30.5.2024

Учебная практика

Студент группы 32/1ИСД-21К: Данилин Николай

Содержание

[1. Введение 2](#_Toc12772)

[2. Описание объекта исследования 3](#_Toc21976)

[3. Логическое моделирование 4](#_Toc3889)

[3.1. Создание таблиц и связей между ними 4](#_Toc3256)

[3.2. Написание логических словарей 7](#_Toc3036)

[4. Физическое моделирование 13](#_Toc2549)

[4.1. Создание базы данных в SQL SERVER MANAGEMENT STUDIO 20 13](#_Toc13580)

[4.2. Создание физической модели 14](#_Toc15318)

[4.3. Листинг SQL запросов 15](#_Toc26299)

[5. Описание программного продукта 18](#_Toc25075)

[5.1. Скриншоты работы WPF приложения 18](#_Toc3972)

[5.2. Листинг WPF приложения 20](#_Toc12375)

[6. Заключение 34](#_Toc31671)

# Введение

**Объектом исследования является компания** “XPilotKarting”.

**Актуальность данной работы обусловлена** ростом популярности картинга как вид активного отдыха.

**Цель данной работы:** разработка информационной системы для учёта картингов, клиентов и сотрудников.

**Для достижения поставленной цели, нужно:**

1. Учёт клиентов, арендованных картов и картов на паркинге, компании поставщиков и типы картов
2. Добавление, удаление и редактирование различных данных в ИС
3. Быстрое расширение ИС
4. Понятное и интуитивное управление

# Описание объекта исследования

Данная учебная практика направлена на разработку информационной системы для учёта проката картинга на разных треках, в разных частях России. Прокат картинга становится всё более популярным видом активного отдыха и для повышения качества обслуживания клиентов и эффективным управлением паркингом картинга для сотрудников, нужна качественно спроектированная ИС учёта проката картинга. Диаграмма увлечения картингом в России с 2020 по 2023 год представлена на Рисунке 1.

*Рисунок 1. Увлечение людей картингом в России*

В ИС можно будет вести учёт:

* Людей, арендующих прокат
* Компаний, которые предоставляют карты
* Виды и характеристики картов.
* Уже арендованных картов
* Картов, которые находятся на паркинге

Таким образом, внедрённая система позволит улучшить эффективность и прибыль компаниям, занимающимися прокатом картов.

# Логическое моделирование

Логическое моделирование будет включать в себя: создание таблиц и связей между ними. Объяснение этих связей между таблицами и даст понятие для проектирования физической модели данных и переноса её в базу данных. Создание таблиц и связей представлена на рисунке 2.

## Создание таблиц и связей между ними

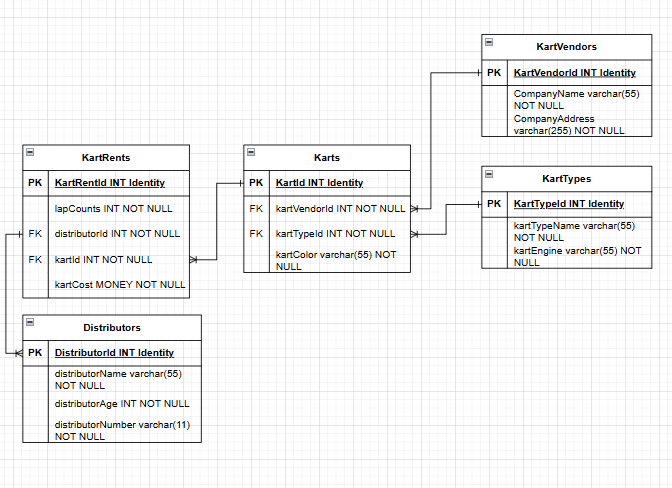


Рисунок 2. Создание таблиц и связей логической модели данных

В логической модели спроектированы следующие таблицы:

* KartVendors (Поставщики картов) – эта таблица будет содержать в себе названия и адреса компаний, которые поставляют карты на паркинг. В таблице есть следующие атрибуты:
  + KartVendorId (Идентификатор поставщика) – атрибут целочисленного типа, который является первичным ключом. По нему можно быстро будет найти нужную компанию
  + CompanyName (Название компании) – атрибут строкового типа в который будет записываться информация о названии компании
  + CompanyAddress (Адрес компании) – атрибут строкового типа в который будет записываться информация о адресе, местоположении компании
* KartTypes (Типы картов) – эта таблица будет содержать в себе названия различных конфигураций картов и двигатель. В таблице есть следующие атрибуты:
  + KartTypeId (Идентификатор типа карта) – атрибут целочисленного типа, который является первичным ключом. По нему можно быстро будет найти нужный тип карта, его название и конфигурацию.
  + KartTypeName (Название типа карта) – атрибут строкового типа в который будет записываться информация о названии типа который был присвоен карту.
  + KartEngine (Двигатель карта) – атрибут строкового типа в который будет записываться информация о двигателе, который был поставлен в текущий тип карта.
* Karts (Карты, которые есть на паркинге) – эта таблица будет содержать в себе карты, которые есть на паркинге, их поставщика и цвет. В таблице есть следующие атрибуты:
  + KartId (Идентификатор карта) – атрибут целочисленного типа, который является первичным ключом. По нему можно быстро будет найти нужный карт на паркинге
  + KartVendorId (Идентификатор поставщика) – атрибут целочисленного типа, который является вторичным ключом. В него записывается информация о поставщике этого карта.
  + KartTypeId (Идентификатор типа карта) – атрибут целочисленного типа, который является вторичным ключом. В него записывается информация каким типом является текущий карт.
  + KartColor (Цвет карта) - атрибут строкового типа в который будет записываться информация о цвете карта.
* KartRents (Арендованные карты) – эта таблица будет содержать в себе арендованные карты. В таблице есть следующие атрибуты:
  + KartRentId (Идентификатор арендованного карта) – атрибут целочисленного типа, который является первичным ключом. По нему можно быстро будет найти нужный арендованный карт.
  + LapsCount (Кол-во кругов) – атрибут целочисленного типа в который будет записываться информация о кол-ве кругов для арендованного карта.
  + DistributorId (Идентификатор арендующего) – атрибут целочисленного типа, который является вторичным ключом. В него будет записываться информация о арендующем.
  + KartId (Идентификатор карта) – атрибут целочисленного типа, который является вторичным ключом.
  + kartCost (Стоимость аренды) – атрибут денежного типа. В него будет записываться стоимость аренды карта.
* Distributors (Арендующие) – эта таблица будет содержать в себе людей, арендующие карты. В таблице есть следующие атрибуты:
  + DistributorId (Идентификатор арендующего) – атрибут целочисленного типа, который является первичным ключом. По нему можно быстро будет найти нужного человека, арендующего карт.
  + DistributorName (Имя арендующего) – атрибут строкового типа в который будет записываться информация имя арендующего.
  + DistributorAge (Возраст арендующего) – атрибут целочисленного типа. В него будет записываться информация о возрасте арендующего.
  + DistributorNumber (Номер арендующего) – атрибут целочисленного типа. В него записывается номер арендующего.

## Написание логических словарей . Логический словарь данных kartTypes представлен на таблице 1.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| kartTypes | | | | |
| Ключ | Название аттрибута | Тип данных | Is Null | Описание |
| Первичный ключ (PK) | KartTypeId | INT | 0 | атрибут целочисленного типа, который является первичным ключом. По нему можно быстро будет найти нужный тип карта, его название и конфигурацию. |
| － | KartTypeName | STRING | 0 | атрибут строкового типа в который будет записываться информация о названии типа который был присвоен карту. |
| － | KartEngine | STRING | 0 | атрибут строкового типа в который будет записываться информация о двигателе, который был поставлен в текущий тип карта. |

Таблица 1. таблица kartTypes

Логический словарь таблицы Karts представлен на таблице 2.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Karts | | | | |
| Ключ | Название аттрибута | Тип данных | Is Null | Описание |
| Первичный ключ (PK) | KartId | INT | 0 | атрибут целочисленного типа, который является первичным ключом. По нему можно быстро будет найти нужный карт на паркинге. |
| Вторичный ключ (FK) | KartVendorId | INT | 0 | атрибут целочисленного типа, который является вторичным ключом. В него записывается информация о поставщике этого карта. |
| Вторичный ключ (FK) | KartTypeId | INT | 0 | атрибут целочисленного типа, который является вторичным ключом. В него записывается информация каким типом является текущий карт. |
| － | KartColor | STRING | 0 | атрибут строкового типа в который будет записываться информация о цвете карта. |

Таблица 2.Логический словарь Karts

Логический словарь таблицы KartsRents представлен на таблице 3.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| KartRents | | | | |
| Ключ | Название аттрибута | Тип данных | Is Null | Описание |
| Первичный ключ (PK) | KartRentId | INT | 0 | атрибут целочисленного типа, который является первичным ключом. По нему можно быстро будет найти нужный арендованный карт. |
| － | LapsCount | INT | 0 | атрибут целочисленного типа в который будет записываться информация о кол-ве кругов для арендованного карта. |
| Вторичный ключ (FK) | DistributorId | INT | 0 | атрибут целочисленного типа, который является вторичным ключом. В него будет записываться информация о арендующем. |
| Вторичный ключ (FK) | KartId | INT | 0 | атрибут целочисленного типа, который является вторичным ключом. |
| － | kartCost | MONEY | 0 | атрибут денежного типа. В него будет записываться стоимость аренды карта. |

Таблица 3. Логический словарь kartRents

Логический словарь таблицы Distributors представлен на таблице 4.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Distributors | | | | |
| Ключ | Название аттрибута | Тип данных | Is Null | Описание |
| Первичный ключ (PK) | DistributorId | INT | 0 | атрибут целочисленного типа, который является первичным ключом. По нему можно быстро будет найти нужного человека, арендующего карт. |
| － | DistributorName | STRING | 0 | атрибут строкового типа в который будет записываться информация имя арендующего. |
| － | DistributorAge | INT | 0 | атрибут целочисленного типа. В него будет записываться информация о возрасте арендующего. |
| － | DistributorNumber | STRING | 0 | атрибут целочисленного типа. В него записывается номер арендующего. |

Таблица 4. Логический словарь distributors

# Физическое моделирование

## Создание базы данных в SQL SERVER MANAGEMENT STUDIO 20

С помощью SQL запросов создаём базу данных KartRental и создаём к ним таблицы и заполняем их данными.

База данных с таблицами представлена на рисунке 4.

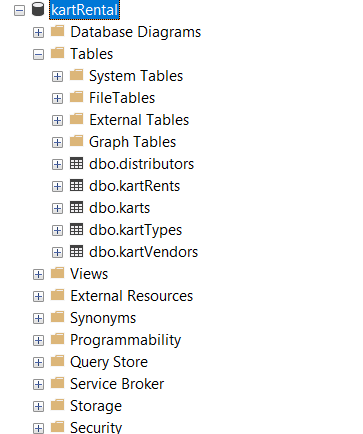


Рисунок 4. База данных с таблицами

Вывод таблиц Distributors, kartTypes и kartVendors представлена на рисунке 5.

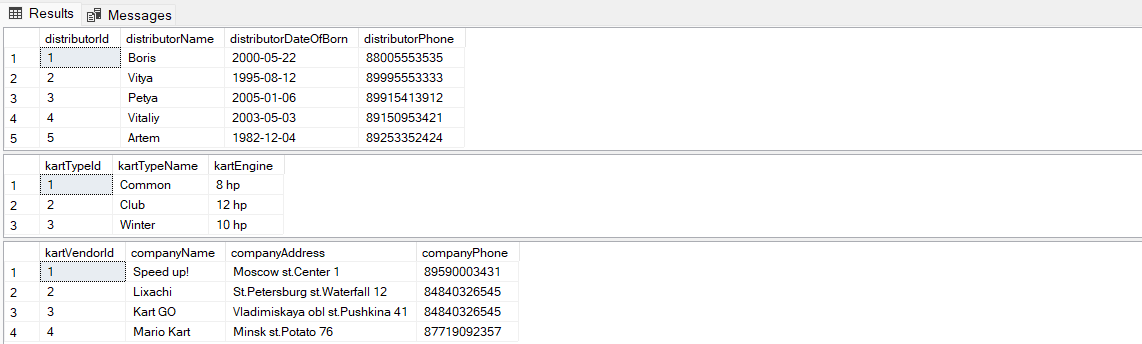


Рисунок 5. Вывод таблиц Distributors, kartTypes и kartVendors

Вывод таблиц kartRent и karts представлена на рисунке 6.

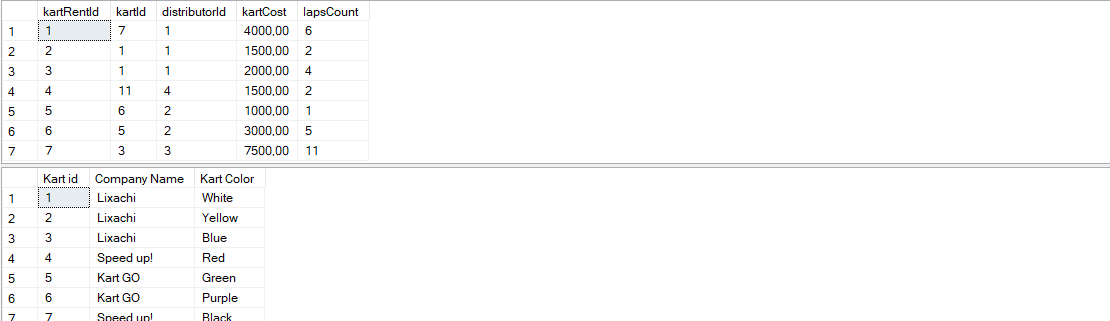


Рисунок 6. Вывод таблиц kartRent и karts

## Создание физической модели

С помощью инструментов SQL SERVER MANAGEMENT STUDIO 20 создаём физическую модель данных.Физическая модель данных представлена на рисунке 7

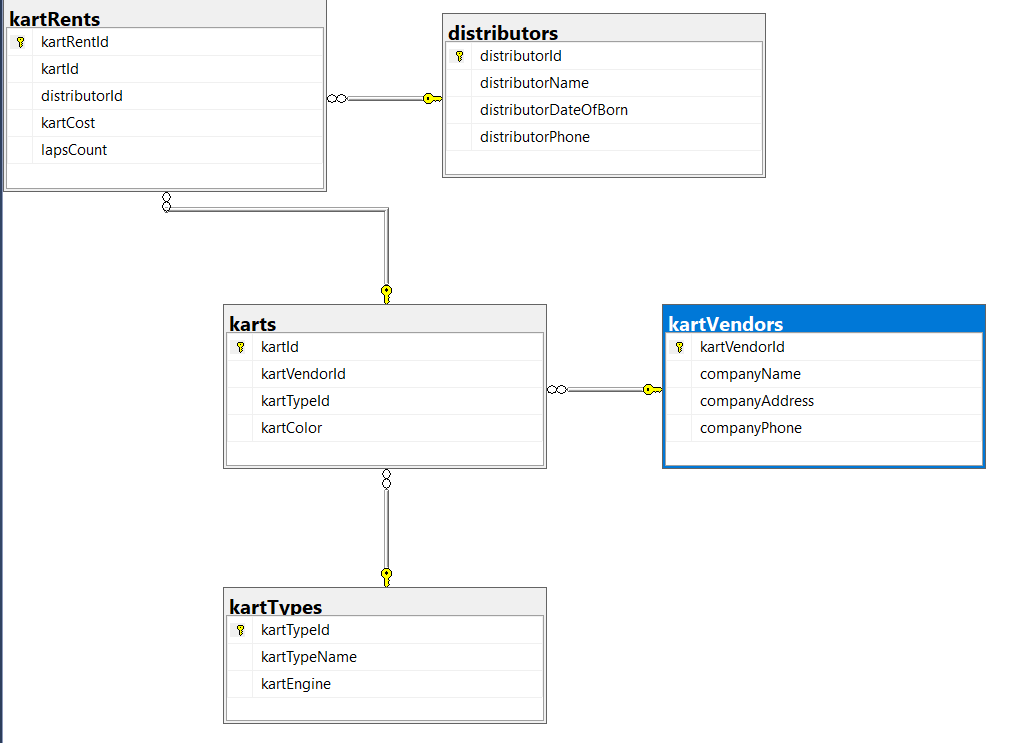


Рис 7. Физическая модель данных

## Листинг SQL запросов

Создание таблиц базы данных представлена на Листинге 1.

|  |
| --- |
| create database kartRental  use kartRental  create table distributors(  distributorId int primary key(distributorId) identity not null,  distributorName varchar(55) not null,  distributorDateOfBorn date not null,  distributorPhone varchar(11) not null  )  create table kartTypes(  kartTypeId int primary key(kartTypeId) identity not null,  kartTypeName varchar(55) not null,  kartEngine varchar(55) not null  )  create table kartVendors(  kartVendorId int primary key(kartVendorId) identity not null,  companyName varchar(55) not null,  companyAddress varchar(255) not null,  companyPhone varchar(11) not null  )  create table karts(  kartId int primary key(kartId) identity not null,  kartVendorId int foreign key(kartVendorId) references kartVendors(kartVendorId) not null,  kartTypeId int foreign key(kartTypeId) references kartTypes(kartTypeId) not null,  kartColor varchar(55) not null,  )  create table kartRents(  kartRentId int primary key(kartRentId) identity not null,  kartId int foreign key(kartId) references karts(kartId) not null,  distributorId int foreign key(distributorId) references distributors(distributorId) not null,  kartCost money not null,  lapsCount int not null,  ) |

Листинг 1. Создание таблиц базы данных

Вставка данных в таблицы базы данных представлена на Листинге 2.

|  |
| --- |
| insert into distributors values  ('Boris', '2000.05.22', '88005553535'),  ('Vitya', '1995.08.12', '89995553333'),  ('Petya', '2005.01.06', '89915413912'),  ('Vitaliy', '2003.05.03', '89150953421'),  ('Artem', '1982.12.04', '89253352424')  insert into kartTypes values  ('Common', '8 hp'),  ('Club', '12 hp'),  ('Winter', '10 hp')  insert into kartVendors values  ('Speed up!','Moscow st.Center 1','89590003431'),  ('Lixachi','St.Petersburg st.Waterfall 12','84840326545'),  ('Kart GO','Vladimiskaya obl st.Pushkina 41','84840326545'),  ('Mario Kart','Minsk st.Potato 76','87719092357')  insert into karts values  (2, 1, 'White'),  (2, 1, 'Yellow'),  (2, 1, 'Blue'),  (1, 3, 'Red'),  (3, 1, 'Green'),  (3, 1, 'Purple'),  (1, 3, 'Black'),  (1, 3, 'Blue'),  (3, 2, 'Pink'),  (1, 2, 'Yellow'),  (2, 2, 'Green')  insert into kartRents values  (7, 1, 4000, 6),  (1, 1, 1500, 2),  (1, 1, 2000, 4),  (11, 4, 1500, 2),  (6, 2, 1000, 1),  (5, 2, 3000, 5), |

Листинг 2. Distributors 1

|  |
| --- |
| drop table distributors  drop table kartTypes  drop table kartVendors  drop table karts  drop table kartRents |

Листинг 3. Листинг удаления таблиц

# Описание программного продукта

Приложение реализовано с помощью WPF, с подключенным к нему API. Оно предоставляет просмотр клиентов, картов, типов картов, арендованных картов и их поставщиков в базе данных картинга

## Скриншоты работы WPF приложения

Вывод таблицы клиентов представлена на рисунке 8.

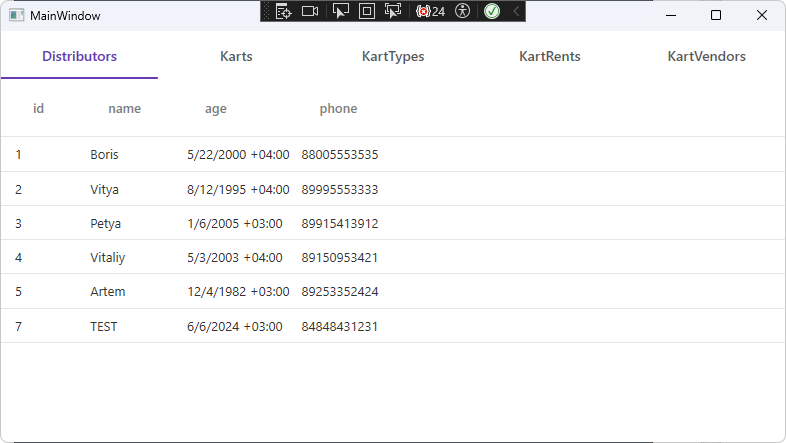


Рис 8. Вывод таблицы клиентов

Вывод таблицы картов представлена на рисунке 9.

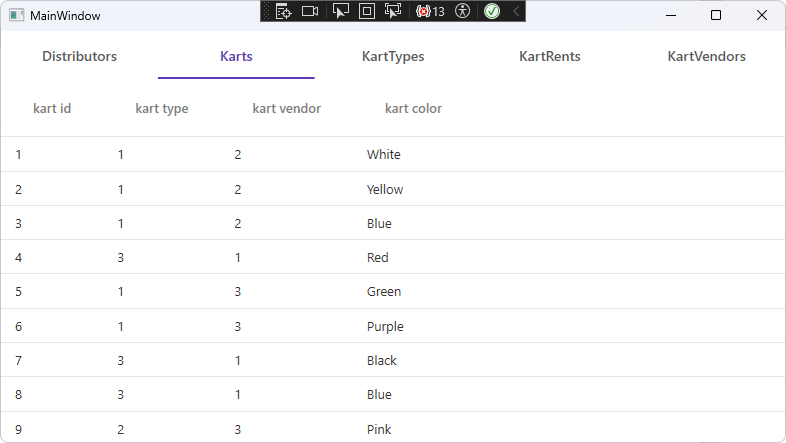
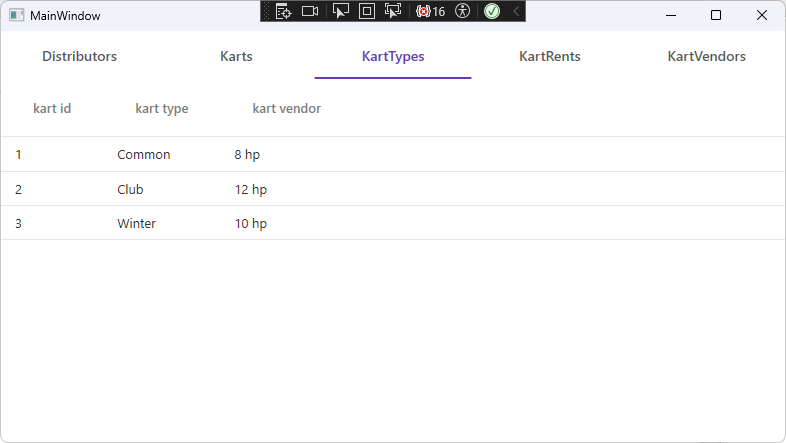


Рис 9. Вывод таблицы Картов

Вывод таблицы типов картов на рисунке 10.



1. Вывод таблицы типов картов

Вывод таблицы арендованных картов на рисунке 11.

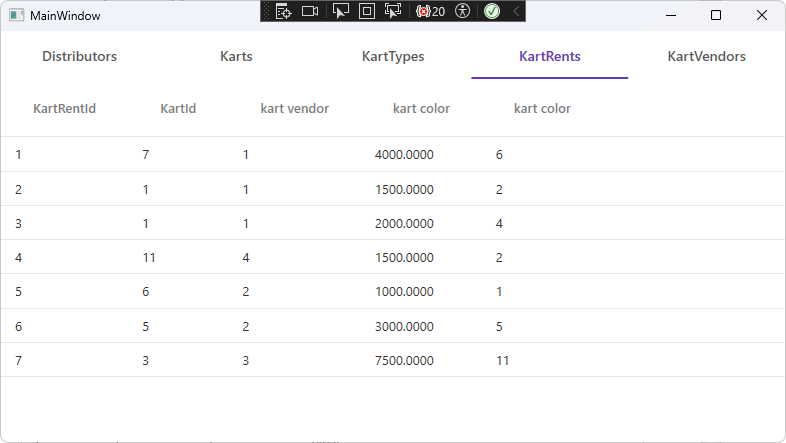


Рис 11. Вывод арендованных картов

Вывод таблицы компаний поставщиков картов на рисунке 12.

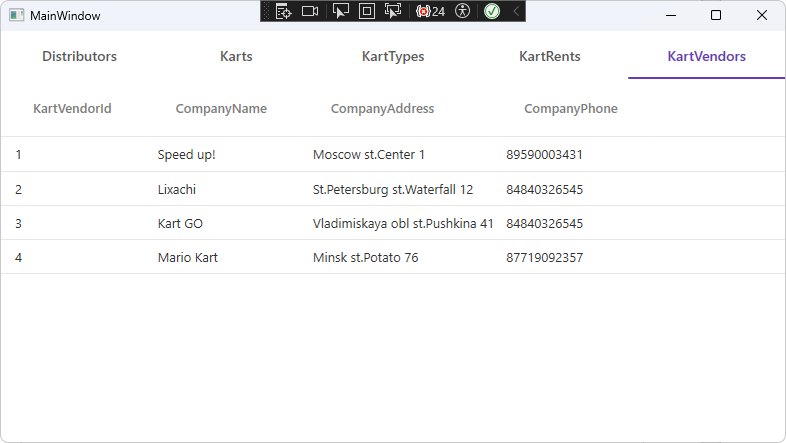


Рис 12. Вывод компаний поставщиков картов

## Листинг WPF приложения

Листинг базовой модели вывода данных представлен на Листинге 4

BaseViewModel

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.ComponentModel;  using System.Linq;  using System.Runtime.CompilerServices;  using System.Text;  using System.Threading.Tasks;  namespace KartRentalWPF.VIewModels  {  public class BaseViewModel : INotifyPropertyChanged  {  public event PropertyChangedEventHandler? PropertyChanged;  public void OnPropertyChanged([CallerMemberName] string propertyName = "")  {  PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));  }  }  } |

Листинг 4.Листинг базовой модели вывода данных

Листинг модели вывода данных клиентов представлен на Листинге 5

DistributorViewModel

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.ComponentModel;  using System.Linq;  using System.Runtime.CompilerServices;  using System.Text;  using System.Threading.Tasks;  namespace KartRentalWPF.VIewModels  {  public class BaseViewModel : INotifyPropertyChanged  {  public event PropertyChangedEventHandler? PropertyChanged;  public void OnPropertyChanged([CallerMemberName] string propertyName = "")  {  PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));  }  }  } |

Листинг 5.Модели вывода данных клиентов

Листинг модели вывода данных картов представлена на листинге 6.

KartViewModel

|  |
| --- |
| using DanilinApiClient;  using System;  using System.Collections.Generic;  using System.Linq;  using System.Net.Http;  using System.Text;  using System.Threading.Tasks;  using danilin\_MDK0104\_4;  namespace KartRentalWPF.VIewModels  {  class KartViewModel : BaseViewModel  {  private IEnumerable<Kart> \_Kart;  public IEnumerable<Kart> Karts  {  get { return \_Kart; }  set { \_Kart = value; OnPropertyChanged(); }  }  //Временный объект для добавления или изменения  private Kart \_CurrentKart;  public Kart CurrentKart  {  get { return \_CurrentKart; }  set { \_CurrentKart = value; OnPropertyChanged(); }  }  public DelegateCommand ProcessKartCommand  {  get  {  return new DelegateCommand(o =>  {  ProcessKart();  });  }  }  private void ProcessKart()  {  if (CurrentKart.KartId == 0)  {  CreateKart();  }  else  {  UpdateKart();  }  }  public DelegateCommand ClearKartCommand  {  get  {  return new DelegateCommand(o =>  {  ClearKart();  });  }  }  private void ClearKart()  {  CurrentKart = new Kart();  }  private void UpdateKart()  {  \_httpClient.UpdateKartAsync(CurrentKart);  GetKart();  }  private void CreateKart()  {  \_httpClient.CreateKartAsync(CurrentKart);  GetKart();  }  private void DeleteKart(int KartId)  {  \_httpClient.DeleteKartAsync(KartId);  GetKart();  }  public DelegateCommand DeleteKartCommand  {  get  {  return new DelegateCommand(o =>  {  DeleteKart((int)o);  });  }  }  private void GetKart()  {  Karts = \_httpClient.GetAllAsync().Result;  }  private KartsHttpClient \_httpClient;  public KartViewModel(KartsHttpClient httpClient)  {  \_httpClient = httpClient;  GetKart();  CurrentKart = new Kart();  }  }  } |

Листинг6. Модели вывода данных картов.

Листинг главной модели вывода данных представлена на листинге 7.

MainViewModel

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace KartRentalWPF.VIewModels  {  public class MainViewModel : BaseViewModel  {  public DistributorsViewModel DistributorsViewModel { get; }  public MainViewModel(DistributorsViewModel distributorsViewModel)  {  DistributorsViewModel = distributorsViewModel;  }  }  } |

Листинг 7.главной модели вывода данных

Листинг модели представления клиентов представлен на листинге 8.

DistributorView.XAML

|  |
| --- |
| <UserControl x:Class="KartRentalWPF.Views.DistributorView"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"  xmlns:local="clr-namespace:KartRentalWPF.Views"  xmlns:materialDesign="http://materialdesigninxaml.net/winfx/xaml/themes"  xmlns:vm="clr-namespace:KartRentalWPF.VIewModels"  mc:Ignorable="d" d:DataContext="{d:DesignInstance vm:DistributorsViewModel}"  d:DesignHeight="450" d:DesignWidth="800">  <materialDesign:DialogHost IsOpen="{Binding IsDialogOpen}">  <materialDesign:DialogHost.DialogContent>  <GroupBox>  <StackPanel>  <StackPanel>  <Label Content="Имя прокатчика"/>  <TextBox Text="{Binding Distributors.distributorName}"/>  </StackPanel>  <StackPanel>  <Label Content="Возраст прокатчика"/>  <TextBox Text="{Binding Distributors.distributorAgeOfBorn}"/>  </StackPanel>  <StackPanel>  <Label Content="Телефон прокатчика"/>  <TextBox Text="{Binding Distributors.distributorPhone}"/>  </StackPanel>  <Button IsDefault="True" Command="{Binding ProcessDistributorCommand}" Content="Добавить прокатчика"  Background="#FFA0E274" BorderBrush="#FFB6FF7C" Foreground="White" FontWeight="Bold"/>  <Button IsCancel="True" Command="{Binding CloseDialogCommand}">  <StackPanel Orientation="Horizontal">  <materialDesign:PackIcon Kind="Close"/>  <TextBlock Text="Отмена"/>  </StackPanel>  </Button>  </StackPanel>  </GroupBox>  </materialDesign:DialogHost.DialogContent>  <StackPanel Background="White">  <StackPanel>  <Button Command="{Binding OpenDialogCommand}">  <StackPanel Orientation="Horizontal">  <materialDesign:PackIcon Kind="AccountAdd"/>  <TextBlock Text="Добавить"/>  </StackPanel>  </Button>  <ListView ItemsSource="{Binding Distributors}">  <ListView.View>  <GridView>  <GridViewColumn Header="id" DisplayMemberBinding="{Binding DistributorId}"></GridViewColumn>  <GridViewColumn Header="name" DisplayMemberBinding="{Binding DistributorName}"></GridViewColumn>  <GridViewColumn Header="age" DisplayMemberBinding="{Binding DistributorDateOfBorn}"></GridViewColumn>  <GridViewColumn Header="phone" DisplayMemberBinding="{Binding DistributorPhone}"></GridViewColumn>  </GridView>  </ListView.View>  </ListView>  </StackPanel>  </StackPanel>  </materialDesign:DialogHost>  </UserControl> |

Листинг 8.модели преставления данных клиента

Листинг XAML разметки приложения представлен на листинге 9.

APP.XAML

|  |
| --- |
| <Application x:Class="KartRentalWPF.App"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  xmlns:local="clr-namespace:KartRentalWPF" xmlns:materialDesign="http://materialdesigninxaml.net/winfx/xaml/themes"  xmlns:vm="clr-namespace:KartRentalWPF.VIewModels"  Startup="Application\_Startup">  <Application.Resources>  <ResourceDictionary>  <ResourceDictionary.MergedDictionaries>  <materialDesign:BundledTheme BaseTheme="Light" PrimaryColor="DeepPurple" SecondaryColor="Lime" />  <ResourceDictionary Source="pack://application:,,,/MaterialDesignThemes.Wpf;component/Themes/MaterialDesign3.Defaults.xaml" />  </ResourceDictionary.MergedDictionaries>  </ResourceDictionary>  </Application.Resources>  </Application> |

Листинг 9.XAML развёртка главного приложения

Листинг код приложения представлен на листинге 10.

APP.CS

|  |
| --- |
| using Microsoft.Extensions.DependencyInjection;  using System.Configuration;  using System.Data;  using System.Windows;  using danilin\_MDK0104\_4;  using DanilinApiClient;  using System.Net.Http;  using KartRentalWPF.VIewModels;  namespace KartRentalWPF  {  /// <summary>  /// Interaction logic for App.xaml  /// </summary>  public partial class App : Application  {  private IServiceProvider serviceProvider;  public App()  {  ServiceCollection services = new ServiceCollection();  ConfigureServices(services);  serviceProvider = services.BuildServiceProvider();  }  private void ConfigureServices(ServiceCollection services)  {  services.AddHttpClient();  const string ApiUrl = "http://localhost:5000";  services.AddSingleton<DistributorsHttpClient>(sp =>  {  var factory = sp.GetRequiredService<IHttpClientFactory>();  var client = factory.CreateClient();  var httpClient = new DistributorsHttpClient(ApiUrl, client);  return httpClient;  });  services.AddSingleton<KartRentsHttpClient>(sp =>  {  var factory = sp.GetRequiredService<IHttpClientFactory>();  var client = factory.CreateClient();  var httpClient = new KartRentsHttpClient(ApiUrl, client);  return httpClient;  });  services.AddSingleton<KartsHttpClient>(sp =>  {  var factory = sp.GetRequiredService<IHttpClientFactory>();  var client = factory.CreateClient();  var httpClient = new KartsHttpClient(ApiUrl, client);  return httpClient;  });  services.AddSingleton<KartTypesHttpClient>(sp =>  {  var factory = sp.GetRequiredService<IHttpClientFactory>();  var client = factory.CreateClient();  var httpClient = new KartTypesHttpClient(ApiUrl, client);  return httpClient;  });  services.AddSingleton<KartVendorHttpClient>(sp =>  {  var factory = sp.GetRequiredService<IHttpClientFactory>();  var client = factory.CreateClient();  var httpClient = new KartVendorHttpClient(ApiUrl, client);  return httpClient;  });  services.AddSingleton<MainWindow>();  services.AddSingleton<MainViewModel>();  services.AddSingleton<KartViewModel>();  services.AddSingleton<DistributorsViewModel>();  }  private void Application\_Startup(object sender, StartupEventArgs e)  {  var mainWindow = serviceProvider.GetService<MainWindow>();  mainWindow.Show();  }  }  } |

Листинг 10.XAML Код приложения

Листинг DelegateCommand представлен на листинге 11.

DelegateCommand.CS

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Input;  namespace danilin\_MDK0104\_4  {  public class DelegateCommand : ICommand  {  private Action<object> execute;  private Func<object, bool> canExecute;  public event EventHandler CanExecuteChanged  {  add { CommandManager.RequerySuggested += value; }  remove { CommandManager.RequerySuggested -= value; }  }  public DelegateCommand(Action<object> execute, Func<object, bool> canExecute = null)  {  this.execute = execute;  this.canExecute = canExecute;  }  public bool CanExecute(object parameter)  {  return this.canExecute == null || this.canExecute(parameter);  }  public void Execute(object parameter)  {  this.execute(parameter);  }  }  } |

Листинг 11. DelegateCommand

Листинг XAML разметки главного окна представлен на листинге 12.

MainWindow.XAML

|  |
| --- |
| <Window x:Class="KartRentalWPF.MainWindow"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"  xmlns:local="clr-namespace:KartRentalWPF"  xmlns:v="clr-namespace:KartRentalWPF.Views"  xmlns:vm="clr-namespace:KartRentalWPF.VIewModels"  mc:Ignorable="d" d:DataContext="{d:DesignInstance vm:MainViewModel}"  Title="MainWindow" Height="450" Width="800">  <TabControl>  <TabItem Header="Distributors">  <v:DistributorView DataContext="{Binding DistributorsViewModel}"/>  </TabItem>  </TabControl>  </Window> |

Листинг 12. XAML разметка главного окна

Листинг кода главного окна представлен на листинге 13.

MainWindow.CS

|  |
| --- |
| using KartRentalWPF.VIewModels;  using System.Text;  using System.Windows;  using System.Windows.Controls;  using System.Windows.Data;  using System.Windows.Documents;  using System.Windows.Input;  using System.Windows.Media;  using System.Windows.Media.Imaging;  using System.Windows.Navigation;  using System.Windows.Shapes;  namespace KartRentalWPF  {  /// <summary>  /// Interaction logic for MainWindow.xaml  /// </summary>  public partial class MainWindow : Window  {  public MainWindow(MainViewModel mainViewModel)  {  DataContext = mainViewModel;  InitializeComponent();  }  }  } |

Листинг 13. код главного окна

# Заключение

Разработка ИС для проката картингов компании «XPilotKarting» была выполнена. Теперь компании будет легко вести учёт картов, клиентов и арендованных картов, а также увеличит эффективность компании.

Теперь компания сможет привлечь ещё больший интерес и популярность этого вида активного отдыха.