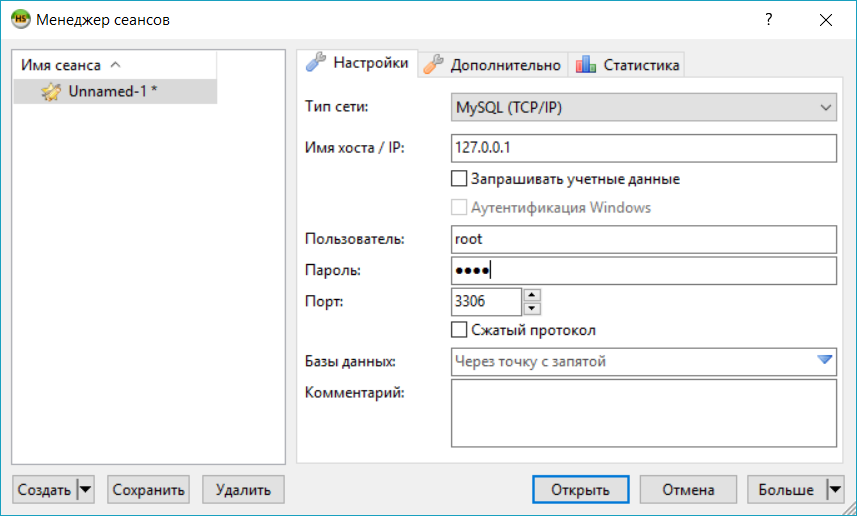
Assignment.   
Advanced Web Applications and Web Services

Как обновить данные:

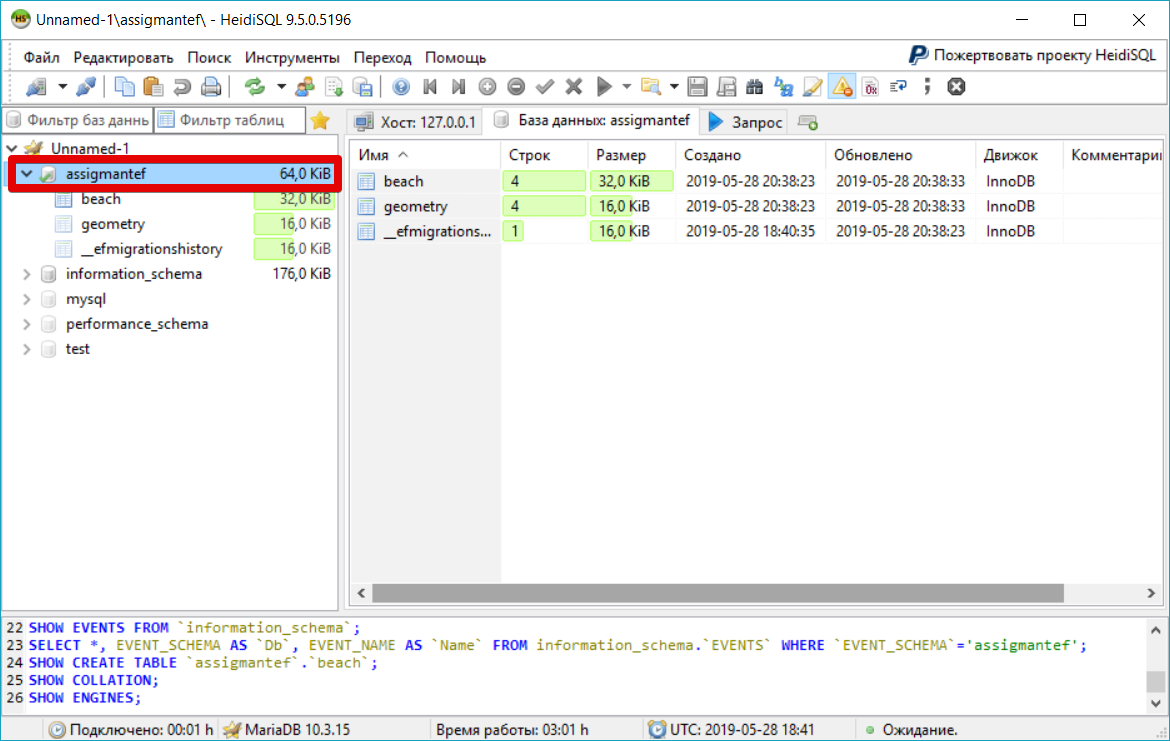
Запускаем HeidiSQL



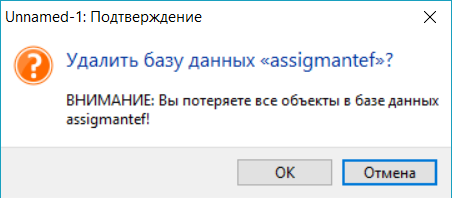
Подключаемся к серверу: вводим данные как на картинке и нажимаем «Открыть» (пароль: root)



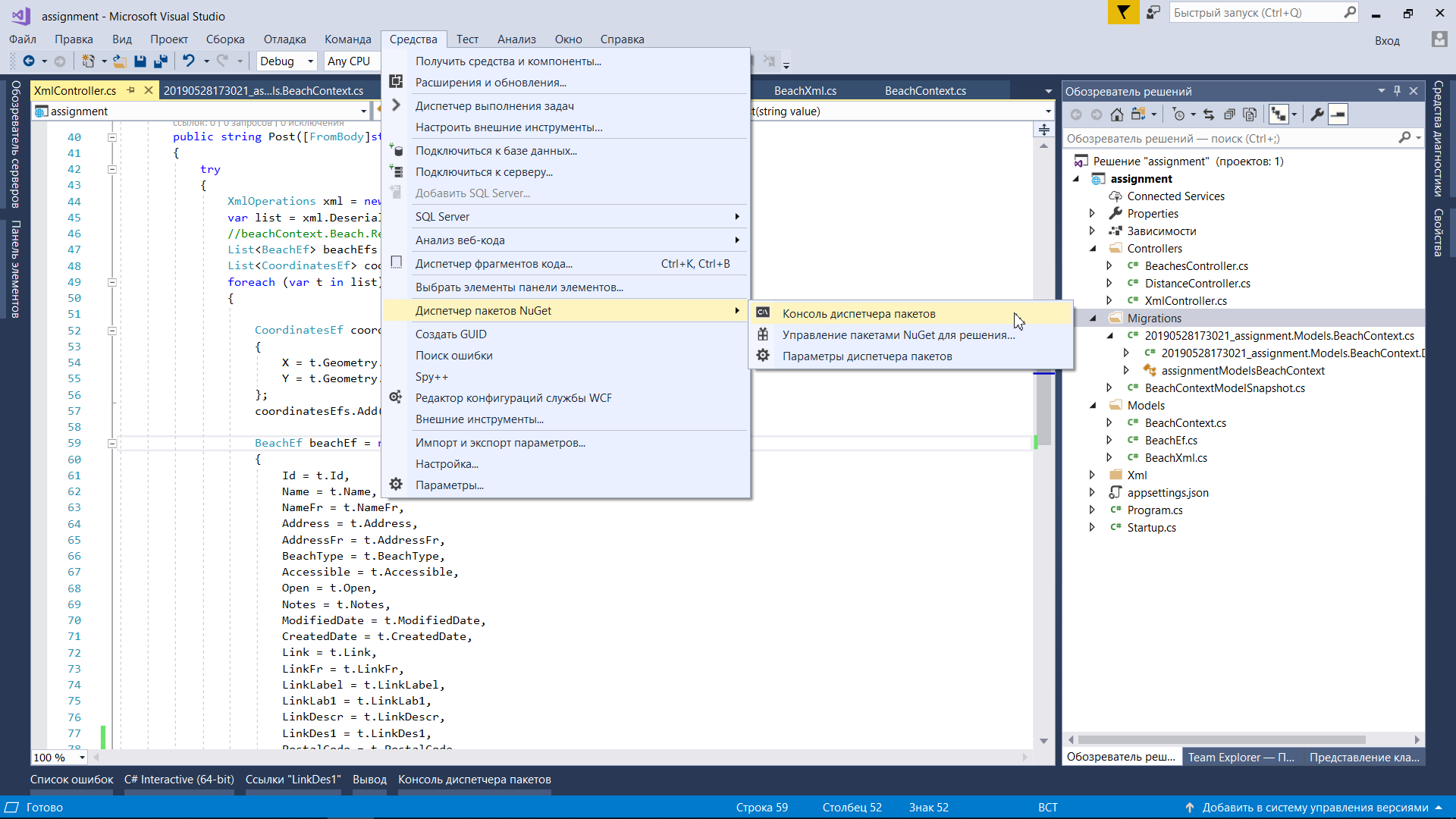
Правой кнопкой мыши нажимаем на assigmentef и выбираем «Уничтожить»



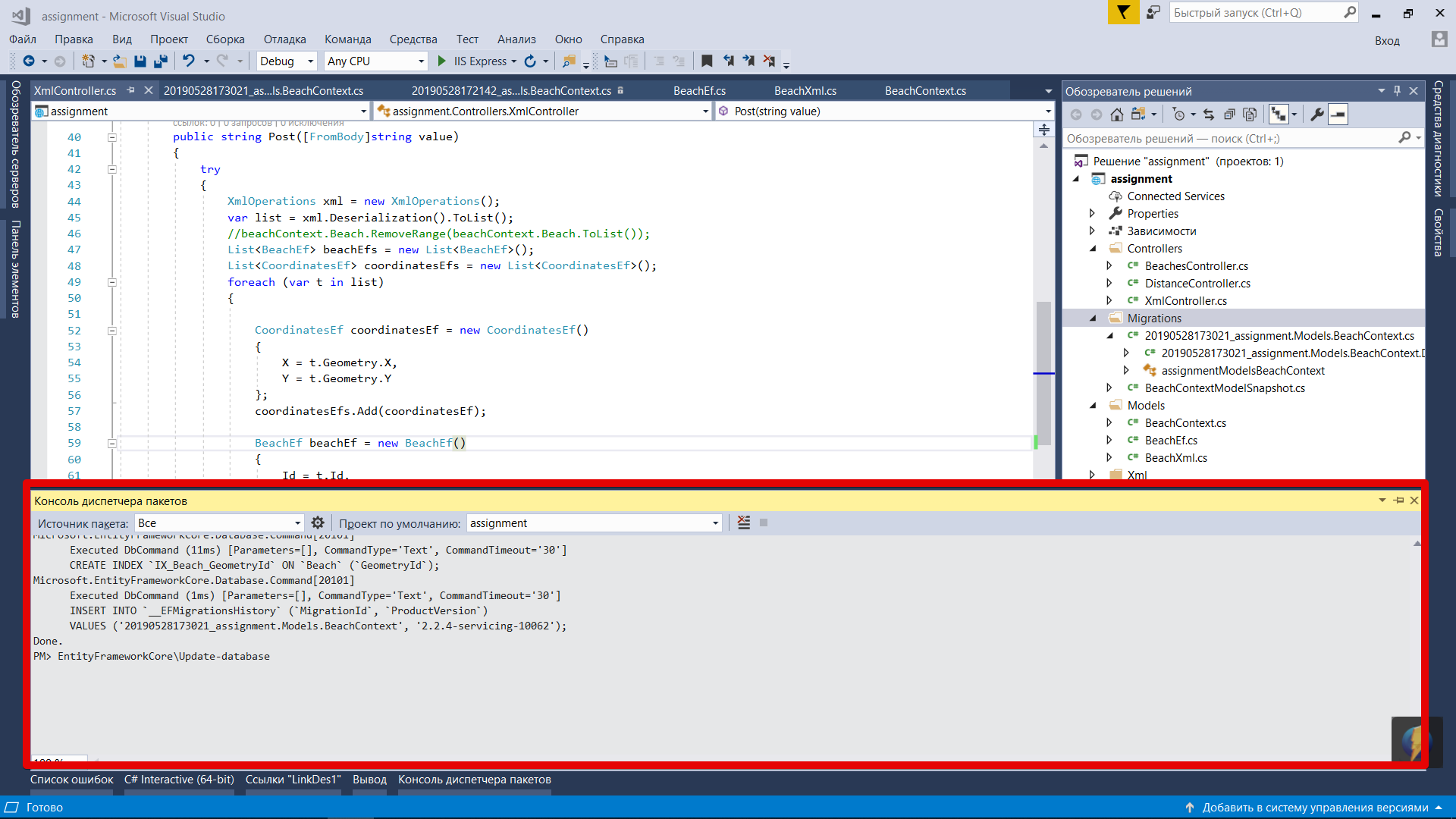
Соглашаемся



Запускаем проект в студии. Выбираем пункты в меню как на картинке

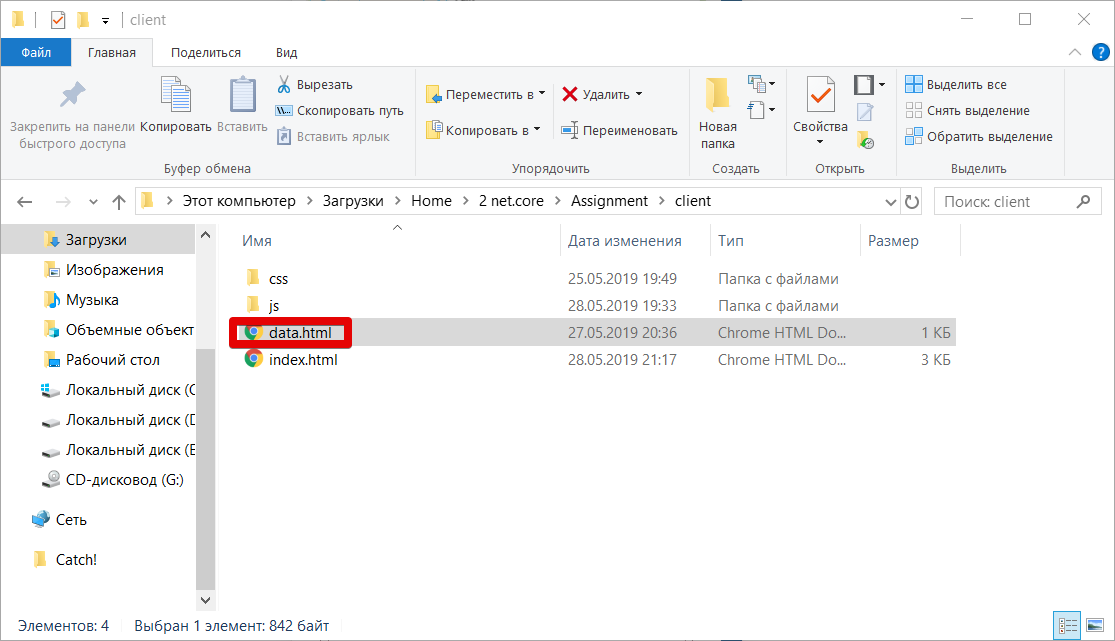


В открывшейся консоли вводим EntityFrameworkCore\Update-database

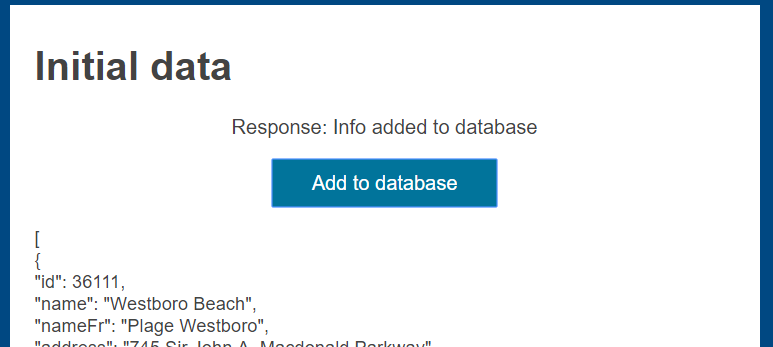


Запускаем сервер



Запускаем data.html

Нажимаем Add to database. Должен быть ответ “Info added to database”



Готово! Теперь можно запускать index.html.

1. **Server**
   1. Работа с БД через EntityFrameworkCore технологией CodeFirst.
      1. Создание моделей сущностей

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

using System.Linq;

using System.Threading.Tasks;

using System.Xml.Serialization;

namespace assignment.Models

{

public class BeachEf

{

[Key]

public long Id { get; set; }

public string Name { get; set; }

public string NameFr { get; set; }

public string Address { get; set; }

public string AddressFr { get; set; }

public string BeachType { get; set; }

public string Accessible { get; set; }

public string Open { get; set; }

public string Notes { get; set; }

public string ModifiedDate { get; set; }

public string CreatedDate { get; set; }

public string Link { get; set; }

public string LinkFr { get; set; }

public string LinkLabel { get; set; }

public string LinkLab1 { get; set; }

public string LinkDescr { get; set; }

public string LinkDes1 { get; set; }

public string PostalCode { get; set; }

public string Photo { get; set; }

public long GeometryId { get; set; }

public virtual CoordinatesEf Geometry { get; set; }

}

public class CoordinatesEf

{

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2214:DoNotCallOverridableMethodsInConstructors")]

public CoordinatesEf()

{

//Beach = new HashSet<BeachEf>();

}

[Key]

[DatabaseGenerated(DatabaseGeneratedOption.Identity)]

public long Id { get; set; }

public double X { get; set; }

public double Y { get; set; }

//[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

//public virtual ICollection<BeachEf> Beach { get; set; }

}

public class BeachEfInfo

{

public BeachEf beach { get; set; }

public CoordinatesEf geometry { get; set; }

}

}

* + 1. Создание контекста

using Microsoft.EntityFrameworkCore;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace assignment.Models

{

public class BeachContext : DbContext

{

public BeachContext(DbContextOptions options)

: base(options)

{

}

public virtual DbSet<BeachEf> Beach { get; set; }

public virtual DbSet<CoordinatesEf> Geometry { get; set; }

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

base.OnModelCreating(modelBuilder);

//modelBuilder.Entity<CoordinatesEf>()

// .HasMany(e => e.Beach)

// .WithOne(e => e.Geometry)

// .HasForeignKey(e => e.GeometryId);

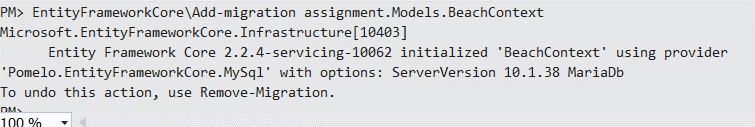
}

}

}

* + 1. Создание миграции через консоль диспетчера пакетов:

EntityFrameworkCore\Add-migration assignment.Models.BeachContext



* + 1. Развертывание БД через консоль диспетчера пакетов

EntityFrameworkCore\Update-database

* 1. Чтение координат пляжей из XML-файла (beaches.xml).

Файл (исходная структура DataSet XML изменена для удобного парсинга):

<?xml version="1.0" encoding="UTF-8" ?>

<root>

<features>

<properties>

<FACILITYID>36111</FACILITYID>

<NAME>Westboro Beach</NAME>

<NAME\_FR>Plage Westboro</NAME\_FR>

<ADDRESS>745 Sir John A. Macdonald Parkway</ADDRESS>

<POSTAL\_CODE>ON K2B 5L7</POSTAL\_CODE>

<BEACH\_TYPE>Plage</BEACH\_TYPE>

<ACCESSIBLE/>

<OPEN/>

<NOTES/>

<MODIFIED\_D>2014/09/29</MODIFIED\_D>

<CREATED\_DA/>

<PHOTO>https://s3-media4.fl.yelpcdn.com/bphoto/q3gEpGr9MO5CNLaG-G9PHg/ls.jpg</PHOTO>

<LINK>http://ottawa.ca/en/taxonomy/term/2639</LINK>

<LINK\_LABEL>Beaches</LINK\_LABEL>

<LINK\_LAB\_1>Plages</LINK\_LAB\_1>

<LINK\_DESCR/>

<LINK\_DES\_1/>

<geometry>

<type>Point</type>

<X>-75.76127098640916</X>

<Y>45.39539726038666</Y>

</geometry>

</properties>

<properties>

<FACILITYID>36113</FACILITYID>

<NAME>Britannia Beach</NAME>

<NAME\_FR>Plage Britannia</NAME\_FR>

<ADDRESS>102 Greenview Rd.</ADDRESS>

<POSTAL\_CODE>V0N 1J0</POSTAL\_CODE>

<BEACH\_TYPE>Plage</BEACH\_TYPE>

<ACCESSIBLE/>

<OPEN/>

<NOTES/>

<MODIFIED\_D>2014/09/29</MODIFIED\_D>

<CREATED\_DA/>

<PHOTO>https://outdoorottawa.com/wp-content/uploads/2017/06/britannia-beach-pano.jpg</PHOTO>

<LINK>https://www.theswimguide.org/beach/1092?set\_language=en</LINK>

<LINK\_LABEL>Beaches</LINK\_LABEL>

<LINK\_LAB\_1>Plages</LINK\_LAB\_1>

<LINK\_DESCR/>

<LINK\_DES\_1/>

<geometry>

<type>Point</type>

<X>-75.80137998477915</X>

<Y>45.36474037353495</Y>

</geometry>

</properties>

<properties>

<FACILITYID>36110</FACILITYID>

<NAME>Petrie Island Beach</NAME>

<NAME\_FR>Plage de l&#x27;île Petrie</NAME\_FR>

<ADDRESS>795 Trim Rd.</ADDRESS>

<POSTAL\_CODE>ON K4A 3P4</POSTAL\_CODE>

<BEACH\_TYPE>Plage</BEACH\_TYPE>

<ACCESSIBLE/>

<OPEN/>

<NOTES/>

<MODIFIED\_D>2014/09/29</MODIFIED\_D>

<CREATED\_DA/>

<PHOTO>https://outdoorottawa.com/wp-content/uploads/2017/06/petrie-island-east-bay-beach.jpg</PHOTO>

<LINK>http://ottawa.ca/en/taxonomy/term/2639</LINK>

<LINK\_LABEL>Beaches</LINK\_LABEL>

<LINK\_LAB\_1>Plages</LINK\_LAB\_1>

<LINK\_DESCR/>

<LINK\_DES\_1/>

<geometry>

<type>Point</type>

<X>-75.4933374</X>

<Y>45.5058593</Y>

</geometry>

</properties>

<properties>

<FACILITYID>36112</FACILITYID>

<NAME>Mooney&#x27;s Bay Beach</NAME>

<NAME\_FR>Plage de Mooney’s Bay</NAME\_FR>

<ADDRESS>2960 Riverside Dr.</ADDRESS>

<POSTAL\_CODE>ON K1V 8N4</POSTAL\_CODE>

<BEACH\_TYPE>Plage</BEACH\_TYPE>

<ACCESSIBLE/>

<OPEN/>

<NOTES/>

<MODIFIED\_D>2014/09/29</MODIFIED\_D>

<CREATED\_DA/>

<PHOTO>https://outdoorottawa.com/wp-content/uploads/2017/06/mooneys-bay.jpg</PHOTO>

<LINK>http://ottawa.ca/en/taxonomy/term/2639</LINK>

<LINK\_FR>http://ottawa.ca/fr/taxonomy/term/2639</LINK\_FR>

<LINK\_LABEL>Beaches</LINK\_LABEL>

<LINK\_LAB\_1>Plages</LINK\_LAB\_1>

<LINK\_DESCR/>

<LINK\_DES\_1/>

<geometry>

<type>Point</type>

<X>-75.69353885169848</X>

<Y>45.366430674244356</Y>

</geometry>

</properties>

</features>

</root>

* + 1. Создание модели данных для XML-файла.

using System;

using System.Xml.Serialization;

namespace assignment.Models

{

[Serializable()]

public class BeachXml

{

[XmlElement("FACILITYID")]

public long Id { get; set; }

[XmlElement("NAME")]

public string Name { get; set; }

[XmlElement("NAME\_FR")]

public string NameFr { get; set; }

[XmlElement("ADDRESS")]

public string Address { get; set; }

[XmlElement("ADDRESS\_FR")]

public string AddressFr { get; set; }

[XmlElement("BEACH\_TYPE")]

public string BeachType { get; set; }

[XmlElement("ACCESSIBLE")]

public string Accessible { get; set; }

[XmlElement("OPEN")]

public string Open { get; set; }

[XmlElement("NOTES")]

public string Notes { get; set; }

[XmlElement("MODIFIED\_D")]

public string ModifiedDate { get; set; }

[XmlElement("REATED\_DA")]

public string CreatedDate { get; set; }

[XmlElement("LINK")]

public string Link { get; set; }

[XmlElement("LINK\_FR")]

public string LinkFr { get; set; }

[XmlElement("LINK\_LABEL")]

public string LinkLabel { get; set; }

[XmlElement("LINK\_LAB\_1")]

public string LinkLab1 { get; set; }

[XmlElement("LINK\_DESCR")]

public string LinkDescr { get; set; }

[XmlElement("LINK\_DES\_1")]

public string LinkDes1 { get; set; }

[XmlElement("POSTAL\_CODE")]

public string PostalCode { get; set; }

[XmlElement("PHOTO")]

public string Photo { get; set; }

[XmlElement("geometry", typeof(CoordinatesXml))]

public CoordinatesXml Geometry { get; set; }

}

[Serializable()]

public class CoordinatesXml

{

[XmlElement("X")]

public double X { get; set; }

[XmlElement("Y")]

public double Y { get; set; }

}

[Serializable()]

[XmlRoot("root")]

public class BeachCollectionXml

{

[XmlArray("features")]

[XmlArrayItem("properties", typeof(BeachXml))]

public BeachXml[] arrBeach { get; set; }

}

}

* + 1. Создание класса для работы с XML-файлом.

using assignment.Models;

using System;

using System.Collections.Generic;

using System.IO;

using System.Xml.Serialization;

namespace assignment.Xml

{

public class XmlOperations

{

XmlSerializer serializer;

string path;

public XmlOperations()

{

serializer = new XmlSerializer(typeof(BeachCollectionXml));

path = AppDomain.CurrentDomain.BaseDirectory + "xml/beaches.xml";

}

public void Serialization(BeachXml[] arrBeach)

{

List<string> str = new List<string>();

using (FileStream fs = new FileStream(path, FileMode.OpenOrCreate))

{

serializer.Serialize(fs, arrBeach);

}

}

public BeachXml[] Deserialization()

{

List<string> str = new List<string>();

using (FileStream fs = new FileStream(path, FileMode.OpenOrCreate))

{

BeachCollectionXml beachCollection = (BeachCollectionXml)serializer.Deserialize(fs);

return beachCollection.arrBeach;

}

}

}

}

* + 1. Создание контроллера для вызова методов работы с XML-файлом.

GET: api/xml возвращает данные, считанные из файла.

POST: api/xml добавляет данные, считанные из файла, в базу данных.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using assignment.Models;

using assignment.Xml;

using Microsoft.AspNetCore.Mvc;

// For more information on enabling Web API for empty projects, visit https://go.microsoft.com/fwlink/?LinkID=397860

namespace assignment.Controllers

{

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

[ApiController]

public class XmlController : Controller

{

BeachContext beachContext;

public XmlController(BeachContext context)

{

beachContext = context;

}

// GET: api/<controller>

[HttpGet]

public IEnumerable<BeachXml> Get()

{

XmlOperations xml = new XmlOperations();

return xml.Deserialization().ToList();

}

// GET api/<controller>/5

[HttpGet("{id}")]

public void Get(int id)

{

}

// POST api/<controller>

[HttpPost]

public string Post([FromBody]string value)

{

try

{

XmlOperations xml = new XmlOperations();

var list = xml.Deserialization().ToList();

//beachContext.Beach.RemoveRange(beachContext.Beach.ToList());

List<BeachEf> beachEfs = new List<BeachEf>();

List<CoordinatesEf> coordinatesEfs = new List<CoordinatesEf>();

foreach (var t in list)

{

CoordinatesEf coordinatesEf = new CoordinatesEf()

{

X = t.Geometry.X,

Y = t.Geometry.Y

};

coordinatesEfs.Add(coordinatesEf);

BeachEf beachEf = new BeachEf()

{

Id = t.Id,

Name = t.Name,

NameFr = t.NameFr,

Address = t.Address,

AddressFr = t.AddressFr,

BeachType = t.BeachType,

Accessible = t.Accessible,

Open = t.Open,

Notes = t.Notes,

ModifiedDate = t.ModifiedDate,

CreatedDate = t.CreatedDate,

Link = t.Link,

LinkFr = t.LinkFr,

LinkLabel = t.LinkLabel,

LinkLab1 = t.LinkLab1,

LinkDescr = t.LinkDescr,

LinkDes1 = t.LinkDes1,

Geometry = coordinatesEf

};

beachEfs.Add(beachEf);

}

beachContext.Beach.AddRange(beachEfs);

beachContext.Geometry.AddRange(coordinatesEfs);

beachContext.SaveChanges();

return "Info added to database";

}

catch (Microsoft.EntityFrameworkCore.DbUpdateException ex)

{

if (ex.HResult == -2146233088)

return "These records are already in the database";

else

return ex.Message + "\nInner:" + ex.InnerException;

}

catch (Exception ex)

{

return ex.Message + "\nInner:" + ex.InnerException;

}

}

// PUT api/<controller>/5

[HttpPut("{id}")]

public void Put(int id, [FromBody]string value)

{

}

// DELETE api/<controller>/5

[HttpDelete("{id}")]

public void Delete(int id)

{

}

}

}

* 1. Передача данных из БД клиенту.

using assignment.Models;

using assignment.Xml;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

// For more information on enabling Web API for empty projects, visit https://go.microsoft.com/fwlink/?LinkID=397860

namespace assignment.Controllers

{

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

[ApiController]

public class BeachesController : Controller

{

private readonly BeachContext \_context;

public BeachesController(BeachContext context)

{

\_context = context;

}

// GET: api/<controller>

[HttpGet]

public async Task<ActionResult<IEnumerable<BeachEf>>> Get()

{

return await \_context.Beach.Include(l => l.Geometry).ToListAsync();

}

// GET api/<controller>/5

[HttpGet("{id}")]

public async Task<ActionResult<BeachEf>> GetTodoItem(long id)

{

var beachItem = await \_context.Beach.FindAsync(id);

if (beachItem == null)

{

return NotFound();

}

return beachItem;

}

// POST api/<controller>

[HttpPost]

public void Post([FromBody]string value)

{

}

// PUT api/<controller>/5

[HttpPut("{id}")]

public void Put(int id, [FromBody]string value)

{

}

// DELETE api/<controller>/5

[HttpDelete("{id}")]

public void Delete(int id)

{

}

}

}

* 1. Расчет дистанции между городами: клиент передает Id городов, сервер рассчитывает расстояние между ними и отправляет результат клиенту.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using assignment.Models;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

// For more information on enabling Web API for empty projects, visit https://go.microsoft.com/fwlink/?LinkID=397860

namespace assignment.Controllers

{

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

public class DistanceController : Controller

{

private readonly BeachContext \_context;

public DistanceController(BeachContext context)

{

\_context = context;

}

// GET: api/<controller>

[HttpGet]

public IEnumerable<string> Get()

{

return new string[] { "value1", "value2" };

}

// GET api/<controller>/5

[HttpGet("{id}")]

public string Get(int id)

{

return "value";

}

public class Req

{

public long first;

public long second;

}

// POST api/<controller>

[HttpPost]

public double Post([FromBody]Req value)

{

var t = \_context.Beach.Include(l => l.Geometry);

var g1 = t.Where(l => l.Id == value.first).First().Geometry;

var g2 = t.Where(l => l.Id == value.second).First().Geometry;

double dx = g1.X - g2.X;

double dy = g1.Y - g2.Y;

var res = Math.Sqrt(dx \* dx + dy \* dy);

return res;

}

// PUT api/<controller>/5

[HttpPut("{id}")]

public void Put(int id, [FromBody]string value)

{

}

// DELETE api/<controller>/5

[HttpDelete("{id}")]

public void Delete(int id)

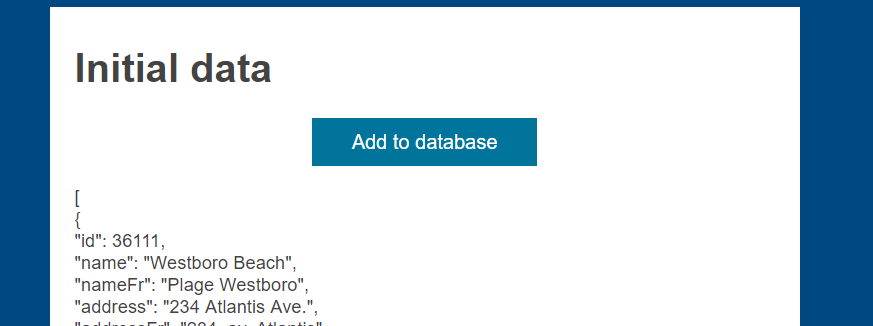
{

}

}

}

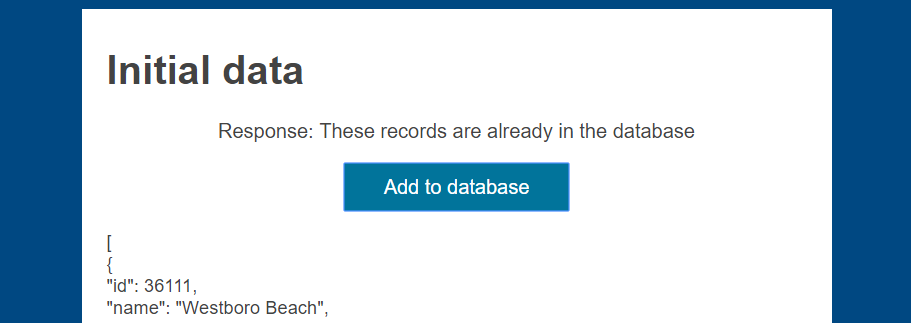
1. **Client (data.html)**



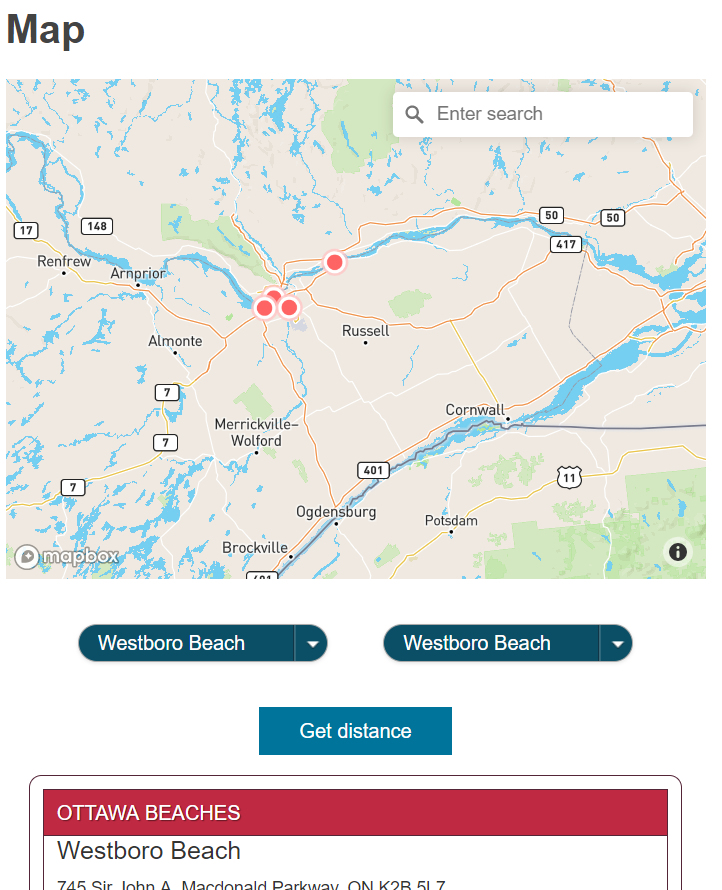
* 1. Работа с XmlController.
     1. Отображение данных и сохранение данных в БД.

Примечание: отображается в json, потому что передача данных между сервером и клиентом идет в формате json.

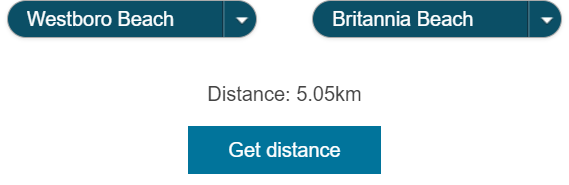
При загрузке срабатывает метод GET, при нажатии на кнопку – метод POST. Ответ метода POST отображается над кнопкой.



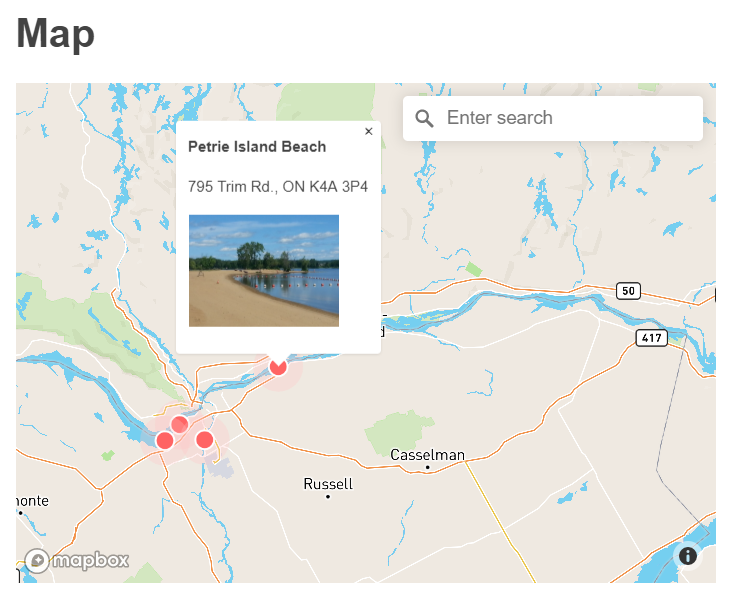
1. **Client (index.html)**



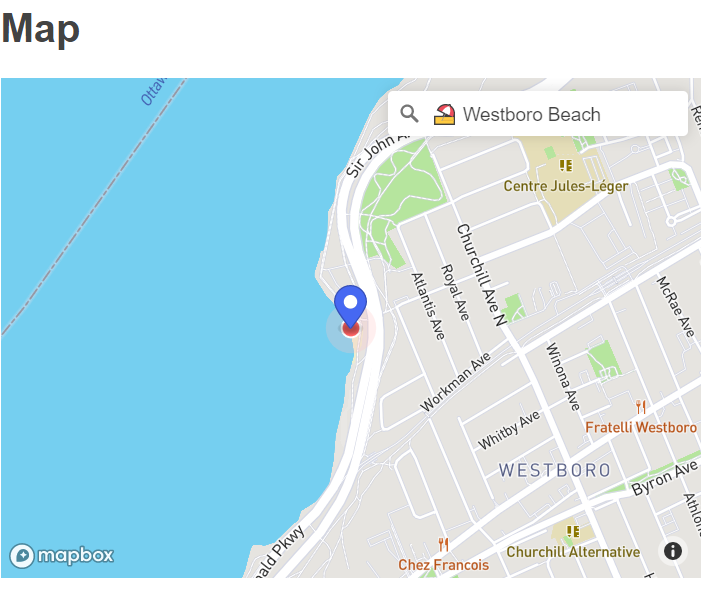
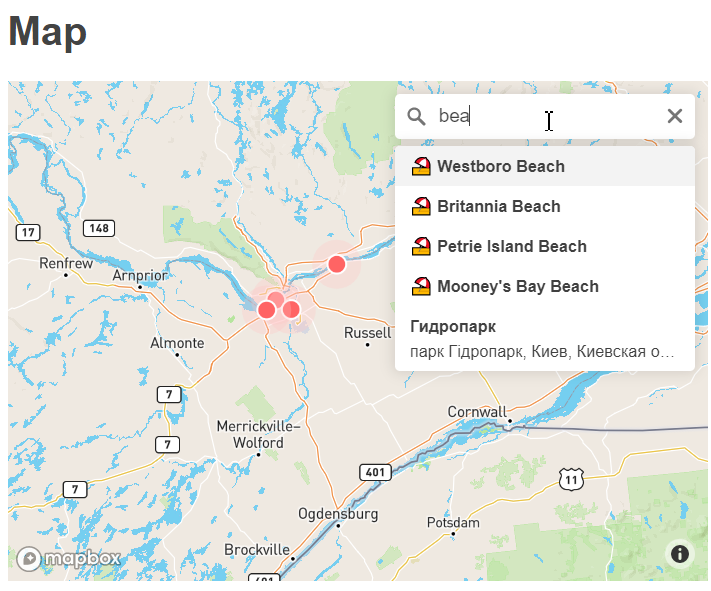
* 1. Работа с BeachesController.
     1. Считывает данные из БД.
     2. Заносит координаты и названия мест в карту.
     3. Заносит названия мест в поля со списком.
     4. Заносит краткие данные в список внизу страницы (список сворачивается/разворачивается по нажатию на OTTAWA BEACHES).
  2. Работа с DistanceController.
     1. По нажатию на кнопку отправляет значения полей со списком серверу. Получает ответ от сервера и показывает над кнопкой.



* 1. Примечания по карте.
     1. Центрирование карты по умолчанию – центр Оттавы.
     2. По нажатию на пляж отображается краткая информация и фото.



* + 1. Поиск идет по местам карты. Пляжи, отмеченные на карте, в поиске имеют специальный значок.



* 1. Примечания по клиенту.
     1. Почти весь дизайн (CSS) взят из интернета.
     2. Данные между клиентом и сервером передаются json.
     3. JS для data.html описан в data.js.
     4. JS для index.html описан в map.js.
     5. CSS для всего клиента описан в main.css.
     6. Ссылки на сайты пляжей битые, поэтому закомментировала строчку с их отображением. Если надо, можете расскоментировать (убрать // в начале строчек).

