <strong>Development</strong>
    and restarting the app.
</p>
```

2.39. _Layout.cshtml

```
<script src="https://ajax.aspnetcdn.com/ajax/jquery/jquery-2.2.0.min.js Jump "></script>
<script>
    window.onload = function () {
        $('#loading-div-background').show();
        $('#header').hide();
        $.ajax({
        url: '/Properties/SetParams',
        dataType: 'html',
        success: function(data) {
            $('#loading-div-background').hide();
```

```
        $('#header').show();
      }})
  };
</script>



<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>@ViewData["Title"] - ElementsMVCApp</title>
  <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
  <link rel="stylesheet" href="~/css/site.css" asp-append-version="true" />
  <link rel="stylesheet" href="~/ElementsMVCApp.styles.css" asp-append-version="true" />
</head>
<body>
  <header>
    <nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-light bg-white border-bottom box-shadow mb-3">
      <div id="header" class="container-fluid">
        <a class="navbar-brand" asp-area="" asp-controller="Home" asp-action="Index">Elements Web Service</a>
        <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target=".navbar-collapse" aria-controls="navbarSupportedContent"
                aria-expanded="false" aria-label="Toggle navigation">
          <span class="navbar-toggler-icon"></span>
        </button>
        <div class="navbar-collapse collapse d-sm-inline-flex justify-content-between">
          <ul class="navbar-nav flex-grow-1">
            <li class="nav-item">
```

```
                <a class="nav-link text-dark" asp-controller="Properties" asp-action="Mendel" asp-route-mode="1">
                    All properties of given element
                </a>
            </li>
            <li class="nav-item">
                <a class="nav-link text-dark" asp-area="" asp-controller="Properties" asp-action="Mendel" asp-route-mode="2">
                    Chosen properties of given element
                </a>
            </li>
            <li class="nav-item">
                <a class="nav-link text-dark" asp-area="" asp-controller="Properties" asp-action="PropsList" asp-route-mode="3">
                    Values of given property
                </a>
            </li>
            <li class="nav-item">
                <a class="nav-link text-dark" asp-area="" asp-controller="Properties" asp-action="PropsList" asp-route-mode="4">
                    Recomended values of given property
                </a>
            </li>
            <li class="nav-item">
                <a class="nav-link text-dark" asp-area="" asp-controller="Properties" asp-action="PropsList" asp-route-mode="5">
                    Query properties
                </a>
            </li>
        </ul>
        <form class="d-flex my-2">
            <a class="btn btn-outline-success" asp-controller="Properties" asp-action="Settings">Settings</a>
```

```
            </form>
          </div>
        </div>
      </nav>
    </header>

    <div class="container">
      <main role="main" class="pb-3">
        @RenderBody()
      </main>

      <div id="loading-div-background">
        <div id="loading-div" class="ui-corner-all">
          <div id="loadingbar"><i class="fa fa-spinner fa-spin fa-3x fa-fw"></i></div>
          <h2 class="text-center">Please wait....</h2>
        </div>
      </div>
    </div>

    @*<footer class="border-top footer text-muted">
      <div class="container">
      &copy; 2022 - ElementsMVCApp - <a asp-area="" asp-controller="Home" asp-action="Privacy">Privacy</a>
      </div>
      </footer>*@
    <script src="~/lib/jquery/dist/jquery.min.js"></script>
    <script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
    <script src="~/js/site.js" asp-append-version="true"></script>
    @await RenderSectionAsync("Scripts", required: false)
  </body>
  </html>
```

2.40. _Layout.cshtml.css

```
a.navbar-brand {
  white-space: normal;
  text-align: center;
  word-break: break-all;
}

a {
  color: #0077cc;
}

.btn-primary {
  color: #fff;
  background-color: #1b6ec2;
  border-color: #1861ac;
}

.nav-pills .nav-link.active, .nav-pills .show > .nav-link {
  color: #fff;
  background-color: #1b6ec2;
  border-color: #1861ac;
}

.border-top {
  border-top: 1px solid #e5e5e5;
}
.border-bottom {
  border-bottom: 1px solid #e5e5e5;
}
```

```
.box-shadow {
  box-shadow: 0 .25rem .75rem rgba(0, 0, 0, .05);
}

button.accept-policy {
  font-size: 1rem;
  line-height: inherit;
}

.footer {
  position: absolute;
  bottom: 0;
  width: 100%;
  white-space: nowrap;
  line-height: 60px;
}
```

2.41. _ValidationScriptsPartial.cshtml

```
<script src="~/lib/jquery-validation/dist/jquery.validate.min.js"></script>
<script src="~/lib/jquery-validation-unobtrusive/jquery.validate.unobtrusive.min.js"></script>
```

2.42. _ViewImports.cshtml

```
@using ElementsMVCApp
@using ElementsMVCApp.Models
@using ElementsClassLibrary
@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
```

2.43. _ViewStart.cshtml

```
@{
   Layout = "_Layout";
}
```

2.44. Home.css

```
element.style {
}

.row {
    --bs-gutter-x: 1.5rem;
    --bs-gutter-y: 0;
    display: flex;
    flex-wrap: wrap;
    margin-top: calc(-1 * var(--bs-gutter-y));
    margin-right: calc(-.5 * var(--bs-gutter-x));
    margin-left: calc(-.5 * var(--bs-gutter-x));
}

*, ::after, ::before {
    box-sizing: border-box;
}

div {
    display: block;
}

body {
    padding-top: 3rem;
    padding-bottom: 3rem;
    color: #5a5a5a;
}

body {
    margin: 0;
```

```
    font-family: var(--bs-body-font-family);
    font-size: var(--bs-body-font-size);
    font-weight: var(--bs-body-font-weight);
    line-height: var(--bs-body-line-height);
    color: var(--bs-body-color);
    text-align: var(--bs-body-text-align);
    background-color: var(--bs-body-bg);
    -webkit-text-size-adjust: 100%;
    -webkit-tap-highlight-color: transparent;
}

:root {
    --bs-blue: #0d6efd;
    --bs-indigo: #6610f2;
    --bs-purple: #6f42c1;
    --bs-pink: #d63384;
    --bs-red: #dc3545;
    --bs-orange: #fd7e14;
    --bs-yellow: #ffc107;
    --bs-green: #198754;
    --bs-teal: #20c997;
    --bs-cyan: #0dcaf0;
    --bs-white: #fff;
    --bs-gray: #6c757d;
    --bs-gray-dark: #343a40;
    --bs-gray-100: #f8f9fa;
    --bs-gray-200: #e9ecef;
    --bs-gray-300: #dee2e6;
    --bs-gray-400: #ced4da;
    --bs-gray-500: #adb5bd;
```

--bs-gray-600: #6c757d;

--bs-gray-700: #495057;

--bs-gray-800: #343a40;

--bs-gray-900: #212529;

--bs-primary: #0d6efd;

--bs-secondary: #6c757d;

--bs-success: #198754;

--bs-info: #0dcaf0;

--bs-warning: #ffc107;

--bs-danger: #dc3545;

--bs-light: #f8f9fa;

--bs-dark: #212529;

--bs-primary-rgb: 13,110,253;

--bs-secondary-rgb: 108,117,125;

--bs-success-rgb: 25,135,84;

--bs-info-rgb: 13,202,240;

--bs-warning-rgb: 255,193,7;

--bs-danger-rgb: 220,53,69;

--bs-light-rgb: 248,249,250;

--bs-dark-rgb: 33,37,41;

--bs-white-rgb: 255,255,255;

--bs-black-rgb: 0,0,0;

--bs-body-color-rgb: 33,37,41;

--bs-body-bg-rgb: 255,255,255;

--bs-font-sans-serif: system-ui,-apple-system,"Segoe UI",Roboto,"Helvetica Neue",Arial,"Noto Sans","Liberation Sans",sans-serif,"Apple Color Emoji","Segoe UI Emoji","Segoe UI Symbol","Noto Color Emoji";

--bs-font-monospace: SFMono-Regular,Menlo,Monaco,Consolas,"Liberation Mono","Courier New",monospace;

--bs-gradient: linear-gradient(180deg, rgba(255, 255, 255, 0.15), rgba(255, 255, 255, 0));

--bs-body-font-family: var(--bs-font-sans-serif);

--bs-body-font-size: 1rem;

--bs-body-font-weight: 400;

```
  --bs-body-line-height: 1.5;
  --bs-body-color: #212529;
  --bs-body-bg: #fff;
}

*, ::after, ::before {
  box-sizing: border-box;
}

*, ::after, ::before {
  box-sizing: border-box;
}
```

2.45. Mendel.css

```
body {
  font-family: Arial;
  -webkit-transform: translateZ(0);
  -ms-transform: translateZ(0);
  transform: translateZ(0);
}

.periodic {
  position: relative;
  height: 200px;
  margin-right: -1px;
  text-shadow: none;
}

.periodic-row {
  clear: both;
  height: 10%;
```

```
}

.cell {
    float: left;
    position: relative;
    width: 5.55%;
    height: 100%;
}

.element {
    position: absolute;
    top: 0;
    left: 0;
    bottom: 1px;
    right: 1px;
    box-sizing: border-box;
    box-shadow: 0px 0px 4px rgba(255, 255, 255, 0.5);
    border: 1px solid rgba(0, 0, 0, 0.05);
    text-align: center;
    cursor: default;
    pointer-events: none;
    -webkit-transition: all 200ms ease;
    transition: all 200ms ease;
    background-color: rgba(0, 128, 128, 0.6);
}

.cell:hover .element {
    border-color: rgba(0, 0, 0, 0.1);
    -webkit-transform: scale(3, 3);
    -ms-transform: scale(3, 3);
```

```
    transform: scale(3, 3);
    z-index: 1;
    background-color: rgba(0, 128, 128, 0.9);
}

.at_num,
.at_details {
    position: absolute;
    font-size: 4px;
    color: rgba(255, 255, 255, 0.5);
    opacity: 0;
}

.at_num {
    top: 4px;
    right: 5px;
}

.symbol {
    position: absolute;
    top: 50%;
    left: 0px;
    right: 0px;
    margin-top: -4px;
    font-size: 9px;
    line-height: 1;
    height: 9px;
    color: rgba(255, 255, 255, 0.9);
}
```

```
.at_details {
   bottom: 4px;
   left: 0px;
   right: 0px;
}

.cell:nth-child(-n+2) .element,
.cell:nth-child(n+13) .element {
   background-color: rgba(0, 160, 96, 0.6);
}

.cell:nth-child(1) .element,
.periodic-row:nth-child(2) .cell:nth-child(n+14) .element,
.periodic-row:nth-child(3) .cell:nth-child(n+15) .element,
.periodic-row:nth-child(4) .cell:nth-child(n+16) .element,
.periodic-row:nth-child(5) .cell:nth-child(n+17) .element {
   background-color: rgba(0, 192, 64, 0.6);
}

.periodic-row:nth-child(-n+6) .cell:nth-child(18) .element {
   background-color: rgba(64, 192, 0, 0.6);
}

.periodic-row:nth-child(n+9) .element {
   background-color: rgba(0, 96, 160, 0.6);
}

.cell:nth-child(-n+2):hover .element,
.cell:nth-child(n+13):hover .element {
   background-color: rgba(0, 160, 96, 0.9);
```

```
}

.cell:nth-child(1):hover .element,
.periodic-row:nth-child(2) .cell:nth-child(n+14):hover .element,
.periodic-row:nth-child(3) .cell:nth-child(n+15):hover .element,
.periodic-row:nth-child(4) .cell:nth-child(n+16):hover .element,
.periodic-row:nth-child(5) .cell:nth-child(n+17):hover .element {
   background-color: rgba(0, 192, 64, 0.9);
}

.periodic-row:nth-child(-n+6) .cell:nth-child(18):hover .element {
   background-color: rgba(64, 192, 0, 0.9);
}

.periodic-row:nth-child(n+9) .cell:hover .element {
   background-color: rgba(0, 96, 160, 0.9);
}

.cell:nth-child(1) .element {
   -webkit-transform-origin: 0 50%;
   -ms-transform-origin: 0 50%;
   transform-origin: 0 50%;
}

.cell:nth-child(18) .element {
   -webkit-transform-origin: 100% 50%;
   -ms-transform-origin: 100% 50%;
   transform-origin: 100% 50%;
}
```

```
.periodic-row:nth-child(10) .cell .element {
   -webkit-transform-origin: 50% 100%;
   -ms-transform-origin: 50% 100%;
   transform-origin: 50% 100%;
}

.periodic-row:nth-child(1) .cell:nth-child(1) .element {
   -webkit-transform-origin: 0 0;
   -ms-transform-origin: 0 0;
   transform-origin: 0 0;
}

.periodic-row:nth-child(1) .cell:nth-child(18) .element {
   -webkit-transform-origin: 100% 0;
   -ms-transform-origin: 100% 0;
   transform-origin: 100% 0;
}

.periodic-row:nth-child(10) .cell:nth-child(18) .element {
   -webkit-transform-origin: 100% 100%;
   -ms-transform-origin: 100% 100%;
   transform-origin: 100% 100%;
}

@media (min-width: 600px) {
   .periodic {
      height: 460px;
      margin-right: -2px;
   }
```

```
    .element {
      right: 2px;
      bottom: 2px;
    }

    .at_num,
    .at_details {
      font-size: 4px;
      opacity: 1;
    }

    .symbol {
      margin-top: -17px;
      font-size: 16px;
      font-weight: bold;
      line-height: 30px;
      height: 30px;
      color: rgba(255, 255, 255, 0.75);
      text-shadow: 0 0 4px rgba(255, 255, 255, 0.5);
    }
  }

  @media (min-width: 800px) {
    .periodic {
      height: 540px;
    }

    .symbol {
      font-size: 20px;
    }
```

```
}

@media (min-width: 992px) {
  .periodic {
    height: 680px;
  }

  .at_num,
  .at_details {
    font-size: 5px;
  }

  .symbol {
    font-size: 24px;
  }
}

@media (min-width: 1200px) {
  .periodic {
    height: 800px;
  }

  .at_num,
  .at_details {
    font-size: 6px;
  }

  .symbol {
    font-size: 30px;
  }
```

```
}
```

2.46. Notfound.css

```
body {
    background-color: #95c2de;
}

.mainbox {
    background-color: #95c2de;
    margin: auto;
    height: 600px;
    width: 600px;
    position: relative;
}

.err {
    color: #ffffff;
    font-family: 'Nunito Sans', sans-serif;
    font-size: 11rem;
    position: absolute;
    left: 20%;
    top: 8%;
}

.far {
    position: absolute;
    font-size: 8.5rem;
    left: 42%;
    top: 15%;
    color: #ffffff;
}
```

```
.err2 {
    color: #ffffff;
    font-family: 'Nunito Sans', sans-serif;
    font-size: 11rem;
    position: absolute;
    left: 68%;
    top: 8%;
}

.msg {
    text-align: center;
    font-family: 'Nunito Sans', sans-serif;
    font-size: 1.6rem;
    position: absolute;
    left: 16%;
    top: 45%;
    width: 75%;
}

a {
    text-decoration: none;
    color: white;
}

    a:hover {
        text-decoration: underline;
    }
```

2.47. Site.css

```
html {
```

```
  font-size: 14px;
}

@media (min-width: 768px) {
  html {
    font-size: 16px;
  }
}

html {
  position: relative;
  min-height: 100%;
}

body {
  margin-bottom: 60px;
}
```

Лист регистрации изменений

Изм.	Номера листов (страниц)				Всего листов (страниц) в документе	№ документа	Входящий № сопроводительного документа и дата	Подпись	Дата
	измененных	замененных	Новых	аннулированных					