**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ**Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**МОСКОВСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**СВЯЗИ И ИНФОРМАТИКИ**

Кафедра «Информационная безопасность»

**Практическое задание**

по дисциплине «Объектно-ориентированное программирование систем защиты информации»

Выполнила студентка группы БПЗ1802

Шилова М.Э.

Проверил:

старший преподаватель кафедры ИБ

Барков В.В.

Москва 2021 г

# 1. ЦЕЛЬ РАБОТЫ

# Разработать программный комплекс для доступа к Порталу открытых данных.

# 2. ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Серверное приложение | | | Клиентское приложение | | |
| Язык и технология и библиотеки | Архитектурный паттерн | UI паттерн | Язык и технология и библиотеки | Архитектурный паттерн | UI паттерн |
| .NET, C#, ASP.NET Core | Explicit Architecture | MVC | .NET, C#, WPF | Explicit Architecture | MVVM |
| Тема | | | | | |
| Запланированные ремонтные работы, которые могут вызвать полное или частичное перекрытие улично-дорожной сети  (<https://data.mos.ru/opendata/7701236617-zaplanirovannye-remontnye-raboty-kotorye-mogut-vyzvat-polnoe-ili-chastichnoe-perekrytie-ulichno-dorojnoy-seti>) | | | | | |

Таблица 1 – индивидуальное задание.

# 3. ПРОГРАММНЫЙ КОД

Листинг 1 – Исходный код файла DataController.cs

|  |
| --- |
| using System;  using System.IO;  using System.Collections.Generic;  using System.Threading.Tasks;  using Microsoft.AspNetCore.Mvc;  using Microsoft.EntityFrameworkCore;  using Web\_Service.Models;  using Web\_Service.Helpers;  namespace Web\_Service.Controllers  {  public class DataController : Controller  {  private readonly PlannedRenovationWorkRoadsdbMainContext db;  private string datanumber = "2587",  api\_key = "fe62b676406136d2e2020a9806809d47";  public DataController(PlannedRenovationWorkRoadsdbMainContext context)  {  db = context;  }  public async Task<IActionResult> Show()  {  return Ok(JWriter<List<PlannedRenovationWorkRoadInfo>>.Write(await db.PlannedRenovationWorkRoads.ToListAsync()));  }  public async Task<IActionResult> Create()  {  try  {  string data = Req\_h.Resp($"https://apidata.mos.ru/v1/datasets/{datanumber}/rows/?api\_key={api\_key}");  var Deser\_Obj = JDeserializer<List<PlannedRenovationWorkRoad>>.Deser(data);  db.PlannedRenovationWorkRoads.RemoveRange(db.PlannedRenovationWorkRoads);  foreach (var item in Deser\_Obj)  {  await db.AddAsync(item.Cells);  }  await db.SaveChangesAsync();  }  catch (Exception ex)  {  throw new Exception(ex.Message);  }  return View();  }  public async Task<IActionResult> Update()  {  int current\_count = await db.PlannedRenovationWorkRoads.CountAsync(), count = 0;  try  {  count = int.Parse(Req\_h.Resp($"https://apidata.mos.ru/v1/datasets/{datanumber}/count/?api\_key={api\_key}"));  if (current\_count < count)  {  string data = Req\_h.Resp($"https://apidata.mos.ru/v1/datasets/{datanumber}/rows/?api\_key={api\_key}&$orderby=global\_id%20desc&$top=" + (count - current\_count));  var Deser\_Obj = JDeserializer<List<PlannedRenovationWorkRoad>>.Deser(data);  foreach (var item in Deser\_Obj)  {  await db.AddAsync(item.Cells);  }  await db.SaveChangesAsync();  return Ok("Данные обновлены");  }  }  catch (Exception ex)  {  throw new Exception(ex.Message);  }  return View();  }  public async Task<IActionResult> Export()  {  try  {  using (StreamWriter sw = new StreamWriter("DataFile.json", false, System.Text.Encoding.Default))  {  sw.WriteLine(JWriter<List<PlannedRenovationWorkRoadInfo>>.Write(await db.PlannedRenovationWorkRoads.ToListAsync()));  }  }  catch (Exception ex)  {  throw new Exception(ex.Message);  }  return View();  }  public IActionResult Index()  {  return View();  }  }  } |

Листинг 2 – Исходный код файла HomeController.cs

|  |
| --- |
| using Microsoft.AspNetCore.Mvc;  using Microsoft.Extensions.Logging;  using Web\_Service.Models;  using System;  using System.Collections.Generic;  using System.Diagnostics;  using System.Linq;  using System.Threading.Tasks;  namespace Web\_Service.Controllers  {  public class HomeController : Controller  {  private readonly ILogger<HomeController> \_logger;  public HomeController(ILogger<HomeController> logger)  {  \_logger = logger;  }  public IActionResult Index()  {  return View();  }  public IActionResult Privacy()  {  return View();  }  [ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]  public IActionResult Error()  {  return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });  }  }  } |

Листинг 3 – Исходный код файла TableController.cs

|  |
| --- |
| using Microsoft.AspNetCore.Mvc;  using Microsoft.EntityFrameworkCore;  using System;  using System.Collections.Generic;  using System.Linq;  using System.Threading.Tasks;  using Web\_Service;  using Web\_Service.Helpers;  using Web\_Service.Models;  namespace Web\_Service.Controllers  {  public class Table : Controller  {  private readonly PlannedRenovationWorkRoadsdbMainContext db;  public Table(PlannedRenovationWorkRoadsdbMainContext context)  {  db = context;  }  public async Task<IActionResult> Index(  string searchString,  string sortOrder,  string currentFilter,  int? pageNumber)  {  if (searchString != null)  {  pageNumber = 1;  }  else  {  searchString = currentFilter;  }  ViewData["CurrentFilter"] = searchString;  var RW = from s in db.PlannedRenovationWorkRoads  select s;  if (!String.IsNullOrEmpty(searchString))  {  RW = RW.Where(s => s.Location.Contains(searchString)  || s.AdmArea.Contains(searchString)  || s.District.Contains(searchString)  || s.WorkStartDate.Contains(searchString)  || s.WorkEndDate.Contains(searchString)  || s.WorkType.Contains(searchString)  || s.WorkStatus.Contains(searchString)  || s.NameOrg.Contains(searchString));  }  int pageSize = 10;  return View(await PaginatedList<PlannedRenovationWorkRoadInfo>.CreateAsync(RW.AsNoTracking(), pageNumber ?? 1, pageSize));  }  }  } |

Листинг 4 – Исходный код файла JDeserializer.cs

|  |
| --- |
| using Newtonsoft.Json;  namespace Web\_Service.Helpers  {  static public class JDeserializer<T>  {  static public T Deser(in string data)  {  try  {  return JsonConvert.DeserializeObject<T>(data);  }  catch (System.Exception ex)  {  throw new System.Exception(ex.Message);  }  }  }  } |

Листинг 5 – Исходный код файлаJWriter.cs

|  |
| --- |
| using System.Text;  using System.IO;  using Newtonsoft.Json;  using Newtonsoft.Json.Linq;  using Web\_Service.Models;  namespace Web\_Service.Helpers  {  static public class JWriter<T>  {  static public string Write(in T collection, string current\_data = null)  {  StringBuilder sb = new StringBuilder();  StringWriter sw = new StringWriter(sb);  try  {  using (Newtonsoft.Json.JsonWriter writer = new JsonTextWriter(sw))  {  writer.Formatting = Formatting.Indented;  writer.WriteStartArray();  foreach (var item in (System.Collections.IList)collection)  {  writer.WriteStartObject();  writer.WritePropertyName("Location");  writer.WriteValue((item as PlannedRenovationWorkRoadInfo).Location);  writer.WritePropertyName("AdmArea");  writer.WriteValue((item as PlannedRenovationWorkRoadInfo).AdmArea);  writer.WritePropertyName("District");  writer.WriteValue((item as PlannedRenovationWorkRoadInfo).District);  writer.WritePropertyName("WorkStartDate");  writer.WriteValue((item as PlannedRenovationWorkRoadInfo).WorkStartDate);  writer.WritePropertyName("WorkEndDate");  writer.WriteValue((item as PlannedRenovationWorkRoadInfo).WorkEndDate);  writer.WritePropertyName("WorkType");  writer.WriteValue((item as PlannedRenovationWorkRoadInfo).WorkType);  writer.WritePropertyName("WorkStatus");  writer.WriteValue((item as PlannedRenovationWorkRoadInfo).WorkStatus);  writer.WritePropertyName("NameOrg");  writer.WriteValue((item as PlannedRenovationWorkRoadInfo).NameOrg);  writer.WriteEndObject();  }  writer.WriteEnd();  if (current\_data != "\r\n" && !string.IsNullOrEmpty(current\_data))  {  JArray current\_doc = JArray.Parse(current\_data);  JArray new\_data = JArray.Parse(sb.ToString());  var child\_new\_data = new\_data.Children();  current\_doc.Add(child\_new\_data);  return current\_doc.ToString();  }  return sb.ToString();  }  }  catch (System.Exception ex)  {  throw new System.Exception(ex.Message);  }  }  }  } |

Листинг 6 – Исходный код файла Req.cs

|  |
| --- |
| using System.IO;  using System.Net;  namespace Web\_Service.Helpers  {  static public class Req\_h  {  static public string Resp(string url\_data)  {  string data;  try  {  HttpWebRequest Request\_data = (HttpWebRequest)WebRequest.Create(url\_data);  HttpWebResponse Response\_data = (HttpWebResponse)Request\_data.GetResponse();  using (StreamReader streamReader = new StreamReader(Response\_data.GetResponseStream()))  {  data = streamReader.ReadToEnd();  }  }  catch (System.Exception ex)  {  throw new System.Exception(ex.Message);  }  return data;  }  }  } |

Листинг 7 – Исходный код файла StartUp.cs

|  |
| --- |
| using Microsoft.AspNetCore.Builder;  using Microsoft.AspNetCore.Hosting;  using Microsoft.AspNetCore.Http;  using Microsoft.Extensions.Configuration;  using Microsoft.Extensions.DependencyInjection;  using Microsoft.Extensions.Hosting;  using Microsoft.EntityFrameworkCore;  using Web\_Service.Models;  namespace Web\_Service  {  public class Startup  {  public Startup(IConfiguration configuration)  {  Configuration = configuration;  }  public IConfiguration Configuration { get; }  public void ConfigureServices(IServiceCollection services)  {  services.AddDbContext<PlannedRenovationWorkRoadsdbMainContext>(options =>  options.UseSqlite(Configuration.GetConnectionString("DefaultConnection")));  services.AddControllersWithViews();  }  public void Configure(IApplicationBuilder app, IWebHostEnvironment env)  {  if (env.IsDevelopment())  {  app.UseDeveloperExceptionPage();  }  else  {  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.UseEndpoints(endpoints =>  {    endpoints.MapControllerRoute(  name: "default",  pattern: "{controller=Home}/{action=Index}/{id?}");  });  }  }  } |

Листинг 8 – Исходный код файла PlannedRenovationWorkRoadInfo.cs

|  |
| --- |
| namespace Web\_Service.Models  {  public class PlannedRenovationWorkRoadInfo  {  public int Id { get; set; }  public string Location { get; set; }  public string AdmArea { get; set; }  public string District { get; set; }  public string WorkStartDate { get; set; }  public string WorkEndDate { get; set; }  public string WorkType { get; set; }  public string WorkStatus { get; set; }  public string NameOrg { get; set; }  }  } |

Листинг 9 – Исходный код файла PlannedRenovationWorkRoadsdbMainContext.cs

|  |
| --- |
| using Microsoft.EntityFrameworkCore;  namespace Web\_Service.Models  {  public class PlannedRenovationWorkRoadsdbMainContext : DbContext  {  public PlannedRenovationWorkRoadsdbMainContext()  {  }  public DbSet<PlannedRenovationWorkRoadInfo> PlannedRenovationWorkRoads { get; set; }  public PlannedRenovationWorkRoadsdbMainContext(DbContextOptions<PlannedRenovationWorkRoadsdbMainContext> options)  : base(options)  {  //Database.EnsureDeleted();  Database.EnsureCreated();  }  }  } |

Листинг 10 – Исходный код файла Caching.cs

|  |
| --- |
| using System;  using Microsoft.Extensions.Caching.Memory;  namespace Client\_WPF.Cache  {  public class Caching<T>  {  private MemoryCache \_cache = new MemoryCache(new MemoryCacheOptions()  {  SizeLimit = 1024  });  public T GetOrCreate(object key, T Tvalue)  {  T cacheEntry;  if (!\_cache.TryGetValue(key, out cacheEntry))  {  cacheEntry = Tvalue;  var cacheEntryOptions = new MemoryCacheEntryOptions()  .SetSize(1)  .SetPriority(CacheItemPriority.High)  .SetAbsoluteExpiration(TimeSpan.FromMinutes(15));  \_cache.Set(key, cacheEntry, cacheEntryOptions);  }  return cacheEntry;  }  public void Remove(object key)  {  \_cache.Remove(key);  }  }  } |

Листинг 11 – Исходный код файла PlannedRenovationWorkRoadsdbContext.cs

|  |
| --- |
| using System;  using Microsoft.EntityFrameworkCore;  using Web\_Service.Models;  namespace Client\_WPF.Models  {  public partial class PlannedRenovationWorkRoadsdbContext : DbContext  {  public PlannedRenovationWorkRoadsdbContext()  {  }  public PlannedRenovationWorkRoadsdbContext(DbContextOptions<PlannedRenovationWorkRoadsdbContext> options) : base(options)  {  }  public virtual DbSet<PlannedRenovationWorkRoadInfo> PlannedRenovationWorkRoads { get; set; }  protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)  {  if (!optionsBuilder.IsConfigured)  {  var current\_dir = new System.IO.DirectoryInfo(System.IO.Directory.GetCurrentDirectory());  var application\_dir = current\_dir.Parent.Parent.Parent.Parent.FullName;  string db\_path = "Data Source=" + application\_dir + "\\Web Service" + "\\PlannedRenovationWorkRoads.db";  optionsBuilder.UseSqlite(db\_path);  }  }  }  } |

Листинг 12 – Исходный код файла PlannedRenovationWorkRoadsdbContext.cs

|  |
| --- |
| using System;  using Microsoft.EntityFrameworkCore;  using Web\_Service.Models;  namespace Client\_WPF.Models  {  public partial class PlannedRenovationWorkRoadsdbContext : DbContext  {  public PlannedRenovationWorkRoadsdbContext()  {  }  public PlannedRenovationWorkRoadsdbContext(DbContextOptions<PlannedRenovationWorkRoadsdbContext> options) : base(options)  {  }  public virtual DbSet<PlannedRenovationWorkRoadInfo> PlannedRenovationWorkRoads { get; set; }  protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)  {  if (!optionsBuilder.IsConfigured)  {  var current\_dir = new System.IO.DirectoryInfo(System.IO.Directory.GetCurrentDirectory());  var application\_dir = current\_dir.Parent.Parent.Parent.Parent.FullName;  string db\_path = "Data Source=" + application\_dir + "\\Web Service" + "\\PlannedRenovationWorkRoads.db";  optionsBuilder.UseSqlite(db\_path);  }  }  }  } |

Листинг 13 – Исходный код файла ApplicationViewModel.cs

|  |
| --- |
| using System.ComponentModel;  using System.Runtime.CompilerServices;  using System.Collections.Generic;  using System.Collections.ObjectModel;  using System.Linq;  using Microsoft.EntityFrameworkCore;  using Web\_Service.Models;  using Web\_Service.Helpers;  using Client\_WPF.Models;  using Client\_WPF.Cache;  using Client\_WPF.Helpers;  namespace Client\_WPF.ViewModels  {  public class ApplicationViewModel : INotifyPropertyChanged  {  public ApplicationViewModel()  {  db = new PoliceStationsdbContext();  dbSize = db.PoliceStations.Count();  PoliceStations = new ObservableCollection<PoliceStationInfo>();  Cache = new Caching<PoliceStationInfo>();  CacheList = new ObservableCollection<PoliceStationInfo>();  Favorites = JDeserializer<ObservableCollection<PoliceStationInfo>>.Deser(ReadWriter<ObservableCollection<PoliceStationInfo>>.Read())  ?? new ObservableCollection<PoliceStationInfo>();  }  private readonly PoliceStationsdbContext db;  public int dbSize { get; private set; }  public ObservableCollection<PoliceStationInfo> PoliceStations { get; set; }  private PoliceStationInfo selectedPoliceStation;  public PoliceStationInfo SelectedPoliceStation  {  get { return selectedPoliceStation; }  set  {  if (value != null)  {  if (value.Id != 0)  {  var item = Cache.GetOrCreate(value.Id.ToString(), value);  if (!CacheList.Contains(item))  CacheList.Insert(0, item);  }  else  {  CacheList.Remove(CacheList.FirstOrDefault(n => n.Id == 0));  CacheList.Insert(0, value);  }  }  selectedPoliceStation = value;  OnPropertyChanged("");  }  }  private Caching<PoliceStationInfo> Cache { get; set; }  public ObservableCollection<PoliceStationInfo> CacheList { get; set; }  public ObservableCollection<PoliceStationInfo> Favorites { get; set; }  // команда подгрузки данных в список  private RelayCommand downCommand;  public RelayCommand DownCommand  {  get  {  return downCommand ??  (downCommand = new RelayCommand(obj =>  {  int sizeList = (int)obj;  if (sizeList == 0)  sizeList = 1;  if (sizeList + 10 >= dbSize)  {  for (int i = sizeList; i < dbSize; i++)  {  PoliceStations.Add(db.PoliceStations.Include(s => s.PublicPhone).FirstOrDefault(p => p.Id == i));  }  }  else  {  for (int i = sizeList; i < sizeList + 10; i++)  {  PoliceStations.Add(db.PoliceStations.Include(s => s.PublicPhone).FirstOrDefault(p => p.Id == i));  }  }  },  (obj) => PoliceStations.Count < dbSize));  }  }  // команда добавления данных в избранное  private RelayCommand addCommand;  public RelayCommand AddCommand  {  get  {  return addCommand ??  (addCommand = new RelayCommand(obj =>  {  PoliceStationInfo elem = obj as PoliceStationInfo;  if (elem != null && elem.Id != 0)  {  ReadWriter<List<PoliceStationInfo>>.Write(new List<PoliceStationInfo>() { elem });  }  else if (elem != null)  {  System.Windows.MessageBox.Show("Выбранный вами элемент уже находится в избранном.",  "Предупреждение", System.Windows.MessageBoxButton.OK, System.Windows.MessageBoxImage.Warning);  }  },  (obj) => CacheList.Count > 0));  }  }  // команда удаления данных из кэша  private RelayCommand removeCommand;  public RelayCommand RemoveCommand  {  get  {  return removeCommand ??  (removeCommand = new RelayCommand(obj =>  {  PoliceStationInfo elem = obj as PoliceStationInfo;  if (elem != null && elem.Id != 0)  {  CacheList.Remove(elem);  Cache.Remove(elem.Id.ToString());  }  else if (elem != null)  CacheList.Remove(elem);  },  (obj) => CacheList.Count > 0));  }  }  // команда обновления вкладки избранное  private RelayCommand updateCommand;  public RelayCommand UpdateCommand  {  get  {  return updateCommand ??  (updateCommand = new RelayCommand(obj =>  {  List<PoliceStationInfo> temp\_list = JDeserializer<List<PoliceStationInfo>>.Deser(ReadWriter<List<PoliceStationInfo>>.Read());  if (temp\_list != null)  {  if (Favorites.Count == 0)  {  foreach (var item in temp\_list)  {  Favorites.Insert(0, item);  }  }  else  {  bool found;  foreach (PoliceStationInfo itemT in temp\_list)  {  found = false;  foreach (var itemD in Favorites)  {  if (itemD.Name == itemT.Name)  found = true;  }  if (!found)  {  Favorites.Insert(0, itemT);  }  }  }  }  }  ));  }  }  // команда очистки избранного  private RelayCommand clearCommand;  public RelayCommand ClearCommand  {  get  {  return clearCommand ??  (clearCommand = new RelayCommand(obj =>  {  ReadWriter<object>.Write(null);  Favorites.Clear();  },  (obj) => Favorites.Count > 0));  }  }  public event PropertyChangedEventHandler PropertyChanged;  public void OnPropertyChanged([CallerMemberName] string prop = "")  {  if (PropertyChanged != null)  PropertyChanged(this, new PropertyChangedEventArgs(prop));  }  }  } |

# 4. СКРИНШОТЫ

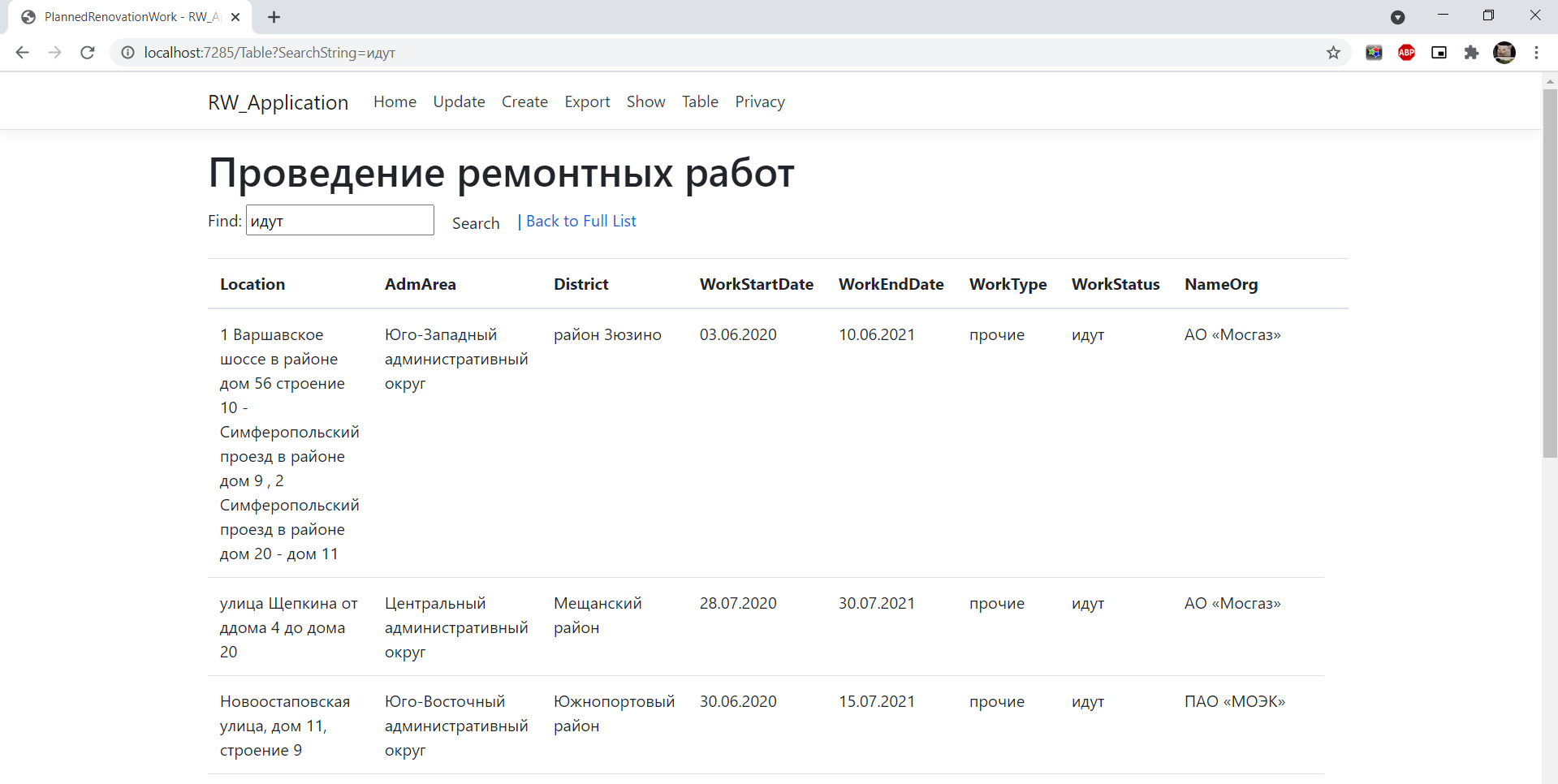


Рисунок 1 – скриншот выполнения Web\_Service.

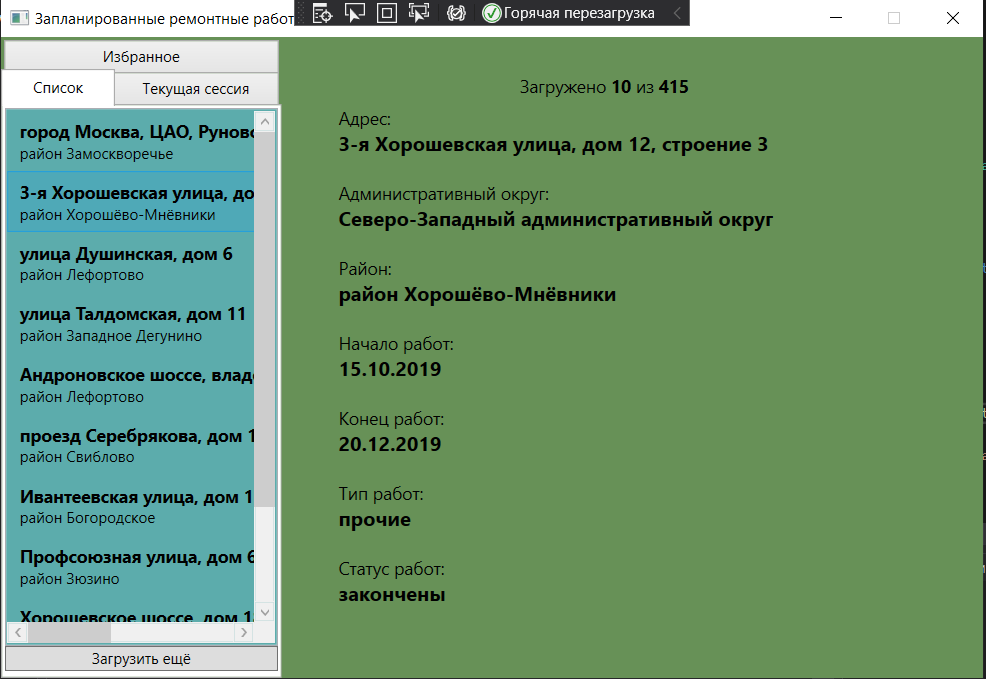


Рисунок 2 – скриншот выполнения Web\_Service.

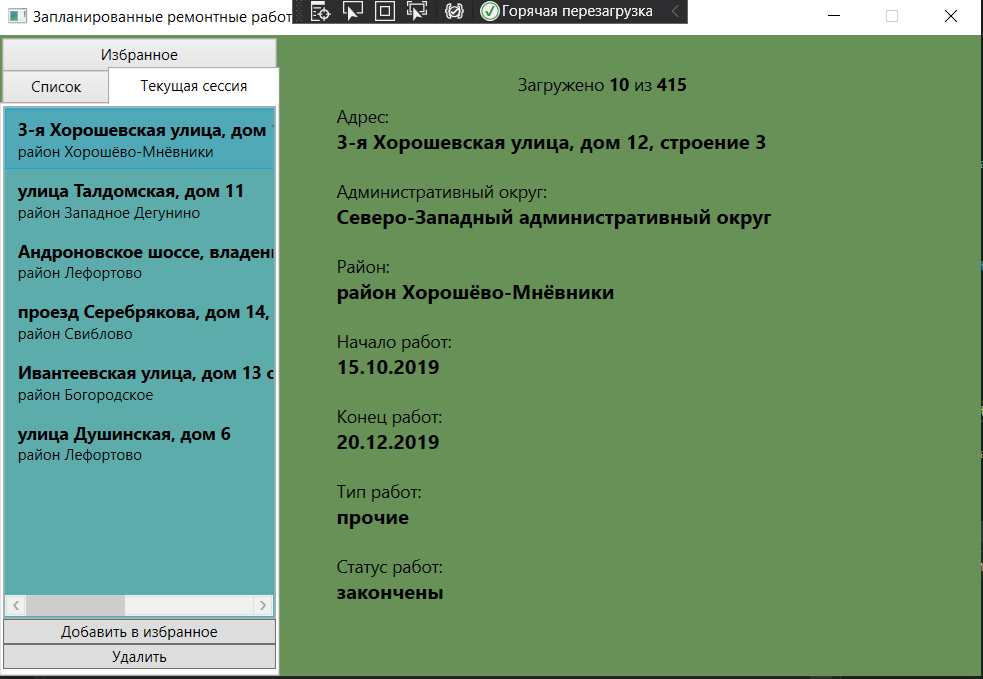


Рисунок 3 – скриншот выполнения Client\_WPF.

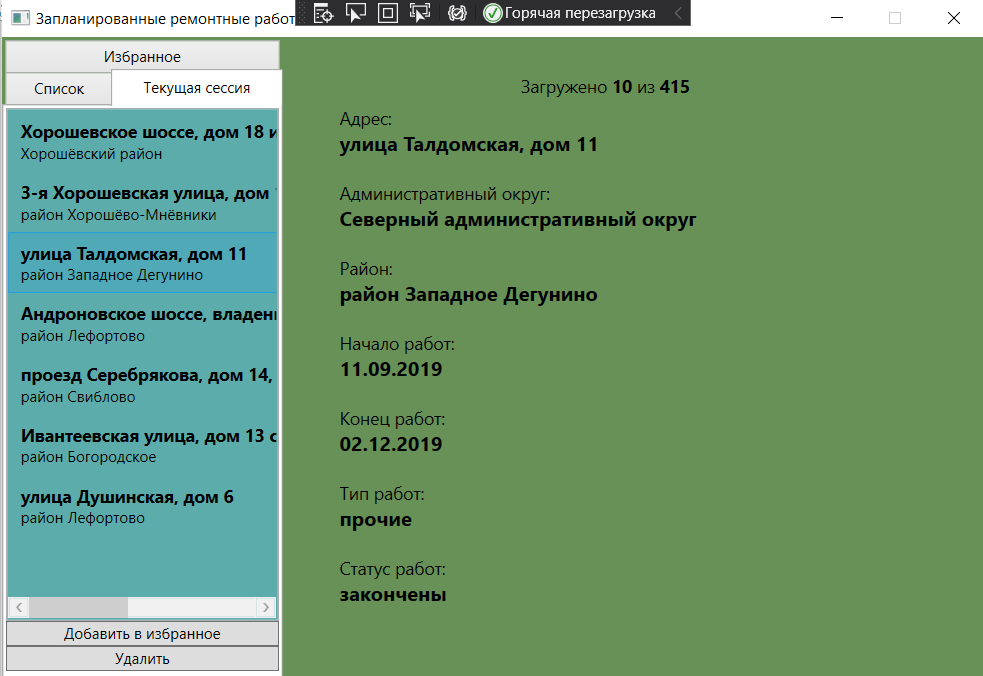


Рисунок 4 – скриншот выполнения Client\_WPF.

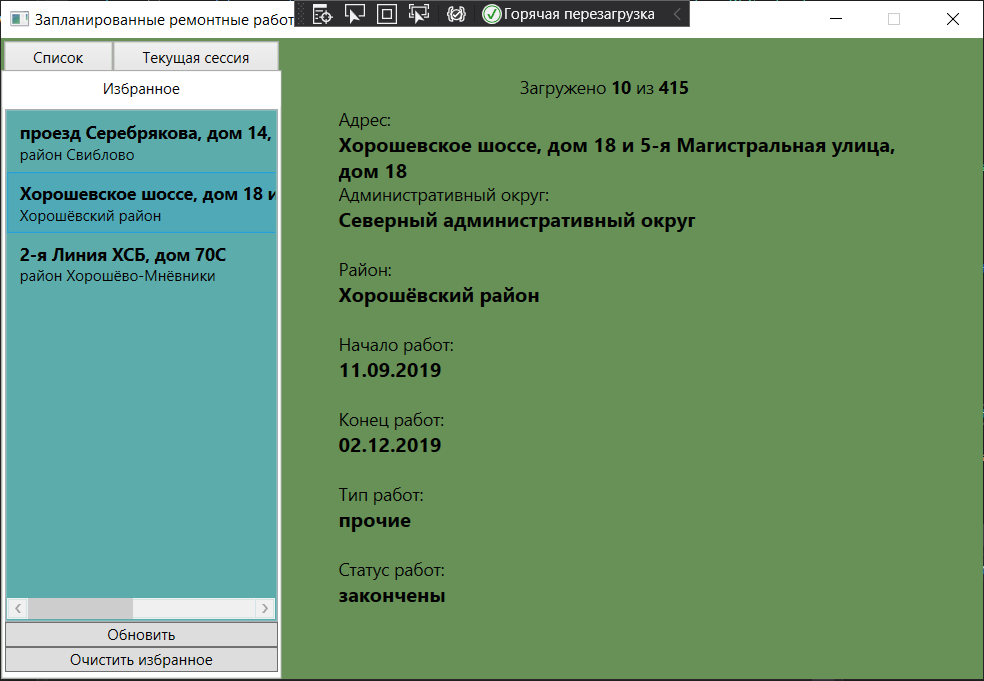


Рисунок 5 – скриншот выполнения Client\_WPF.

https://github.com/2pa-h8/OOPSZI

# 5. ВЫВОД

# Разработали программный комплекс для доступа к Порталу открытых данных.