**ТЕКСТ ПРОГРАММЫ**

using System;

using System.IO;

using System.Security.Cryptography;

using System.Text;

namespace OneWay.Class

{

internal class AesEncryption

{

private static readonly string key = "1234567890123456";

private static readonly string iv = "1234567890123456";

public static string EncryptString(string plainText)

{

byte[] keyBytes = Encoding.UTF8.GetBytes(key);

byte[] ivBytes = Encoding.UTF8.GetBytes(iv);

using (Aes aes = Aes.Create())

{

aes.Key = keyBytes;

aes.IV = ivBytes;

ICryptoTransform encryptor = aes.CreateEncryptor(aes.Key, aes.IV);

using (MemoryStream ms = new MemoryStream())

{

using (CryptoStream cs = new CryptoStream(ms, encryptor, CryptoStreamMode.Write))

{

using (StreamWriter sw = new StreamWriter(cs))

{

sw.Write(plainText);

}

return Convert.ToBase64String(ms.ToArray());

}

}

}

}

public static string DecryptString(string cipherText)

{

byte[] keyBytes = Encoding.UTF8.GetBytes(key);

byte[] ivBytes = Encoding.UTF8.GetBytes(iv);

byte[] cipherBytes = Convert.FromBase64String(cipherText);

using (Aes aes = Aes.Create())

{

aes.Key = keyBytes;

aes.IV = ivBytes;

ICryptoTransform decryptor = aes.CreateDecryptor(aes.Key, aes.IV);

using (MemoryStream ms = new MemoryStream(cipherBytes))

{

using (CryptoStream cs = new CryptoStream(ms, decryptor, CryptoStreamMode.Read))

{

using (StreamReader sr = new StreamReader(cs))

{

return sr.ReadToEnd();

}

}

}

}

}

}

}

namespace OneWay.Class

{

internal class Car

{

public int IdCar { get; set; }

public int IdUser { get; set; }

public string Brand { get; set; }

public string Model { get; set; }

public string Generation { get; set; }

public string Equipment { get; set; }

public string Year { get; set; }

public string Body { get; set; }

public string Drive { get; set; }

public string GearBox { get; set; }

public int Seats { get; set; }

public int Doors { get; set; }

public int? MaxSpeed { get; set; }

public string EngineType { get; set; }

public double? Volume { get; set; }

public string Fuel { get; set; }

public int? FuelOctane { get; set; }

public string Conditioner { get; set; }

public double FuelConsumption { get; set; }

public double? RangePerCharge { get; set; }

public double? BatteryCapacity { get; set; }

public Car()

{

Conditioner = "Нету";

}

}

}

using OneWay.Controls;

using System;

using System.Collections.Generic;

using System.Data.SQLite;

using System.Linq;

using System.Windows;

namespace OneWay.Class

{

internal class DataBase

{

private string connectionString;

public DataBase(string databasePath)

{

connectionString = $"Data Source={databasePath};";

}

/\* пользователи \*/

public string GetFirstNameById(int userId)

{

try

{

using (SQLiteConnection connection = new SQLiteConnection(connectionString))

{

connection.Open();

string query = "SELECT FirstName FROM User WHERE IdUser = @IdUser";

using (SQLiteCommand command = new SQLiteCommand(query, connection))

{

command.Parameters.AddWithValue("@IdUser", userId);

object result = command.ExecuteScalar();

if (result != null)

{

return result.ToString();

}

else

{

return null;

}

}

}

}

catch (Exception ex)

{

CustomMessageBox.Show("Уведомление", $"Произошла ошибка {ex.Message}.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

return null;

}

}

public bool InsertUser(User user)

{

string insertQuery = "INSERT INTO User (FirstName, Login, Password, RegistrationDate) " +

"VALUES (@FirstName, @Login, @Password, @RegistrationDate)";

string selectQuery = "SELECT COUNT(\*) FROM User WHERE Login = @Login AND Password = @Password";

try

{

using (SQLiteConnection connection = new SQLiteConnection(connectionString))

{

connection.Open();

using (SQLiteCommand selectCommand = new SQLiteCommand(selectQuery, connection))

{

selectCommand.Parameters.AddWithValue("@Login", user.Login);

selectCommand.Parameters.AddWithValue("@Password", user.Password);

object result = selectCommand.ExecuteScalar();

int userCount = result != null ? Convert.ToInt32(result) : 0;

if (userCount > 0)

{

return false;

}

}

using (SQLiteCommand insertCommand = new SQLiteCommand(insertQuery, connection))

{

insertCommand.Parameters.AddWithValue("@FirstName", user.FirstName);

insertCommand.Parameters.AddWithValue("@Login", user.Login);

insertCommand.Parameters.AddWithValue("@Password", user.Password);

insertCommand.Parameters.AddWithValue("@RegistrationDate", user.RegistrationDate);

int rowsAffected = insertCommand.ExecuteNonQuery();

if (rowsAffected > 0)

{

return true;

}

else

{

return false;

}

}

}

}

catch (Exception ex)

{

CustomMessageBox.Show("Уведомление", $"Произошла ошибка {ex.Message}.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

return false;

}

}

public int CheckUser(string login, string password)

{

string selectQuery = "SELECT IdUser FROM User WHERE Login = @Login AND Password = @Password";

try

{

using (SQLiteConnection connection = new SQLiteConnection(connectionString))

{

connection.Open();

using (SQLiteCommand command = new SQLiteCommand(selectQuery, connection))

{

command.Parameters.AddWithValue("@Login", login);

command.Parameters.AddWithValue("@Password", password);

object result = command.ExecuteScalar();

if (result != null) // проверка результата

{

int idUser = Convert.ToInt32(result);

return idUser;

}

else

{

return -1;

}

}

}

}

catch (Exception ex) // обработка исключений

{

CustomMessageBox.Show("Уведомление", $"Произошла ошибка {ex.Message}.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

return -1;

}

}

/\* автомобили \*/

public bool InsertCar(Car car)

{

string query = @"INSERT INTO Car (IdUser, Brand, Model, Generation, Equipment, Year, Body, Drive, GearBox, Seats, Doors, MaxSpeed, EngineType, Volume, Fuel, FuelOctan, FuelConsumption, Conditioner, RangePerCharge, BatteryCapacity)

VALUES (@IdUser, @Brand, @Model, @Generation, @Equipment, @Year, @Body, @Drive, @GearBox, @Seats, @Doors, @MaxSpeed, @EngineType, @Volume, @Fuel, @FuelOctan, @FuelConsumption, @Conditioner, @RangePerCharge, @BatteryCapacity)";

try

{

using (SQLiteConnection connection = new SQLiteConnection(connectionString))

{

connection.Open();

using (SQLiteCommand command = new SQLiteCommand(query, connection))

{

command.Parameters.AddWithValue("@IdUser", car.IdUser);

command.Parameters.AddWithValue("@Brand", car.Brand);

command.Parameters.AddWithValue("@Model", car.Model);

command.Parameters.AddWithValue("@Generation", car.Generation.Split('\r')[0]);

command.Parameters.AddWithValue("@Equipment", car.Equipment);

command.Parameters.AddWithValue("@Year", car.Year);

command.Parameters.AddWithValue("@Body", car.Body);

command.Parameters.AddWithValue("@Drive", car.Drive);

command.Parameters.AddWithValue("@GearBox", car.GearBox);

command.Parameters.AddWithValue("@Seats", car.Seats);

command.Parameters.AddWithValue("@Doors", car.Doors);

command.Parameters.AddWithValue("@MaxSpeed", car.MaxSpeed);

command.Parameters.AddWithValue("@EngineType", car.EngineType);

command.Parameters.AddWithValue("@Volume", car.Volume);

command.Parameters.AddWithValue("@Fuel", car.Fuel);

if (car.FuelOctane.HasValue)

{

command.Parameters.AddWithValue("@FuelOctan", car.FuelOctane);

}

else

{

command.Parameters.AddWithValue("@FuelOctan", DBNull.Value);

}

command.Parameters.AddWithValue("@Conditioner", car.Conditioner);

command.Parameters.AddWithValue("@FuelConsumption", car.FuelConsumption);

if (car.RangePerCharge.HasValue)

{

command.Parameters.AddWithValue("@RangePerCharge", car.RangePerCharge);

}

else

{

command.Parameters.AddWithValue("@RangePerCharge", DBNull.Value);

}

if (car.BatteryCapacity.HasValue)

{

command.Parameters.AddWithValue("@BatteryCapacity", car.BatteryCapacity);

}

else

{

command.Parameters.AddWithValue("@BatteryCapacity", DBNull.Value);

};

command.ExecuteNonQuery();

}

}

return true;

}

catch (Exception ex)

{

CustomMessageBox.Show("Уведомление", $"Произошла ошибка {ex.Message}.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

return false;

}

}

public bool DeleteCar(int carId)

{

try

{

using (SQLiteConnection connection = new SQLiteConnection(connectionString))

{

connection.Open();

string deleteFavoriteCarQuery = @"DELETE FROM FavoriteCar WHERE IdCar = @IdCar";

using (SQLiteCommand deleteFavoriteCarCommand = new SQLiteCommand(deleteFavoriteCarQuery, connection))

{

deleteFavoriteCarCommand.Parameters.AddWithValue("@IdCar", carId);

deleteFavoriteCarCommand.ExecuteNonQuery();

}

string deleteTripsQuery = @"DELETE FROM Trips WHERE IdCar = @IdCar";

using (SQLiteCommand deleteTripsCommand = new SQLiteCommand(deleteTripsQuery, connection))

{

deleteTripsCommand.Parameters.AddWithValue("@IdCar", carId);

deleteTripsCommand.ExecuteNonQuery();

}

string deleteCarQuery = @"DELETE FROM Car WHERE IdCar = @IdCar";

using (SQLiteCommand deleteCarCommand = new SQLiteCommand(deleteCarQuery, connection))

{

deleteCarCommand.Parameters.AddWithValue("@IdCar", carId);

int rowsAffected = deleteCarCommand.ExecuteNonQuery();

if (rowsAffected > 0)

{

return true;

}

else

{

return false;

}

}

}

}

catch (Exception ex)

{

CustomMessageBox.Show("Уведомление", $"Произошла ошибка {ex.Message}.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

return false;

}

}

public List<Car> GetAllCars(int userId)

{

List<Car> allCars = new List<Car>();

string favoriteCarsQuery = @"SELECT c.\* FROM Car c INNER JOIN FavoriteCar fc ON c.IdCar = fc.IdCar WHERE fc.IdUser = @IdUser;";

string otherCarsQuery = @"SELECT \* FROM Car WHERE IdUser = @IdUser AND IdCar NOT IN (SELECT IdCar FROM FavoriteCar WHERE IdUser = @IdUser);";

try

{

using (SQLiteConnection connection = new SQLiteConnection(connectionString))

{

connection.Open();

using (SQLiteCommand command = new SQLiteCommand(favoriteCarsQuery, connection))

{

command.Parameters.AddWithValue("@IdUser", userId);

using (SQLiteDataReader reader = command.ExecuteReader())

{

while (reader.Read())

{

Car newCar = ReadCarFromReader(reader);

allCars.Add(newCar);

}

}

}

using (SQLiteCommand command = new SQLiteCommand(otherCarsQuery, connection))

{

command.Parameters.AddWithValue("@IdUser", userId);

using (SQLiteDataReader reader = command.ExecuteReader())

{

while (reader.Read())

{

Car newCar = ReadCarFromReader(reader);

allCars.Add(newCar);

}

}

}

}

return allCars;

}

catch (Exception ex)

{

CustomMessageBox.Show("Уведомление", $"Произошла ошибка {ex.Message}.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

return new List<Car>();

}

}

public List<Car> GetFilterCars(string filter, int IdUser)

{

List<Car> allCars = new List<Car>();

string otherCarsQuery = $"SELECT \* FROM Car WHERE IdUser = @IdUser ";

if (!string.IsNullOrEmpty(filter))

{

otherCarsQuery += $" AND {filter}";

}

try

{

using (SQLiteConnection connection = new SQLiteConnection(connectionString))

{

connection.Open();

using (SQLiteCommand command = new SQLiteCommand(otherCarsQuery, connection))

{

command.Parameters.AddWithValue("@IdUser", IdUser);

using (SQLiteDataReader reader = command.ExecuteReader())

{

while (reader.Read())

{

allCars.Add(ReadCarFromReader(reader));

}

}

}

}

return allCars;

}

catch (Exception ex)

{

CustomMessageBox.Show("Уведомление", $"Произошла ошибка {ex.Message}.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

return new List<Car>();

}

}

public Car GetCar(int IdCar)

{

Car car = new Car();

string favoriteCarsQuery = $"SELECT \* FROM Car WHERE IdCar = @IdCar";

try

{

using (SQLiteConnection connection = new SQLiteConnection(connectionString))

{

connection.Open();

using (SQLiteCommand command = new SQLiteCommand(favoriteCarsQuery, connection))

{

command.Parameters.AddWithValue("@IdCar", IdCar);

using (SQLiteDataReader reader = command.ExecuteReader())

{

while (reader.Read())

{

car = ReadCarFromReader(reader);

}

}

}

}

return car;

}

catch (Exception ex)

{

CustomMessageBox.Show("Уведомление", $"Произошла ошибка {ex.Message}.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

return new Car();

}

}

private Car ReadCarFromReader(SQLiteDataReader reader)

{

return new Car

{

IdCar = reader.GetInt32(0),

IdUser = reader.GetInt32(1),

Brand = reader.GetString(2),

Model = reader.GetString(3),

Generation = reader.GetString(4),

Equipment = reader.GetString(5),

Year = reader.GetString(6),

Body = reader.GetString(7),

Drive = reader.GetString(8),

GearBox = reader.GetString(9),

Seats = reader.GetInt32(10),

Doors = reader.GetInt32(11),

MaxSpeed = reader.GetInt32(12),

EngineType = reader.GetString(13),

Volume = reader.IsDBNull(14) ? 0 : reader.GetDouble(14),

Fuel = reader.GetString(15),

FuelOctane = reader.IsDBNull(16) ? 0 : reader.GetInt32(16),

Conditioner = reader.GetString(17),

FuelConsumption = reader.GetDouble(18),

RangePerCharge = reader.IsDBNull(19) ? 0 : reader.GetDouble(19),

BatteryCapacity = reader.IsDBNull(20) ? 0 : reader.GetDouble(20)

};

}

/\* любимые автомобили\*/

public int CheckFavoriteCar(int userId, int carId)

{

try

{

using (SQLiteConnection connection = new SQLiteConnection(connectionString))

{

connection.Open();

string query = "SELECT COUNT(\*) FROM FavoriteCar WHERE IdUser = @IdUser AND IdCar = @IdCar";

using (SQLiteCommand command = new SQLiteCommand(query, connection))

{

command.Parameters.AddWithValue("@IdUser", userId);

command.Parameters.AddWithValue("@IdCar", carId);

int count = Convert.ToInt32(command.ExecuteScalar());

return count > 0 ? 1 : 0;

}

}

}

catch (Exception ex)

{

CustomMessageBox.Show("Уведомление", $"Произошла ошибка {ex.Message}.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

return -1;

}

}

public bool AddFavoriteCar(int userId, int carId)

{

try

{

using (SQLiteConnection connection = new SQLiteConnection(connectionString))

{

connection.Open();

string query = "INSERT INTO FavoriteCar (IdUser, IdCar) VALUES (@IdUser, @IdCar)";

using (SQLiteCommand command = new SQLiteCommand(query, connection))

{

command.Parameters.AddWithValue("@IdUser", userId);

command.Parameters.AddWithValue("@IdCar", carId);

int rowsAffected = command.ExecuteNonQuery();

return rowsAffected > 0;

}

}

}

catch (Exception ex)

{

CustomMessageBox.Show("Уведомление", $"Произошла ошибка {ex.Message}.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

return false;

}

}

public bool DeleteFavoriteCar(int userId, int carId)

{

try

{

using (SQLiteConnection connection = new SQLiteConnection(connectionString))

{

connection.Open();

string query = "DELETE FROM FavoriteCar WHERE IdUser = @IdUser AND IdCar = @IdCar";

using (SQLiteCommand command = new SQLiteCommand(query, connection))

{

command.Parameters.AddWithValue("@IdUser", userId);

command.Parameters.AddWithValue("@IdCar", carId);

int rowsAffected = command.ExecuteNonQuery();

return rowsAffected > 0;

}

}

}

catch (Exception ex)

{

CustomMessageBox.Show("Уведомление", $"Произошла ошибка {ex.Message}.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

return false;

}

}

/// Топливо

public List<Fuel> GetFuelTypes(int userId)

{

List<Fuel> fuelInfoList = new List<Fuel>();

/\* string query = @"

SELECT f.FuelType, f.OctaneNumber, fph.Price

FROM Fuel f

JOIN (SELECT IdFuel, MAX(DateChanged) AS MaxDate FROM FuelPriceHistory GROUP BY IdFuel)

AS latest\_date ON f.IdFuel = latest\_date.IdFuel

JOIN FuelPriceHistory fph ON f.IdFuel = fph.IdFuel AND fph.DateChanged = latest\_date.MaxDate

WHERE fph.IdUser = @UserId OR IdUser = -1";\*/

string query = @"

SELECT f.FuelType, f.OctaneNumber, fph.Price FROM Fuel AS f JOIN (

SELECT IdFuel, Price

FROM FuelPriceHistory

WHERE IdHistory IN (SELECT MAX(IdHistory) FROM FuelPriceHistory WHERE IdUser = @UserId OR IdUser = -1

GROUP BY IdFuel)) AS fph ON f.IdFuel = fph.IdFuel

ORDER BY f.IdFuel;";

try

{

using (SQLiteConnection connection = new SQLiteConnection(connectionString))

{

connection.Open();

using (SQLiteCommand command = new SQLiteCommand(query, connection))

{

command.Parameters.AddWithValue("@UserId", userId);

using (SQLiteDataReader reader = command.ExecuteReader())

{

while (reader.Read())

{

Fuel fuelInfo = new Fuel

{

FuelType = reader.GetString(0),

OctaneNumber = reader.GetString(1),

Price = Math.Round(reader.GetDouble(2), 2)

};

fuelInfoList.Add(fuelInfo);

}

}

}

}

return fuelInfoList;

}

catch (Exception ex)

{

CustomMessageBox.Show("Уведомление", $"Произошла ошибка {ex.Message}.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

return new List<Fuel>();

}

}

public int GetFuelId(string name)

{

string query = @"

SELECT IdFuel FROM Fuel WHERE OctaneNumber = @OctaneNumber";

try

{

using (SQLiteConnection connection = new SQLiteConnection(connectionString))

{

connection.Open();

using (SQLiteCommand command = new SQLiteCommand(query, connection))

{

command.Parameters.AddWithValue("@OctaneNumber", name);

using (SQLiteDataReader reader = command.ExecuteReader())

{

while (reader.Read())

{

return reader.GetInt32(0);

}

}

}

}

}

catch (Exception ex)

{

CustomMessageBox.Show("Уведомление", $"Произошла ошибка {ex.Message}.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

return 0;

}

return 0;

}

public string GetFuelName(int IdFuel)

{

string query = @"

SELECT OctaneNumber FROM Fuel WHERE IdFuel = @IdFuel";

try

{

using (SQLiteConnection connection = new SQLiteConnection(connectionString))

{

connection.Open();

using (SQLiteCommand command = new SQLiteCommand(query, connection))

{

command.Parameters.AddWithValue("@IdFuel", IdFuel);

using (SQLiteDataReader reader = command.ExecuteReader())

{

while (reader.Read())

{

return reader.GetString(0);

}

}

}

}

}

catch (Exception ex)

{

CustomMessageBox.Show("Уведомление", $"Произошла ошибка {ex.Message}.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

return null;

}

return null;

}

public List<(int id, string octaneNumber)> GetOctaneNumber(string fuel)

{

List<(int id, string octaneNumber)> octaneNumber = new List<(int id, string octaneNumber)>();

string query = $"SELECT IdFuel ,OctaneNumber FROM Fuel WHERE FuelType = '{fuel}'";

try

{

using (SQLiteConnection connection = new SQLiteConnection(connectionString))

{

connection.Open();

using (SQLiteCommand command = new SQLiteCommand(query, connection))

{

using (SQLiteDataReader reader = command.ExecuteReader())

{

while (reader.Read())

{

octaneNumber.Add((reader.GetInt32(0), reader.GetString(1)));

}

}

}

}

return octaneNumber;

}

catch (Exception ex)

{

CustomMessageBox.Show("Уведомление", $"Произошла ошибка {ex.Message}.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

return new List<(int id, string octaneNumber)>();

}

}

public List<Fuel> GetFilterFuelTypes(List<string> selectedFuelTypes)

{

List<Fuel> fuelInfoList = new List<Fuel>();

string query = @"SELECT f.FuelType, f.OctaneNumber, fph.Price

FROM Fuel f

JOIN ( SELECT IdFuel, MAX(DateChanged) AS MaxDate

FROM FuelPriceHistory

GROUP BY IdFuel)

AS latest\_date ON f.IdFuel = latest\_date.IdFuel

JOIN FuelPriceHistory fph ON f.IdFuel = fph.IdFuel AND fph.DateChanged = latest\_date.MaxDate";

if (selectedFuelTypes.Count > 0)

{

query += " WHERE f.FuelType IN (";

query += string.Join(",", selectedFuelTypes.Select(fuelType => $"'{fuelType}'"));

query += ")";

}

query += ";";

try

{

using (SQLiteConnection connection = new SQLiteConnection(connectionString))

{

connection.Open();

using (SQLiteCommand command = new SQLiteCommand(query, connection))

{

using (SQLiteDataReader reader = command.ExecuteReader())

{

while (reader.Read())

{

Fuel fuelInfo = new Fuel

{

FuelType = reader.GetString(0),

OctaneNumber = reader.GetString(1),

Price = Math.Round(reader.GetDouble(2), 2)

};

fuelInfoList.Add(fuelInfo);

}

}

}

}

return fuelInfoList;

}

catch (Exception ex)

{

CustomMessageBox.Show("Уведомление", $"Произошла ошибка {ex.Message}.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

return new List<Fuel>();

}

}

public string GetLastDate()

{

string query = "SELECT DateChanged FROM FuelPriceHistory WHERE IdUser = -1 ORDER BY IdHistory DESC LIMIT 1";

try

{

using (SQLiteConnection connection = new SQLiteConnection(connectionString))

{

connection.Open();

using (SQLiteCommand command = new SQLiteCommand(query, connection))

{

using (SQLiteDataReader reader = command.ExecuteReader())

{

if (reader.Read())

{

string dateChanged = reader.GetString(0);

DateTime parsedDate;

if (DateTime.TryParse(dateChanged, out parsedDate))

{

return parsedDate.ToString("dd.MM.yyyy");//////////

}

else

{

throw new Exception("Ошибка при преобразовании даты");

}

}

else

{

throw new Exception("Запись с указанным IdUser не найдена");

}

}

}

}

}

catch (Exception ex)

{

CustomMessageBox.Show("Уведомление", $"Произошла ошибка {ex.Message}.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

return null;

}

}

public List<(double price, DateTime date)> GetFuelPriceHistory(int IdFuel, int IdUser)

{

List<(double price, DateTime date)> priceHistory = new List<(double price, DateTime date)>();

string query = @"SELECT Price, DateChanged

FROM FuelPriceHistory

WHERE IdFuel = @IdFuel AND (IdUser = -1 OR IdUser = @IdUser) ORDER BY IdHistory DESC LIMIT 4";

try

{

using (SQLiteConnection connection = new SQLiteConnection(connectionString))

{

connection.Open();

using (SQLiteCommand command = new SQLiteCommand(query, connection))

{

command.Parameters.AddWithValue("@IdUser", IdUser);

command.Parameters.AddWithValue("@IdFuel", IdFuel);

using (SQLiteDataReader reader = command.ExecuteReader())

{

while (reader.Read())

{

double price = reader.GetDouble(0);

DateTime date = Convert.ToDateTime(reader.GetString(1));

priceHistory.Add((price, date));

}

}

}

}

priceHistory.Reverse();

//priceHistory.Sort((x, y) => x.date.CompareTo(y.date));

return priceHistory;

}

catch (Exception ex)

{

CustomMessageBox.Show("Уведомление", $"Произошла ошибка {ex.Message}.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

return new List<(double price, DateTime date)>();

}

}

public string GetFuel(int IdFuel)

{

string favoriteCarsQuery = $"SELECT OctaneNumber FROM Fuel WHERE IdFuel = @IdFuel";

string name = "";

try

{

using (SQLiteConnection connection = new SQLiteConnection(connectionString))

{

connection.Open();

using (SQLiteCommand command = new SQLiteCommand(favoriteCarsQuery, connection))

{

command.Parameters.AddWithValue("@IdFuel", IdFuel);

using (SQLiteDataReader reader = command.ExecuteReader())

{

while (reader.Read())

{

name = reader.GetString(0);

}

}

}

}

return name;

}

catch (Exception ex)

{

CustomMessageBox.Show("Уведомление", $"Произошла ошибка {ex.Message}.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

return null;

}

}

public bool UpdatePrice(int idFuel, int idUser, string price, string data)

{

try

{

using (SQLiteConnection connection = new SQLiteConnection(connectionString))

{

connection.Open();

if(idUser == -1)

{

string checkQuery = "SELECT COUNT(\*) FROM FuelPriceHistory WHERE IdFuel = @IdFuel AND DateChanged = @DateChanged";

using (SQLiteCommand checkCommand = new SQLiteCommand(checkQuery, connection))

{

checkCommand.Parameters.AddWithValue("@IdFuel", idFuel);

checkCommand.Parameters.AddWithValue("@DateChanged", data);

int existingRecords = Convert.ToInt32(checkCommand.ExecuteScalar());

if (existingRecords > 0)

{

return false;

}

}

}

string insertQuery = "INSERT INTO FuelPriceHistory (IdFuel, IdUser, Price, DateChanged) VALUES (@IdFuel, @IdUser, @Price, @DateChanged)";

using (SQLiteCommand command = new SQLiteCommand(insertQuery, connection))

{

command.Parameters.AddWithValue("@IdFuel", idFuel);

command.Parameters.AddWithValue("@IdUser", idUser);

command.Parameters.AddWithValue("@Price", price);

command.Parameters.AddWithValue("@DateChanged", data);

int rowsAffected = command.ExecuteNonQuery();

return rowsAffected > 0;

}

}

}

catch (Exception ex)

{

CustomMessageBox.Show("Уведомление", $"Произошла ошибка {ex.Message}.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

return false;

}

}

// Города

public bool InsertPoint(string name, string coordinates,int idUser)

{

try

{

using (SQLiteConnection connection = new SQLiteConnection(connectionString))

{

connection.Open();

string selectQuery = "SELECT COUNT(\*) FROM Points WHERE Name = @Name AND IdUser = @IdUser";

using (SQLiteCommand selectCommand = new SQLiteCommand(selectQuery, connection))

{

selectCommand.Parameters.AddWithValue("@Name", name.Split(',')[name.Split(',').Length - 1]);

selectCommand.Parameters.AddWithValue("@IdUser", idUser);

int existingCount = Convert.ToInt32(selectCommand.ExecuteScalar());

if (existingCount > 0)

{

return false;

}

}

string insertQuery = @"INSERT INTO Points (IdUser, Name, Coordinates) VALUES (@IdUser, @Name, @Coordinates)";

using (SQLiteCommand command = new SQLiteCommand(insertQuery, connection))

{

command.Parameters.AddWithValue("@IdUser", idUser);

command.Parameters.AddWithValue("@Name", name.Split(',')[name.Split(',').Length - 1]);

command.Parameters.AddWithValue("@Coordinates", coordinates);

command.ExecuteNonQuery();

}

}

return true;

}

catch (Exception ex)

{

CustomMessageBox.Show("Уведомление", $"Произошла ошибка {ex.Message}.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

return false;

}

}

public List<Points> GetAllPoints(int userId)

{

List<Points> allPoints = new List<Points>();

string query = @"SELECT IdPoint, Name, Coordinates FROM Points WHERE IdUser = @IdUser;";

try

{

using (SQLiteConnection connection = new SQLiteConnection(connectionString))

{

connection.Open();

using (SQLiteCommand command = new SQLiteCommand(query, connection))

{

command.Parameters.AddWithValue("@IdUser", userId);

using (SQLiteDataReader reader = command.ExecuteReader())

{

while (reader.Read())

{

Points newPoint = new Points

{

IdPoint = reader.GetInt32(0),

Name = reader.GetString(1),

Coordinates = !reader.IsDBNull(2) ? reader.GetString(2) : null

};

allPoints.Add(newPoint);

}

}

}

}

return allPoints;

}

catch (Exception ex)

{

CustomMessageBox.Show("Уведомление", $"Произошла ошибка {ex.Message}.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

return new List<Points>();

}

}

public int GetCityIdByName(int IdUser, string cityName)

{

try

{

using (SQLiteConnection connection = new SQLiteConnection(connectionString))

{

connection.Open();

string query = "SELECT IdPoint FROM Points WHERE Name = @Name AND IdUser = @IdUser";

using (SQLiteCommand command = new SQLiteCommand(query, connection))

{

command.Parameters.AddWithValue("@Name", cityName);

command.Parameters.AddWithValue("@IdUser", IdUser);

object result = command.ExecuteScalar();

if (result != null && int.TryParse(result.ToString(), out int cityId))

{

return cityId;

}

else

{

return -1;

}

}

}

}

catch (Exception ex)

{

CustomMessageBox.Show("Уведомление", $"Произошла ошибка {ex.Message}.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

return -1;

}

}

//маршруты

public List<Route> GetAllRoutes(int userId)

{

List<Route> allRoutes = new List<Route>();

string query = @"SELECT r.IdRoute, r.Name, r.PointOne, r.PointTwo, r.AdditionPoint, r.Distance, r.TravelTime

FROM Routes r

LEFT JOIN FavoriteRoute f ON r.IdRoute = f.IdRoute

WHERE r.IdUser = @IdUser

ORDER BY f.IdFavoriteRoute DESC;";

try

{

using (SQLiteConnection connection = new SQLiteConnection(connectionString))

{

connection.Open();

using (SQLiteCommand command = new SQLiteCommand(query, connection))

{

command.Parameters.AddWithValue("@IdUser", userId);

using (SQLiteDataReader reader = command.ExecuteReader())

{

while (reader.Read())

{

Route newRoute = new Route

{

IdRoute = reader.GetInt32(0),

Name = reader.GetString(1),

PointOne = reader.IsDBNull(2) ? (int?)null : reader.GetInt32(2),

PointTwo = reader.IsDBNull(3) ? (int?)null : reader.GetInt32(3),

AdditionPoint = reader.IsDBNull(4) ? null : reader.GetString(4),

Distance = reader.GetDouble(5),

TravelTime = reader.GetString(6)

};

allRoutes.Add(newRoute);

}

}

}

}

return allRoutes;

}

catch (Exception ex)

{

CustomMessageBox.Show("Уведомление", $"Произошла ошибка {ex.Message}.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

return new List<Route>();

}

}

public bool InsertRoute(Route route)

{

try

{

using (SQLiteConnection connection = new SQLiteConnection(connectionString))

{

connection.Open();

string selectQuery = "SELECT COUNT(\*) FROM Routes WHERE IdUser = @IdUser AND PointOne = @PointOne AND PointTwo = @PointTwo AND Distance = @Distance";

using (SQLiteCommand selectCommand = new SQLiteCommand(selectQuery, connection))

{

selectCommand.Parameters.AddWithValue("@IdUser", route.IdUser);

selectCommand.Parameters.AddWithValue("@PointOne", route.PointOne);

selectCommand.Parameters.AddWithValue("@PointTwo", route.PointTwo);

selectCommand.Parameters.AddWithValue("@Distance", route.Distance);

int existingCount = Convert.ToInt32(selectCommand.ExecuteScalar());

if (existingCount > 0)

{

return false;

}

}

string insertQuery = @"INSERT INTO Routes (IdUser, Name, PointOne, PointTwo, AdditionPoint, Distance, TravelTime)

VALUES (@IdUser, @Name, @PointOne, @PointTwo, @AdditionPoint, @Distance, @TravelTime)";

using (SQLiteCommand command = new SQLiteCommand(insertQuery, connection))

{

command.Parameters.AddWithValue("@IdUser", route.IdUser);

command.Parameters.AddWithValue("@Name", route.Name);

command.Parameters.AddWithValue("@PointOne", route.PointOne);

command.Parameters.AddWithValue("@PointTwo", route.PointTwo);

command.Parameters.AddWithValue("@AdditionPoint", route.AdditionPoint);

command.Parameters.AddWithValue("@Distance", route.Distance);

command.Parameters.AddWithValue("@TravelTime", route.TravelTime);

command.ExecuteNonQuery();

}

}

return true;

}

catch (Exception ex)

{

CustomMessageBox.Show("Уведомление", $"Произошла ошибка {ex.Message}.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

return false;

}

}

public bool DeleteRoute(int id)

{

DeleteTripByIdRoute(id);

string query = "DELETE FROM Routes WHERE IdRoute = @IdRoute;";

bool success = false;

try

{

using (SQLiteConnection connection = new SQLiteConnection(connectionString))

{

connection.Open();

using (SQLiteCommand command = new SQLiteCommand(query, connection))

{

command.Parameters.AddWithValue("@IdRoute", id);

int rowsAffected = command.ExecuteNonQuery();

success = rowsAffected > 0;

}

}

return success;

}

catch (Exception ex)

{

CustomMessageBox.Show("Уведомление", $"Произошла ошибка {ex.Message}.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

return false;

}

}

public List<Route> SelectRoute(string name, int userId)

{

List<Route> allRoutes = new List<Route>();

string query = $"SELECT IdRoute, Name, PointOne, PointTwo, AdditionPoint, Distance,TravelTime FROM Routes WHERE NAME = @Name AND IdUser = @IdUser;";

try

{

using (SQLiteConnection connection = new SQLiteConnection(connectionString))

{

connection.Open();

using (SQLiteCommand command = new SQLiteCommand(query, connection))

{

command.Parameters.AddWithValue("@Name", name);

command.Parameters.AddWithValue("@IdUser", userId);

using (SQLiteDataReader reader = command.ExecuteReader())

{

while (reader.Read())

{

Route newRoute = new Route

{

IdRoute = reader.GetInt32(0),

Name = reader.GetString(1),

PointOne = reader.IsDBNull(2) ? (int?)null : reader.GetInt32(2),

PointTwo = reader.IsDBNull(3) ? (int?)null : reader.GetInt32(3),

AdditionPoint = reader.IsDBNull(4) ? null : reader.GetString(4),

Distance = reader.GetDouble(5),

TravelTime = reader.GetString(6)

};

allRoutes.Add(newRoute);

}

}

}

}

return allRoutes;

}

catch (Exception ex)

{

CustomMessageBox.Show("Уведомление", $"Произошла ошибка {ex.Message}.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

return new List<Route>();

}

}

public (string, string, double, string, string) GetRouteInfoById(int routeId)

{

string startPoint = "";

string finishPoint = "";

double distance = 0;

string travelTime = "";

string points = "";

try

{

using (SQLiteConnection connection = new SQLiteConnection(connectionString))

{

connection.Open();

string query = "SELECT Points.Name AS DeparturePoint, Points2.Name AS ArrivalPoint, Routes.Distance, Routes.TravelTime, Routes.AdditionPoint " +

"FROM Routes " +

"JOIN Points ON Routes.PointOne = Points.IdPoint " +

"JOIN Points AS Points2 ON Routes.PointTwo = Points2.IdPoint " +

"WHERE Routes.IdRoute = @RouteId";

using (SQLiteCommand command = new SQLiteCommand(query, connection))

{

command.Parameters.AddWithValue("@RouteId", routeId);

using (SQLiteDataReader reader = command.ExecuteReader())

{

if (reader.Read())

{

startPoint = reader["DeparturePoint"].ToString();

finishPoint = reader["ArrivalPoint"].ToString();

distance = Convert.ToDouble(reader["Distance"]);

travelTime = reader["TravelTime"].ToString();

points = reader["AdditionPoint"].ToString();

}

}

}

}

}

catch (Exception ex)

{

CustomMessageBox.Show("Уведомление", $"Произошла ошибка {ex.Message}.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

}

return (startPoint, finishPoint, distance, travelTime, points);

}

//любимые маршруты

public int CheckFavoriteRoute(int userId, int routeId)

{

try

{

using (SQLiteConnection connection = new SQLiteConnection(connectionString))

{

connection.Open();

string query = "SELECT COUNT(\*) FROM FavoriteRoute WHERE IdUser = @IdUser AND IdRoute = @IdRoute";

using (SQLiteCommand command = new SQLiteCommand(query, connection))

{

command.Parameters.AddWithValue("@IdUser", userId);

command.Parameters.AddWithValue("@IdRoute", routeId);

int count = Convert.ToInt32(command.ExecuteScalar());

return count > 0 ? 1 : 0;

}

}

}

catch (Exception ex)

{

CustomMessageBox.Show("Уведомление", $"Произошла ошибка {ex.Message}.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

return -1;

}

}

public bool AddFavoriteRoute(int userId, int routeId)

{

try

{

using (SQLiteConnection connection = new SQLiteConnection(connectionString))

{

connection.Open();

string query = "INSERT INTO FavoriteRoute (IdUser, IdRoute) VALUES (@IdUser, @IdRoute)";

using (SQLiteCommand command = new SQLiteCommand(query, connection))

{

command.Parameters.AddWithValue("@IdUser", userId);

command.Parameters.AddWithValue("@IdRoute", routeId);

int rowsAffected = command.ExecuteNonQuery();

return rowsAffected > 0;

}

}

}

catch (Exception ex)

{

CustomMessageBox.Show("Уведомление", $"Произошла ошибка {ex.Message}.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

return false;

}

}

public bool DeleteFavoriteRoute(int userId, int routeId)

{

try

{

using (SQLiteConnection connection = new SQLiteConnection(connectionString))

{

connection.Open();

string query = "DELETE FROM FavoriteRoute WHERE IdUser = @IdUser AND IdRoute = @IdRoute";

using (SQLiteCommand command = new SQLiteCommand(query, connection))

{

command.Parameters.AddWithValue("@IdUser", userId);

command.Parameters.AddWithValue("@IdRoute", routeId);

int rowsAffected = command.ExecuteNonQuery();

return rowsAffected > 0;

}

}

}

catch (Exception ex)

{

CustomMessageBox.Show("Уведомление", $"Произошла ошибка {ex.Message}.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

return false;

}

}

public List<Route> SelectRoutesByPoints(int userId, int pointOne, int pointTwo)

{

List<Route> selectedRoutes = new List<Route>();

///

string query = $"SELECT IdRoute, Name, PointOne, PointTwo, AdditionPoint, Distance, TravelTime FROM Routes WHERE IdUser = @IdUser";

if (pointOne != -1)

{

query += " AND PointOne = @PointOne";

}

if (pointTwo != -1)

{

query += " AND PointTwo = @PointTwo";

}

try

{

using (SQLiteConnection connection = new SQLiteConnection(connectionString))

{

connection.Open();

using (SQLiteCommand command = new SQLiteCommand(query, connection))

{

command.Parameters.AddWithValue("@PointOne", pointOne);

command.Parameters.AddWithValue("@PointTwo", pointTwo);

command.Parameters.AddWithValue("@IdUser", userId);

using (SQLiteDataReader reader = command.ExecuteReader())

{

while (reader.Read())

{

Route newRoute = new Route

{

IdRoute = reader.GetInt32(0),

Name = reader.GetString(1),

PointOne = reader.IsDBNull(2) ? (int?)null : reader.GetInt32(2),

PointTwo = reader.IsDBNull(3) ? (int?)null : reader.GetInt32(3),

AdditionPoint = reader.IsDBNull(4) ? null : reader.GetString(4),

Distance = reader.GetDouble(5),

TravelTime = reader.GetString(6)

};

selectedRoutes.Add(newRoute);

}

}

}

}

return selectedRoutes;

}

catch (Exception ex)

{

CustomMessageBox.Show("Уведомление", $"Произошла ошибка {ex.Message}.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

return new List<Route>();

}

}

//поездки

public bool InsertTrip(Trip trip)

{

try

{

using (SQLiteConnection connection = new SQLiteConnection(connectionString))

{

connection.Open();

string insertQuery = @"INSERT INTO Trips (IdUser, IdRoute, IdCar, StartTime, FinishTime, UsedFuel, UsedMoney, People)

VALUES (@IdUser, @IdRoute, @IdCar, @StartTime, @FinishTime, @UsedFuel, @UsedMoney, @People)";

using (SQLiteCommand command = new SQLiteCommand(insertQuery, connection)) // добавление параметров запроса

{

command.Parameters.AddWithValue("@IdUser", trip.IdUser);

command.Parameters.AddWithValue("@IdRoute", trip.IdRoute);

command.Parameters.AddWithValue("@IdCar", trip.IdCar);

command.Parameters.AddWithValue("@StartTime", trip.StartTime);

command.Parameters.AddWithValue("@FinishTime", trip.FinishTime);

command.Parameters.AddWithValue("@UsedFuel", trip.UsedFuel);

command.Parameters.AddWithValue("@UsedMoney", trip.UsedMoney);

command.Parameters.AddWithValue("@People", trip.People);

command.ExecuteNonQuery();

}

}

return true;

}

catch (Exception ex)

{

CustomMessageBox.Show("Уведомление", $"Произошла ошибка {ex.Message}.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

return false;

}

}

public List<Trip> GetAllTrips()

{

List<Trip> trips = new List<Trip>();

try

{

using (SQLiteConnection connection = new SQLiteConnection(connectionString))

{

connection.Open();

string selectQuery = "SELECT \* FROM Trips";

using (SQLiteCommand command = new SQLiteCommand(selectQuery, connection))

{

using (SQLiteDataReader reader = command.ExecuteReader())

{

while (reader.Read())

{

Trip trip = new Trip

{

IdTrip = Convert.ToInt32(reader["IdTrip"]),

IdUser = Convert.ToInt32(reader["IdUser"]),

IdRoute = Convert.ToInt32(reader["IdRoute"]),

IdCar = Convert.ToInt32(reader["IdCar"]),

StartTime = Convert.ToString(reader["StartTime"]),

FinishTime = Convert.ToString(reader["FinishTime"]),

UsedFuel = Convert.ToDouble(reader["UsedFuel"]),

UsedMoney = Convert.ToDouble(reader["UsedMoney"]),

People = Convert.ToInt32(reader["People"])

};

trips.Add(trip);

}

}

}

}

}

catch (Exception ex)

{

CustomMessageBox.Show("Уведомление", $"Произошла ошибка {ex.Message}.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

}

return trips;

}

public bool DeleteTripByIdRoute(int id)

{

string query = "DELETE FROM Trips WHERE IdRoute = @IdRoute;";

bool success = false;

try

{

using (SQLiteConnection connection = new SQLiteConnection(connectionString))

{

connection.Open();

using (SQLiteCommand command = new SQLiteCommand(query, connection))

{

command.Parameters.AddWithValue("@IdRoute", id);

int rowsAffected = command.ExecuteNonQuery();

success = rowsAffected > 0;

}

}

return success;

}

catch (Exception ex)

{

CustomMessageBox.Show("Уведомление", $"Произошла ошибка {ex.Message}.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

return false;

}

}

public double GetDistance(int routeId)

{

double distance = 0;

try

{

using (SQLiteConnection connection = new SQLiteConnection(connectionString))

{

connection.Open();

string selectQuery = "SELECT Distance FROM Routes WHERE IdRoute = @IdRoute";

using (SQLiteCommand command = new SQLiteCommand(selectQuery, connection))

{

command.Parameters.AddWithValue("@IdRoute", routeId);

object result = command.ExecuteScalar();

if (result != null)

{

distance = Convert.ToDouble(result);

}

}

}

}

catch (Exception ex)

{

CustomMessageBox.Show("Уведомление", $"Произошла ошибка {ex.Message}.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

}

return distance;

}

public double GetFuelLastPrice(int? fuelId)

{

double lastPrice = 0;

try

{

using (SQLiteConnection connection = new SQLiteConnection(connectionString))

{

connection.Open();

string selectQuery = "SELECT Price FROM FuelPriceHistory WHERE IdFuel = @IdFuel ORDER BY DateChanged DESC LIMIT 1";

using (SQLiteCommand command = new SQLiteCommand(selectQuery, connection))

{

command.Parameters.AddWithValue("@IdFuel", fuelId);

object result = command.ExecuteScalar();

if (result != null)

{

lastPrice = Convert.ToDouble(result);

}

}

}

}

catch (Exception ex)

{

CustomMessageBox.Show("Уведомление", $"Произошла ошибка {ex.Message}.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

}

return lastPrice;

}

public string GetTime(int routeId)

{

string travelTime = "";

try

{

using (SQLiteConnection connection = new SQLiteConnection(connectionString))

{

connection.Open();

string selectQuery = "SELECT TravelTime FROM Routes WHERE IdRoute = @IdRoute";

using (SQLiteCommand command = new SQLiteCommand(selectQuery, connection))

{

command.Parameters.AddWithValue("@IdRoute", routeId);

object result = command.ExecuteScalar();

if (result != null)

{

travelTime = Convert.ToString(result);

}

}

}

}

catch (Exception ex)

{

CustomMessageBox.Show("Уведомление", $"Ошибка {ex.Message}.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

}

return travelTime;

}

//поездки

public List<Trip> GetAllTripByUserId(int userId)

{

List<Trip> trips = new List<Trip>();

try

{

using (SQLiteConnection connection = new SQLiteConnection(connectionString))

{

connection.Open();

string query = "SELECT \* FROM Trips WHERE IdUser = @UserId ORDER BY IdTrip DESC";

using (SQLiteCommand command = new SQLiteCommand(query, connection))

{

command.Parameters.AddWithValue("@UserId", userId);

using (SQLiteDataReader reader = command.ExecuteReader())

{

while (reader.Read())

{

Trip trip = new Trip

{

IdTrip = Convert.ToInt32(reader["IdTrip"]),

IdUser = Convert.ToInt32(reader["IdUser"]),

IdRoute = Convert.ToInt32(reader["IdRoute"]),

IdCar = Convert.ToInt32(reader["IdCar"]),

StartTime = reader["StartTime"].ToString(),

FinishTime = reader["FinishTime"].ToString(),

UsedFuel = Convert.ToDouble(reader["UsedFuel"]),

UsedMoney = Convert.ToDouble(reader["UsedMoney"]),

People = Convert.ToInt32(reader["People"])

};

trips.Add(trip);

}

}

}

}

return trips;

}

catch (Exception ex)

{

CustomMessageBox.Show("Уведомление", $"Ошибка {ex.Message}.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

return new List<Trip>();

}

}

public bool DeleteTrip(int id)

{

bool success = false;

try

{

using (SQLiteConnection connection = new SQLiteConnection(connectionString))

{

connection.Open();

string query = "DELETE FROM Trips WHERE IdTrip = @Id";

using (SQLiteCommand command = new SQLiteCommand(query, connection))

{

command.Parameters.AddWithValue("@Id", id);

int rowsAffected = command.ExecuteNonQuery();

if (rowsAffected > 0)

{

success = true;

}

}

}

return success;

}

catch (Exception ex)

{

CustomMessageBox.Show("Уведомление", $"Ошибка {ex.Message}.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

return false;

}

}

public List<Trip> GetAllTripByUserIdAndDateRange(int userId, string filter)

{

List<Trip> trips = new List<Trip>();

try

{

using (SQLiteConnection connection = new SQLiteConnection(connectionString))

{

connection.Open();

string query = $"SELECT \* FROM Trips WHERE IdUser = @UserId {filter}";

using (SQLiteCommand command = new SQLiteCommand(query, connection))

{

command.Parameters.AddWithValue("@UserId", userId);

using (SQLiteDataReader reader = command.ExecuteReader())

{

while (reader.Read())

{

Trip trip = new Trip

{

IdTrip = Convert.ToInt32(reader["IdTrip"]),

IdUser = Convert.ToInt32(reader["IdUser"]),

IdRoute = Convert.ToInt32(reader["IdRoute"]),

IdCar = Convert.ToInt32(reader["IdCar"]),

StartTime = reader["StartTime"].ToString(),

FinishTime = reader["FinishTime"].ToString(),

UsedFuel = Convert.ToDouble(reader["UsedFuel"]),

UsedMoney = Convert.ToDouble(reader["UsedMoney"]),

People = Convert.ToInt32(reader["People"])

};

trips.Add(trip);

}

}

}

}

return trips;

}

catch (Exception ex)

{

CustomMessageBox.Show("Уведомление", $"Произошла ошибка {ex.Message}.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

return new List<Trip>();

}

}

public List<Trip> GetHistory(string query)

{

List<Trip> trips = new List<Trip>();

try

{

using (SQLiteConnection connection = new SQLiteConnection(connectionString))

{

connection.Open();

using (SQLiteCommand command = new SQLiteCommand(query, connection))

{

using (SQLiteDataReader reader = command.ExecuteReader())

{

while (reader.Read())

{

Trip trip = new Trip

{

IdTrip = Convert.ToInt32(reader["IdTrip"]),

IdUser = Convert.ToInt32(reader["IdUser"]),

IdRoute = Convert.ToInt32(reader["IdRoute"]),

IdCar = Convert.ToInt32(reader["IdCar"]),

StartTime = reader["StartTime"].ToString(),

FinishTime = reader["FinishTime"].ToString(),

UsedFuel = Convert.ToDouble(reader["UsedFuel"]),

UsedMoney = Convert.ToDouble(reader["UsedMoney"]),

People = Convert.ToInt32(reader["People"])

};

trips.Add(trip);

}

}

}

}

return trips;

}

catch (Exception ex)

{

CustomMessageBox.Show("Уведомление", $"Произошла ошибка {ex.Message}.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

return new List<Trip>();

}

}

//статистика

public Trip GetTripById(int idTrip)

{

Trip trip = null;

try

{

using (SQLiteConnection connection = new SQLiteConnection(connectionString))

{

connection.Open();

string query = "SELECT \* FROM Trips WHERE IdTrip = @IdTrip";

using (SQLiteCommand command = new SQLiteCommand(query, connection))

{

command.Parameters.AddWithValue("@IdTrip", idTrip);

using (SQLiteDataReader reader = command.ExecuteReader())

{

if (reader.Read())

{

trip = new Trip

{

IdTrip = Convert.ToInt32(reader["IdTrip"]),

IdUser = Convert.ToInt32(reader["IdUser"]),

IdRoute = Convert.ToInt32(reader["IdRoute"]),

IdCar = Convert.ToInt32(reader["IdCar"]),

StartTime = reader["StartTime"].ToString(),

FinishTime = reader["FinishTime"].ToString(),

UsedFuel = Convert.ToDouble(reader["UsedFuel"]),

UsedMoney = Convert.ToDouble(reader["UsedMoney"]),

People = Convert.ToInt32(reader["People"])

};

}

}

}

}

return trip;

}

catch (Exception ex)

{

CustomMessageBox.Show("Уведомление", $"Произошла ошибка {ex.Message}.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

return new Trip();

}

}

public (int, double) GetCountAndSpendMoney(int userId)

{

int totalTrips = 0;

double totalMoneySpent = 0.0;

string query = @"SELECT

COUNT(IdTrip) AS TotalTrips,

SUM(UsedMoney) AS TotalMoney

FROM

Trips

WHERE

IdUser = @UserId

AND StartTime >= DATE('now', '-1 month');";

try

{

using (SQLiteConnection connection = new SQLiteConnection(connectionString))

{

connection.Open();

using (SQLiteCommand command = new SQLiteCommand(query, connection))

{

command.Parameters.AddWithValue("@UserId", userId);

using (SQLiteDataReader reader = command.ExecuteReader())

{

if (reader.Read())

{

if (!(reader["TotalTrips"] is DBNull))

totalTrips = Convert.ToInt32(reader["TotalTrips"]);

if (!(reader["TotalMoney"] is DBNull))

totalMoneySpent = Convert.ToDouble(reader["TotalMoney"]);

}

}

}

}

}

catch (Exception ex)

{

CustomMessageBox.Show("Уведомление", $"Произошла ошибка {ex.Message}.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

}

return (totalTrips, totalMoneySpent);

}

public int GetFavoriteCarCount(int userId)

{

int count = 0;

string query = @"SELECT

Count(IdUser) as Count

FROM FavoriteCar

WHERE IdUser = @UserId";

try

{

using (SQLiteConnection connection = new SQLiteConnection(connectionString))

{

connection.Open();

using (SQLiteCommand command = new SQLiteCommand(query, connection))

{

command.Parameters.AddWithValue("@UserId", userId);

using (SQLiteDataReader reader = command.ExecuteReader())

{

if (reader.Read())

{

count = Convert.ToInt32(reader["Count"]);

}

}

}

}

}

catch (Exception ex)

{

CustomMessageBox.Show("Уведомление", $"Произошла ошибка {ex.Message}.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

}

return count;

}

public double GetDistanceByMonth(int userId)

{

double totalDistance = 0.0;

string query = @"

SELECT

SUM(R.Distance) AS Distance

FROM

Trips AS T

JOIN

Routes AS R ON T.IdRoute = R.IdRoute

WHERE

T.IdUser = @UserId

AND T.StartTime >= DATE('now', '-1 month');";

try

{

using (SQLiteConnection connection = new SQLiteConnection(connectionString))

{

connection.Open();

using (SQLiteCommand command = new SQLiteCommand(query, connection))

{

command.Parameters.AddWithValue("@UserId", userId);

using (SQLiteDataReader reader = command.ExecuteReader())

{

if (reader.Read())

{

if (!(reader["Distance"] is DBNull))

totalDistance = Convert.ToDouble(reader["Distance"]);

}

}

}

}

}

catch (Exception ex)

{

CustomMessageBox.Show("Уведомление", $"Произошла ошибка {ex.Message}.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

}

return totalDistance;

}

}

}

namespace OneWay.Class

{

internal class Fuel

{

public string FuelType { get; set; }

public string OctaneNumber { get; set; }

public double Price { get; set; }

public Fuel() { }

}

}

namespace OneWay.Class

{

internal class Points

{

public int IdPoint { get; set; }

public string Name { get; set; }

public string Coordinates { get; set; }

public Points() { }

}

}

namespace OneWay.Class

{

internal class Route

{

public int IdRoute { get; set; }

public int IdUser { get; set; }

public string Name { get; set; }

public int? PointOne { get; set; }

public int? PointTwo { get; set; }

public string AdditionPoint { get; set; }

public double Distance { get; set; }

public string TravelTime { get; set; }

public Route() { }

}

}

namespace OneWay.Class

{

internal class Trip

{

public int IdTrip { get; set; }

public int IdUser { get; set; }

public int IdRoute { get; set; }

public int IdCar { get; set; }

public string StartTime { get; set; }

public string FinishTime { get; set; }

public double UsedFuel { get; set; }

public double UsedMoney { get; set; }

public int People { get; set; }

public Trip()

{}

}

}

namespace OneWay.Class

{

internal class User

{

public int IdUser { get; set; }

public string FirstName { get; set; }

public string Login { get; set; }

public string Password { get; set; }

public string RegistrationDate { get; set; }

public User() { }

}

}

<UserControl x:Class="OneWay.Controls.CarItem"

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:iconPacks="http://metro.mahapps.com/winfx/xaml/iconpacks"

xmlns:local="clr-namespace:OneWay.Controls" Name="this"

mc:Ignorable="d">

<Border Height="auto" Width="290" Background="Transparent" BorderBrush="{DynamicResource TextSecundaryColor}" BorderThickness="1" CornerRadius="5" Margin="10" HorizontalAlignment="Left" MouseLeftButtonDown="GridClick">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="60"/>

<RowDefinition/>

</Grid.RowDefinitions>

<TextBlock Text="{Binding CarName, ElementName=this, FallbackValue = BMW}" Grid.Row="0" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="20,10,86,10"/>

<TextBlock Text="{Binding Model, ElementName=this, FallbackValue = M6 Competition}" Grid.Row="0" FontSize ="15" Foreground="{DynamicResource TextSecundaryColor}" Margin="20,30,81,10"/>

<Button x:Name="FavoriteIcon" Width="31" Background="Transparent" Foreground="Gray" BorderThickness="0" HorizontalAlignment="Right" Margin="0,10,50,19" Click="FavoriteCar\_Click">

<Button.Template>

<ControlTemplate TargetType="Button">

<Grid Background="Transparent">

<!-- Добавлен прозрачный фон -->

<iconPacks:PackIconUnicons Kind="Favorite" HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Grid>

</ControlTemplate>

</Button.Template>

</Button>

<Button x:Name="DeleteButtom" Width="31" Background="Transparent" Foreground="Gray" BorderThickness="0" HorizontalAlignment="Right" Margin="0,10,10,19" Click="DeleteButtom\_Click">

<Button.Template>

<ControlTemplate TargetType="Button">

<Grid Background="Transparent">

<!-- Добавлен прозрачный фон -->

<iconPacks:PackIconUnicons Kind="Trash" HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Grid>

</ControlTemplate>

</Button.Template>

</Button>

<Grid Grid.Row="1" Background="Transparent">

<Grid.RowDefinitions>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="25\*"/>

<ColumnDefinition Width="90\*"/>

</Grid.ColumnDefinitions>

<!-- Двигатель -->

<iconPacks:PackIconMaterial Grid.Column="0" Kind="Engine" Height="20" Width="20" Foreground="{DynamicResource TextSecundaryColor}" VerticalAlignment="Center" Margin="21,0,0,0"/>

<TextBlock Grid.Row="0" Grid.Column="1" Text="{Binding Volume, ElementName=this, FallbackValue = 2.4 л / Дизель}" FontSize="20" TextAlignment="Center" Foreground="{DynamicResource TextSecundaryColor}" HorizontalAlignment="Left" VerticalAlignment="Center" Margin="5"/>

<!-- Коробка передач -->

<iconPacks:PackIconMaterial Grid.Row="1" Grid.Column="0" Kind="CarShiftPattern" Height="20" Width="20" Foreground="{DynamicResource TextSecundaryColor}" VerticalAlignment="Center" Margin="21,0,0,0"/>

<TextBlock Grid.Row="1" Grid.Column="1" Text="{Binding Transmission, ElementName=this, FallbackValue = Механика}" FontSize="20" TextAlignment="Center" Foreground="{DynamicResource TextSecundaryColor}" HorizontalAlignment="Left" VerticalAlignment="Center" Margin="5"/>

<!-- Кузов -->

<iconPacks:PackIconMaterial Grid.Row="2" Grid.Column="0" Kind="Car" Height="20" Width="20" Foreground="{DynamicResource TextSecundaryColor}" VerticalAlignment="Center" Margin="21,0,0,0"/>

<TextBlock Grid.Row="2" Grid.Column="1" Text="{Binding Body, ElementName=this, FallbackValue = Седан}" FontSize="20" TextAlignment="Center" Foreground="{DynamicResource TextSecundaryColor}" HorizontalAlignment="Left" VerticalAlignment="Center" Margin="5" />

<!-- Места -->

<iconPacks:PackIconMaterial Grid.Row="3" Grid.Column="0" Kind="CarSeat" Height="20" Width="20" Foreground="{DynamicResource TextSecundaryColor}" VerticalAlignment="Center" Margin="21,0,0,0"/>

<TextBlock Grid.Row="3" Grid.Column="1" Text="{Binding Seat, ElementName=this}" FontSize="20" TextAlignment="Center" Foreground="{DynamicResource TextSecundaryColor}" HorizontalAlignment="Left" VerticalAlignment="Center" Margin="5" />

<!-- Доп инфа -->

<iconPacks:PackIconUnicons Grid.Row="4" Grid.Column="0" Kind="Pump" Height="20" Width="20" Foreground="{DynamicResource TextSecundaryColor}" VerticalAlignment="Center" Margin="21,0,0,0"/>

<TextBlock Grid.Row="4" Grid.Column="1" Text="{Binding FuelConsumption, ElementName=this, FallbackValue = 3.5 л / 100 км}" FontSize="20" TextAlignment="Center" Foreground="{DynamicResource TextSecundaryColor}" HorizontalAlignment="Left" VerticalAlignment="Center" Margin="5" />

</Grid>

</Grid>

</Border>

</UserControl>

using System;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Input;

using System.Windows.Media;

namespace OneWay.Controls

{

/// <summary>

/// Логика взаимодействия для CarItem.xaml

/// </summary>

public partial class CarItem : UserControl

{

public event EventHandler FavoriteButtonClicked;

public event EventHandler DeleteButtonClicked;

public event EventHandler GridClicked;

public CarItem()

{

InitializeComponent();

DataContext = this;

}

private void FavoriteCar\_Click(object sender, RoutedEventArgs e)

{

FavoriteButtonClicked?.Invoke(this, EventArgs.Empty);

}

private void DeleteButtom\_Click(object sender, RoutedEventArgs e)

{

DeleteButtonClicked?.Invoke(this, EventArgs.Empty);

}

private void GridClick(object sender, RoutedEventArgs e)

{

GridClicked?.Invoke(this, EventArgs.Empty);

}

public string Id

{

get { return (string)GetValue(IdProperty); }

set { SetValue(IdProperty, value); }

}

public static readonly DependencyProperty IdProperty =

DependencyProperty.Register("Id", typeof(string), typeof(CarItem));

public string CarName

{

get { return (string)GetValue(CarNameProperty); }

set { SetValue(CarNameProperty, value); }

}

public static readonly DependencyProperty CarNameProperty =

DependencyProperty.Register("CarName", typeof(string), typeof(CarItem));

public string Model

{

get { return (string)GetValue(ModelProperty); }

set { SetValue(ModelProperty, value); }

}

public static readonly DependencyProperty ModelProperty =

DependencyProperty.Register("Model", typeof(string), typeof(CarItem));

public string Volume

{

get { return (string)GetValue(VolumeProperty); }

set { SetValue(VolumeProperty, value); }

}

public static readonly DependencyProperty VolumeProperty =

DependencyProperty.Register("Volume", typeof(string), typeof(CarItem));

public string Transmission

{

get { return (string)GetValue(TransmissionProperty); }

set { SetValue(TransmissionProperty, value); }

}

public static readonly DependencyProperty TransmissionProperty =

DependencyProperty.Register("Transmission", typeof(string), typeof(CarItem));

public string Body

{

get { return (string)GetValue(BodyProperty); }

set { SetValue(BodyProperty, value); }

}

public static readonly DependencyProperty BodyProperty =

DependencyProperty.Register("Body", typeof(string), typeof(CarItem));

public string Seat

{

get { return (string)GetValue(SeatProperty); }

set { SetValue(SeatProperty, value); }

}

public static readonly DependencyProperty SeatProperty =

DependencyProperty.Register("Seat", typeof(string), typeof(CarItem));

public string FuelConsumption

{

get { return (string)GetValue(FuelConsumptionProperty); }

set { SetValue(FuelConsumptionProperty, value); }

}

public static readonly DependencyProperty FuelConsumptionProperty =

DependencyProperty.Register("FuelConsumption", typeof(string), typeof(CarItem));

public string FavoriteCar

{

get { return (string)GetValue(FavoriteCarProperty); }

set { SetValue(FavoriteCarProperty, value); }

}

public static readonly DependencyProperty FavoriteCarProperty =

DependencyProperty.Register("FavoriteCar", typeof(string), typeof(CarItem), new PropertyMetadata(OnFavoriteCarChanged));

private static void OnFavoriteCarChanged(DependencyObject d, DependencyPropertyChangedEventArgs e)

{

CarItem carItem = (CarItem)d;

string newValue = (string)e.NewValue;

if (newValue == "1")

{

carItem.FavoriteIcon.Foreground = new SolidColorBrush(Color.FromRgb(255, 215, 0));

}

else

{

carItem.FavoriteIcon.Foreground = new SolidColorBrush(Color.FromRgb(176, 183, 195));

}

}

}

}

<Window x:Class="OneWay.Controls.CustomMessageBox"

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:OneWay.Controls"

xmlns:iconPacks="http://metro.mahapps.com/winfx/xaml/iconpacks"

mc:Ignorable="d"

Title="WpfMessageBox" MinHeight="160"

MinWidth="420" MaxHeight="750" MaxWidth="420"

Background="Transparent"

SizeToContent="WidthAndHeight"

WindowStartupLocation="CenterScreen"

ShowInTaskbar="False" ResizeMode="NoResize"

WindowStyle="None" Topmost="True">

<Window.Resources>

<Style TargetType="Button" x:Key="MessageBoxButtonStyle">

<Setter Property="Background" Value="Transparent" />

<Setter Property="TextBlock.TextAlignment" Value="Center" />

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="Button">

<Border Name="Border" CornerRadius="5"

BorderBrush="#000" BorderThickness="1,1,1,1"

Background="{TemplateBinding Background}">

<ContentPresenter x:Name="contentPresenter"

ContentTemplate="{TemplateBinding ContentTemplate}"

Content="{TemplateBinding Content}"

HorizontalAlignment="{TemplateBinding HorizontalAlignment}"

Margin="{TemplateBinding Padding}"

VerticalAlignment="{TemplateBinding VerticalAlignment}" />

</Border>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

</Window.Resources>

<Border BorderBrush="LightSlateGray" BorderThickness="0" CornerRadius="0">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="Auto"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="Auto"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<Rectangle Grid.Row="1" Grid.Column="0" Grid.RowSpan="2" Grid.ColumnSpan="2" Fill="#FF131725"/>

<Rectangle Grid.Row="0" Grid.Column="0" Grid.ColumnSpan="2" Fill="#1d2335"/>

<Grid Grid.Row="0" Grid.ColumnSpan="2"

MinHeight="40" >

<TextBlock Margin="10,1,0,1" Name="MessageTitle"

FontWeight="Bold" TextTrimming="CharacterEllipsis"

LineHeight="22" FontSize="16"

VerticalAlignment="Center" Foreground="White"/>

</Grid>

<iconPacks:PackIconUnicons Name="img" Margin="10 5 5 5" Grid.Row="1"

Grid.Column="0" Width="50"

Height="50" Foreground="#FFB0B7C3" VerticalAlignment="Center"/>

<TextBlock Margin="10,5,10,5" VerticalAlignment="Center" TextWrapping="Wrap" Name="txtMsg" Grid.Row="1" Foreground="#B0B7C3" FontSize="20"

Grid.Column="1"

LineHeight="20" />

<Grid Grid.Row="2" Grid.ColumnSpan="2"

Grid.Column="0" >

<StackPanel Orientation="Horizontal"

HorizontalAlignment="Right" >

<Button Name="btnOk" Content="ОК"

Margin="3,5" MinWidth="70" Height="35"

Click="Button\_Click" Foreground="Black"

FontSize="14"

Style="{StaticResource MessageBoxButtonStyle}"

Background="#b6dbd6" VerticalAlignment="Center"

HorizontalAlignment="Stretch"

VerticalContentAlignment="Center"

HorizontalContentAlignment="Center" />

<Button Name="btnYes" Content="Да"

Margin="3,5" MinWidth="70" Height="35"

Click="Button\_Click" Foreground="Black" FontSize="14"

Style="{StaticResource MessageBoxButtonStyle}"

Background="#b6dbd6" VerticalAlignment="Center"

HorizontalAlignment="Stretch"

VerticalContentAlignment="Center"

HorizontalContentAlignment="Center"/>

<Button Name="btnNo" Content="Нет"

Margin="3,5" MinWidth="70" Height="35"

Click="Button\_Click" Foreground="Black"

FontSize="14"

Style="{StaticResource MessageBoxButtonStyle}"

Background="#dbb6b6" VerticalAlignment="Center"

HorizontalAlignment="Stretch"

VerticalContentAlignment="Center"

HorizontalContentAlignment="Center" />

<Button Name="btnCancel" Margin="3,5"

Content="Отмена" MinWidth="70"

Height="35" Click="Button\_Click"

Style="{StaticResource MessageBoxButtonStyle}"

Foreground="Black"

Background="#dbb6b6" FontSize="14"

VerticalAlignment="Center"

HorizontalAlignment="Stretch"

VerticalContentAlignment="Center"

HorizontalContentAlignment="Center"/>

</StackPanel>

</Grid>

</Grid>

</Border>

</Window>

using MahApps.Metro.IconPacks;

using System.Windows;

namespace OneWay.Controls

{

/// <summary>

/// Логика взаимодействия для CustomMessageBox.xaml

/// </summary>

public partial class CustomMessageBox : Window

{

public enum MessageBoxType

{

ConfirmationWithYesNo = 0,

ConfirmationWithYesNoCancel,

Information,

Error,

Warning

}

public enum MessageBoxImage

{

Warning = 0,

Question,

Information,

Error,

None

}

public CustomMessageBox()

{

InitializeComponent();

}

static CustomMessageBox \_messageBox;

static MessageBoxResult \_result = MessageBoxResult.No;

public static MessageBoxResult Show

(string caption, string msg, MessageBoxType type)

{

switch (type)

{

case MessageBoxType.ConfirmationWithYesNo:

return Show(caption, msg, MessageBoxButton.YesNo,

MessageBoxImage.Question);

case MessageBoxType.ConfirmationWithYesNoCancel:

return Show(caption, msg, MessageBoxButton.YesNoCancel,

MessageBoxImage.Question);

case MessageBoxType.Information:

return Show(caption, msg, MessageBoxButton.OK,

MessageBoxImage.Information);

case MessageBoxType.Error:

return Show(caption, msg, MessageBoxButton.OK,

MessageBoxImage.Error);

case MessageBoxType.Warning:

return Show(caption, msg, MessageBoxButton.OK,

MessageBoxImage.Warning);

default:

return MessageBoxResult.No;

}

}

public static MessageBoxResult Show(string msg, MessageBoxType type)

{

return Show(string.Empty, msg, type);

}

public static MessageBoxResult Show(string msg)

{

return Show(string.Empty, msg,

MessageBoxButton.OK, MessageBoxImage.None);

}

public static MessageBoxResult Show

(string caption, string text)

{

return Show(caption, text,

MessageBoxButton.OK, MessageBoxImage.None);

}

public static MessageBoxResult Show

(string caption, string text, MessageBoxButton button)

{

return Show(caption, text, button,

MessageBoxImage.None);

}

public static MessageBoxResult Show

(string caption, string text,

MessageBoxButton button, MessageBoxImage image)

{

\_messageBox = new CustomMessageBox

{ txtMsg = { Text = text }, MessageTitle = { Text = caption } };

SetVisibilityOfButtons(button);

SetImageOfMessageBox(image);

\_messageBox.ShowDialog();

return \_result;

}

private static void SetVisibilityOfButtons(MessageBoxButton button)

{

switch (button)

{

case MessageBoxButton.OK:

\_messageBox.btnCancel.Visibility = Visibility.Collapsed;

\_messageBox.btnNo.Visibility = Visibility.Collapsed;

\_messageBox.btnYes.Visibility = Visibility.Collapsed;

\_messageBox.btnOk.Focus();

break;

case MessageBoxButton.OKCancel:

\_messageBox.btnNo.Visibility = Visibility.Collapsed;

\_messageBox.btnYes.Visibility = Visibility.Collapsed;

\_messageBox.btnCancel.Focus();

break;

case MessageBoxButton.YesNo:

\_messageBox.btnOk.Visibility = Visibility.Collapsed;

\_messageBox.btnCancel.Visibility = Visibility.Collapsed;

\_messageBox.btnNo.Focus();

break;

case MessageBoxButton.YesNoCancel:

\_messageBox.btnOk.Visibility = Visibility.Collapsed;

\_messageBox.btnCancel.Focus();

break;

default:

break;

}

}

private static void SetImageOfMessageBox(MessageBoxImage image)

{

switch (image)

{

case MessageBoxImage.Warning:

\_messageBox.img.Kind = PackIconUniconsKind.ExclamationTriangle;

break;

case MessageBoxImage.Question:

\_messageBox.img.Kind = PackIconUniconsKind.QuestionCircle;

break;

case MessageBoxImage.Information:

\_messageBox.img.Kind = PackIconUniconsKind.InfoCircle;

break;

case MessageBoxImage.Error:

\_messageBox.img.Kind = PackIconUniconsKind.TimesCircle;

break;

default:

\_messageBox.img.Visibility = Visibility.Collapsed;

break;

}

}

private void Button\_Click(object sender, RoutedEventArgs e)

{

if (sender == btnOk)

\_result = MessageBoxResult.OK;

else if (sender == btnYes)

\_result = MessageBoxResult.Yes;

else if (sender == btnNo)

\_result = MessageBoxResult.No;

else if (sender == btnCancel)

\_result = MessageBoxResult.Cancel;

else

\_result = MessageBoxResult.None;

\_messageBox.Close();

\_messageBox = null;

}

}

}

<UserControl x:Class="OneWay.Controls.FuelItem"

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:OneWay.Controls"

xmlns:lvc="clr-namespace:LiveCharts.Wpf;assembly=LiveCharts.Wpf"

mc:Ignorable="d" Name="this" Width="auto" Loaded="this\_Loaded">

<Border Height="180" Width="280" BorderThickness="2" BorderBrush="{DynamicResource TextSecundaryColor}" Margin="10" CornerRadius="10" Background="Transparent" MouseLeftButtonDown="GridClick">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<TextBlock Text="{Binding OctanNumber, ElementName=this, FallbackValue = АИ-95}" Grid.Row="0" FontSize ="25" Foreground="{DynamicResource TextSecundaryColor}" Margin="15,15,105,27"/>

<TextBlock Text="{Binding FuelType, ElementName=this, FallbackValue = Бензин}" Grid.Row="0" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="15,38,10,5"/>

<TextBlock Text="{Binding PriceRub, ElementName=this, FallbackValue = 3.25руб}" TextAlignment="Right" Grid.Row="0" FontSize ="25" Foreground="{DynamicResource TextSecundaryColor}" Margin="140,15,10,27"/>

<lvc:CartesianChart Name="liveChart" Margin="10,10,10,20" Grid.Row="1" Zoom="None">

<lvc:CartesianChart.AxisX>

<lvc:Axis ShowLabels="false">

<lvc:Axis.Labels>

<Binding Path="TripDates"/>

</lvc:Axis.Labels>

<lvc:Axis.Separator>

<lvc:Separator StrokeThickness="0"></lvc:Separator>

</lvc:Axis.Separator>

</lvc:Axis>

</lvc:CartesianChart.AxisX>

<lvc:CartesianChart.AxisY>

<lvc:Axis ShowLabels="False" MinValue="0">

<lvc:Axis.Separator>

<lvc:Separator StrokeThickness="0"></lvc:Separator>

</lvc:Axis.Separator>

</lvc:Axis>

</lvc:CartesianChart.AxisY>

</lvc:CartesianChart>

</Grid>

</Border>

</UserControl>

using LiveCharts;

using LiveCharts.Wpf;

using OneWay.Class;

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.Linq;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Media;

namespace OneWay.Controls

{

public partial class FuelItem : UserControl

{

public event EventHandler GridClicked;

public FuelItem()

{

InitializeComponent();

}

private void GridClick(object sender, RoutedEventArgs e)

{

GridClicked?.Invoke(this, EventArgs.Empty);

}

public ObservableCollection<string> TripDates { get; set; }

public static readonly DependencyProperty OctanNumberProperty =

DependencyProperty.Register("OctanNumber", typeof(string), typeof(FuelItem));

public string OctanNumber

{

get { return (string)GetValue(OctanNumberProperty); }

set { SetValue(OctanNumberProperty, value); }

}

public static readonly DependencyProperty FuelTypeProperty =

DependencyProperty.Register("FuelType", typeof(string), typeof(FuelItem));

public string FuelType

{

get { return (string)GetValue(FuelTypeProperty); }

set { SetValue(FuelTypeProperty, value); }

}

public static readonly DependencyProperty PriceRubProperty =

DependencyProperty.Register("PriceRub", typeof(string), typeof(FuelItem));

public string PriceRub

{

get { return (string)GetValue(PriceRubProperty); }

set { SetValue(PriceRubProperty, value); }

}

public static readonly DependencyProperty IdUserProperty =

DependencyProperty.Register("IdUser", typeof(string), typeof(FuelItem));

public string IdUser

{

get { return (string)GetValue(IdUserProperty); }

set { SetValue(IdUserProperty, value); }

}

private void this\_Loaded(object sender, RoutedEventArgs e)

{

DataBase db = new DataBase("OneWayDB.db");

List<(double price, DateTime Date)> fuelPriceHistory = db.GetFuelPriceHistory(db.GetFuelId(OctanNumber),Convert.ToInt32(IdUser));

var seriesNew = new LineSeries

{

Title = "Цена",

Values = new ChartValues<double>(fuelPriceHistory.Select(item => item.price)),

StrokeThickness = 2,

Fill = Brushes.Transparent,

LineSmoothness = 0

};

TripDates = new ObservableCollection<string>();

foreach (var priceEntry in fuelPriceHistory)

{

TripDates.Add(priceEntry.Date.ToString("dd.MM.yyyy"));

}

TripDates.Add(fuelPriceHistory.Last().Date.ToString("dd.MM.yyyy"));

DataContext = this;

liveChart.Series = new SeriesCollection { seriesNew };

}

}

}

<UserControl x:Class="OneWay.Controls.MenuItem"

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:OneWay.Controls"

mc:Ignorable="d" Name="this">

<UserControl.Resources>

<Style x:Key="IndicatorStyle" TargetType="{x:Type Border}">

<Setter Property="CornerRadius" Value="0 5 5 0"/>

<Setter Property="Background" Value="{DynamicResource PrimaryBlueColor}"/>

<Setter Property="Visibility" Value="Hidden"/>

<Setter Property="Height" Value="35"/>

<Setter Property="Width" Value="4"/>

<Setter Property="HorizontalAlignment" Value="Left"/>

<Setter Property="VerticalAlignment" Value="Center"/>

<Style.Triggers>

<DataTrigger Binding="{Binding IsMouseOver, RelativeSource={RelativeSource Mode=FindAncestor, AncestorType=ToggleButton}}" Value="True">

<Setter Property="Visibility" Value="Visible"/>

<Setter Property="Background" Value="{DynamicResource TextFourthColor}"/>

</DataTrigger>

<DataTrigger Binding="{Binding IsChecked, RelativeSource={RelativeSource Mode=FindAncestor, AncestorType=ToggleButton}}" Value="True">

<Setter Property="Visibility" Value="Visible"/>

<Setter Property="Background" Value="{DynamicResource PrimaryBlueColor}"/>

</DataTrigger>

</Style.Triggers>

</Style>

<Style x:Key="MenuIconStyle" TargetType="{x:Type Path}">

<Setter Property="Fill" Value="{DynamicResource SecundaryIconColor}"/>

<Style.Triggers>

<DataTrigger Binding="{Binding IsMouseOver, RelativeSource={RelativeSource Mode=FindAncestor, AncestorType=ToggleButton}}" Value="True">

<Setter Property="Fill" Value="{DynamicResource TextFourthColor}"/>

</DataTrigger>

<DataTrigger Binding="{Binding IsChecked, RelativeSource={RelativeSource Mode=FindAncestor, AncestorType=ToggleButton}}" Value="True">

<Setter Property="Fill" Value="{DynamicResource PrimaryBlueColor}"/>

</DataTrigger>

</Style.Triggers>

</Style>

<Style x:Key="MenuTextStyle" TargetType="{x:Type TextBlock}">

<Setter Property="Foreground" Value="{DynamicResource SecundaryIconColor}"/>

<Setter Property="VerticalAlignment" Value="Center"/>

<Setter Property="FontSize" Value="12"/>

<Setter Property="FontWeight" Value="Regular"/>

<Setter Property="Margin" Value="18 0 0 0"/>

<Style.Triggers>

<DataTrigger Binding="{Binding IsMouseOver, RelativeSource={RelativeSource Mode=FindAncestor, AncestorType=ToggleButton}}" Value="True">

<Setter Property="Foreground" Value="{DynamicResource TextFourthColor}"/>

</DataTrigger>

<DataTrigger Binding="{Binding IsChecked, RelativeSource={RelativeSource Mode=FindAncestor, AncestorType=ToggleButton}}" Value="True">

<Setter Property="Foreground" Value="{DynamicResource PrimaryBlueColor}"/>

</DataTrigger>

</Style.Triggers>

</Style>

<Style x:Key="MenuButtonStyle" TargetType="{x:Type ToggleButton}">

<Setter Property="Background" Value="Transparent"/>

<Setter Property="BorderThickness" Value="{DynamicResource BorderThickness}"/>

<Setter Property="HorizontalContentAlignment" Value="Center"/>

<Setter Property="Width" Value="250"/>

<Setter Property="Height" Value="40"/>

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="{x:Type ToggleButton}">

<Border Background="{TemplateBinding Background}" BorderThickness="{TemplateBinding BorderThickness}" Padding="{Binding Padding, ElementName=this}">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="60"/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<Border Style="{StaticResource IndicatorStyle}"/>

<!-- иконка -->

<Path Data="{Binding Icon, ElementName=this, FallbackValue={StaticResource DefaultIcon}, TargetNullValue={StaticResource DefaultIcon}}" Margin="{Binding IconMargin, FallbackValue='0,0,0,0', TargetNullValue='20,0,0,0'}"

Stretch="Uniform" Width="{Binding IconWidth, ElementName=this, FallbackValue=20,TargetNullValue=20}" Style="{StaticResource MenuIconStyle}"/>

<!-- тест меню -->

<TextBlock Style="{StaticResource MenuTextStyle}" FontWeight="Regular" FontSize="14" Grid.Column="1" Text="{Binding Text, ElementName=this, FallbackValue=MenuText, TargetNullValue=MenuText}" Margin="0,0,0,0"/>

</Grid>

</Border>

<ControlTemplate.Triggers>

<DataTrigger Binding="{Binding IsSelected, ElementName=this}" Value="True">

<Setter Property="IsChecked" Value="True"/>

</DataTrigger>

</ControlTemplate.Triggers>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

</UserControl.Resources>

<Grid>

<RadioButton Style="{StaticResource MenuButtonStyle}" GroupName="{Binding GroupName, ElementName=this}"/>

</Grid>

</UserControl>

using System.Windows;

using System.Windows.Controls;

using System.Windows.Media;

namespace OneWay.Controls

{

/// <summary>

/// Логика взаимодействия для MenuItem.xaml

/// </summary>

public partial class MenuItem : UserControl

{

public MenuItem()

{

InitializeComponent();

}

public PathGeometry Icon

{

get { return (PathGeometry)GetValue(IconProperty); }

set { SetValue(IconProperty, value); }

}

public static readonly DependencyProperty IconProperty =

DependencyProperty.Register("Icon", typeof(PathGeometry), typeof(MenuItem));

public int IconWidth

{

get { return (int)GetValue(IconWidthProperty); }

set { SetValue(IconWidthProperty, value); }

}

public static readonly DependencyProperty IconWidthProperty =

DependencyProperty.Register("IconWidth", typeof(int), typeof(MenuItem));

public SolidColorBrush IndicatorBrush

{

get { return (SolidColorBrush)GetValue(IndicatorBrushProperty); }

set { SetValue(IndicatorBrushProperty, value); }

}

public static readonly DependencyProperty IndicatorBrushProperty =

DependencyProperty.Register("IndicatorBrush", typeof(SolidColorBrush), typeof(MenuItem));

public int IndicatorIndicatorCornerRadius

{

get { return (int)GetValue(IndicatorIndicatorCornerRadiusProperty); }

set { SetValue(IndicatorIndicatorCornerRadiusProperty, value); }

}

public static readonly DependencyProperty IndicatorIndicatorCornerRadiusProperty =

DependencyProperty.Register("IndicatorIndicatorCornerRadius", typeof(int), typeof(MenuItem));

public string Text

{

get { return (string)GetValue(TextProperty); }

set { SetValue(TextProperty, value); }

}

public static readonly DependencyProperty TextProperty =

DependencyProperty.Register("Text", typeof(string), typeof(MenuItem));

public new Thickness Padding

{

get { return (Thickness)GetValue(PaddingProperty); }

set { SetValue(PaddingProperty, value); }

}

public static new readonly DependencyProperty PaddingProperty =

DependencyProperty.Register("Padding", typeof(Thickness), typeof(MenuItem));

public bool IsSelected

{

get { return (bool)GetValue(IsSelectedProperty); }

set { SetValue(IsSelectedProperty, value); }

}

public static readonly DependencyProperty IsSelectedProperty =

DependencyProperty.Register("IsSelected", typeof(bool), typeof(MenuItem));

public string GroupName

{

get { return (string)GetValue(GroupNameProperty); }

set { SetValue(GroupNameProperty, value); }

}

public static readonly DependencyProperty GroupNameProperty =

DependencyProperty.Register("GroupName", typeof(string), typeof(MenuItem));

}

}

<UserControl x:Class="OneWay.Controls.RouteItem"

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:OneWay.Controls"

xmlns:iconPacks="http://metro.mahapps.com/winfx/xaml/iconpacks"

Name="this" mc:Ignorable="d" Loaded="Route\_Loaded">

<Border BorderBrush="{DynamicResource TextSecundaryColor}" BorderThickness="1" CornerRadius="5" Height="100" Margin="10">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="174"/>

</Grid.ColumnDefinitions>

<Grid Grid.Column="1">

<Grid.RowDefinitions>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<Button x:Name="InfoButton" Grid.Column="0" Height="30" Width="160" VerticalAlignment="Center" HorizontalAlignment="Center" Click="InfoButton\_Click">

<Button.Template>

<ControlTemplate TargetType="Button">

<Grid>

<Border Background="Transparent" CornerRadius="10" BorderThickness="1" BorderBrush="{DynamicResource TextSecundaryColor}">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

<TextBlock Text="Информация" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" VerticalAlignment="Center" HorizontalAlignment="Center"/>

</Grid>

</ControlTemplate>

</Button.Template>

</Button>

<Grid Grid.Row="1">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<Button x:Name="DeleteButton" Grid.Column="0" Height="30" Width="160" VerticalAlignment="Center" HorizontalAlignment="Center" Click="DeleteButton\_Click">

<Button.Template>

<ControlTemplate TargetType="Button">

<Grid>

<Border Background="Transparent" CornerRadius="10" BorderThickness="1" BorderBrush="{DynamicResource TextSecundaryColor}">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

<TextBlock Text="Удалить" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" VerticalAlignment="Center" HorizontalAlignment="Center"/>

</Grid>

</ControlTemplate>

</Button.Template>

</Button>

</Grid>

</Grid>

<Grid Grid.Column="0">

<Grid.RowDefinitions>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<Grid Grid.Column="0">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="2\*"/>

</Grid.ColumnDefinitions>

<StackPanel Orientation="Horizontal" Grid.Column="0">

<TextBlock Text="{Binding TripName, ElementName=this, FallbackValue = Минск-Брест}" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 10 0 10"/>

<iconPacks:PackIconUnicons x:Name="PointIcon" Kind="MapMarkerInfo" VerticalAlignment="Center" HorizontalAlignment="Center" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" Margin="10 0 0 0" Visibility="Visible">

<iconPacks:PackIconUnicons.ToolTip>

<ToolTip Background="Transparent" BorderThickness="0" Placement="Bottom" HorizontalOffset="90">

<Border Height="auto" Width="auto" CornerRadius="5" Background="{DynamicResource PrimaryBlueColor}">

<TextBlock Text="{Binding Point}" x:Name="ToolTipTextBlock" Foreground="White" FontSize="14" FontWeight="Regular" HorizontalAlignment="Center" VerticalAlignment="Center" Margin="10 5 10 5" TextWrapping="Wrap"/>

</Border>

</ToolTip>

</iconPacks:PackIconUnicons.ToolTip>

</iconPacks:PackIconUnicons>

<Button x:Name="FavoriteIcon" Height="16" Width="16" Background="Transparent" Foreground="{DynamicResource TextSecundaryColor}" BorderThickness="0" HorizontalAlignment="Right" Margin="10,0,0,0" Click="FavoriteIcon\_Click">

<Button.Template>

<ControlTemplate TargetType="Button">

<Grid Background="Transparent">

<!-- Добавлен прозрачный фон -->

<iconPacks:PackIconUnicons Kind="Favorite" HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Grid>

</ControlTemplate>

</Button.Template>

</Button>

</StackPanel>

</Grid>

<Grid Grid.Row="1">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<Grid Grid.Column="0">

<Grid.RowDefinitions>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<TextBlock Text="Расстояние" Grid.Row="0" FontSize ="15" Foreground="{DynamicResource TextSecundaryColor}" VerticalAlignment="Bottom" Margin="10 0 0 0"/>

<TextBlock Text="{Binding Distance, ElementName=this, FallbackValue = 220 км}" Grid.Row="1" FontSize ="15" Foreground="{DynamicResource TextSecundaryColor}" VerticalAlignment="Top" Margin="10 0 0 0"/>

</Grid>

<Grid Grid.Column="1">

<Grid.RowDefinitions>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<TextBlock Text="Время в пути" Grid.Row="0" FontSize ="15" Foreground="{DynamicResource TextSecundaryColor}" VerticalAlignment="Bottom" Margin="10 0 0 0"/>

<TextBlock Text="{Binding Time, ElementName=this, FallbackValue = 2.30 ч}" Grid.Row="1" FontSize ="15" Foreground="{DynamicResource TextSecundaryColor}" VerticalAlignment="Top" Margin="10 0 0 0"/>

</Grid>

</Grid>

</Grid>

</Grid>

</Border>

</UserControl>

using System;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Media;

namespace OneWay.Controls

{

public partial class RouteItem : UserControl

{

public event EventHandler InfoButtonClicked;

public event EventHandler DeleteButtonClicked;

public event EventHandler FavoriteIconClicked;

public RouteItem()

{

InitializeComponent();

DataContext = this;

}

private void Route\_Loaded(object sender, RoutedEventArgs e)

{

if (Point.ToString().Length == 0)

{

PointIcon.Visibility = Visibility.Collapsed;

}

}

private void InfoButton\_Click(object sender, RoutedEventArgs e)

{

InfoButtonClicked?.Invoke(this, EventArgs.Empty);

}

private void DeleteButton\_Click(object sender, RoutedEventArgs e)

{

DeleteButtonClicked?.Invoke(this, EventArgs.Empty);

}

public string TripName

{

get { return (string)GetValue(TripNameProperty); }

set { SetValue(TripNameProperty, value); }

}

public static readonly DependencyProperty TripNameProperty =

DependencyProperty.Register("TripName", typeof(string), typeof(RouteItem));

public string Distance

{

get { return (string)GetValue(DistanceProperty); }

set { SetValue(DistanceProperty, value); }

}

public static readonly DependencyProperty DistanceProperty =

DependencyProperty.Register("Distance", typeof(string), typeof(RouteItem));

public string Time

{

get { return (string)GetValue(TimeProperty); }

set { SetValue(TimeProperty, value); }

}

public static readonly DependencyProperty TimeProperty =

DependencyProperty.Register("Time", typeof(string), typeof(RouteItem));

public string Point

{

get { return (string)GetValue(PointProperty); }

set { SetValue(PointProperty, value); }

}

public static readonly DependencyProperty PointProperty =

DependencyProperty.Register("Point", typeof(string), typeof(RouteItem));

public string Id

{

get { return (string)GetValue(IdProperty); }

set { SetValue(IdProperty, value); }

}

public static readonly DependencyProperty IdProperty =

DependencyProperty.Register("Id", typeof(string), typeof(RouteItem));

public string FavoriteRoute

{

get { return (string)GetValue(FavoriteRouteProperty); }

set { SetValue(FavoriteRouteProperty, value); }

}

public static readonly DependencyProperty FavoriteRouteProperty =

DependencyProperty.Register("FavoriteRoute", typeof(string), typeof(RouteItem), new PropertyMetadata(OnFavoriteRouteChanged));

private static void OnFavoriteRouteChanged(DependencyObject d, DependencyPropertyChangedEventArgs e)

{

RouteItem carItem = (RouteItem)d;

string newValue = (string)e.NewValue;

if (newValue == "1")

{

carItem.FavoriteIcon.Foreground = new SolidColorBrush(Color.FromRgb(255, 215, 0));

}

else

{

carItem.FavoriteIcon.Foreground = new SolidColorBrush(Color.FromRgb(176, 183, 195));

}

}

private void FavoriteIcon\_Click(object sender, RoutedEventArgs e)

{

FavoriteIconClicked?.Invoke(this, EventArgs.Empty);

}

}

}

<UserControl x:Class="OneWay.Controls.ShortCarItem"

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:iconPacks="http://metro.mahapps.com/winfx/xaml/iconpacks"

xmlns:local="clr-namespace:OneWay.Controls"

mc:Ignorable="d" Name="this">

<Border Height="auto" Width="auto" Background="Transparent" BorderThickness="2" BorderBrush="{DynamicResource TextSecundaryColor}" Margin="10" CornerRadius="10">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<StackPanel Orientation="Horizontal" Grid.Row="0" HorizontalAlignment="Left" Margin="10 0 0 0">

<TextBlock Text="{Binding CarName, ElementName=this, FallbackValue = BMW}" FontSize="23" Foreground="{DynamicResource TextSecundaryColor}" Margin="10"/>

</StackPanel>

<Grid Grid.Row="1" Margin="0 0 0 10">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*" />

<ColumnDefinition Width="\*" />

</Grid.ColumnDefinitions>

<Grid Grid.Column="0">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="60"/>

<ColumnDefinition Width="\*" />

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<iconPacks:PackIconMaterial x:Name ="Engine" Grid.Row="0" Grid.Column="0" Kind="Engine" Height="20" Width="20" Foreground="{DynamicResource TextSecundaryColor}" HorizontalAlignment="Center" VerticalAlignment="Center"/>

<TextBlock Grid.Row="0" Grid.Column="1" Text="{Binding Volume, ElementName=this, FallbackValue = 2.4 л}" FontSize="15" TextAlignment="Center" Foreground="{DynamicResource TextSecundaryColor}" HorizontalAlignment="Center" VerticalAlignment="Center"/>

<!-- Подсказка -->

<Border Grid.Row="0" Grid.Column="0" Width="auto" Height="auto" Background="Transparent" >

<Border.ToolTip>

<ToolTip Background="Transparent" BorderThickness="0" Placement="Bottom" HorizontalOffset="90">

<Border Height="30" Width="auto" CornerRadius="5" Background="{DynamicResource PrimaryBlueColor}">

<TextBlock Text="Объем двигателя и тип топлива" Foreground="White" FontSize="14" FontWeight="Regular" HorizontalAlignment="Center" VerticalAlignment="Center" Margin="10 5 10 5"/>

</Border>

</ToolTip>

</Border.ToolTip>

</Border>

<iconPacks:PackIconMaterial Grid.Row="1" Grid.Column="0" Kind="CarShiftPattern" Height="20" Width="20" Foreground="{DynamicResource TextSecundaryColor}" HorizontalAlignment="Center" VerticalAlignment="Center" />

<TextBlock Grid.Row="1" Grid.Column="1" Text="{Binding Transmission, ElementName=this, FallbackValue = Механика}" FontSize="15" TextAlignment="Center" Foreground="{DynamicResource TextSecundaryColor}" HorizontalAlignment="Center" VerticalAlignment="Center"/>

<!-- Подсказка -->

<Border Grid.Row="1" Grid.Column="0" Width="auto" Height="auto" Background="Transparent" >

<Border.ToolTip>

<ToolTip Background="Transparent" BorderThickness="0" Placement="Bottom" HorizontalOffset="40">

<Border Height="30" Width="auto" CornerRadius="5" Background="{DynamicResource PrimaryBlueColor}">

<TextBlock Text="Коробка передач" Foreground="White" FontSize="14" FontWeight="Regular" HorizontalAlignment="Center" VerticalAlignment="Center" Margin="10 5 10 5"/>

</Border>

</ToolTip>

</Border.ToolTip>

</Border>

</Grid>

<Grid Grid.Column="1">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="60" />

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<iconPacks:PackIconMaterial Grid.Row="0" Grid.Column="0" Kind="Car" Height="20" Width="20" Foreground="{DynamicResource TextSecundaryColor}" HorizontalAlignment="Center" VerticalAlignment="Center"/>

<Border Grid.Row="0" Grid.Column="0" Width="auto" Height="auto" Background="Transparent" >

<Border.ToolTip>

<ToolTip Background="Transparent" BorderThickness="0" Placement="Bottom" HorizontalOffset="20">

<Border Height="30" Width="auto" CornerRadius="5" Background="{DynamicResource PrimaryBlueColor}">

<TextBlock Text="Тип кузова" Foreground="White" FontSize="14" FontWeight="Regular" HorizontalAlignment="Center" VerticalAlignment="Center" Margin="10 5 10 5"/>

</Border>

</ToolTip>

</Border.ToolTip>

</Border>

<TextBlock Grid.Row="0" Grid.Column="1" Text="{Binding Body, ElementName=this, FallbackValue = Седан}" FontSize="15" TextAlignment="Center" Foreground="{DynamicResource TextSecundaryColor}" HorizontalAlignment="Center" VerticalAlignment="Center" />

<iconPacks:PackIconMaterial Grid.Row="1" Grid.Column="0" Kind="CarSeat" Height="20" Width="20" Foreground="{DynamicResource TextSecundaryColor}" HorizontalAlignment="Center" VerticalAlignment="Center"/>

<Border Grid.Row="1" Grid.Column="0" Width="auto" Height="auto" Background="Transparent" >

<Border.ToolTip>

<ToolTip Background="Transparent" BorderThickness="0" Placement="Bottom" HorizontalOffset="70">

<Border Height="30" Width="auto" CornerRadius="5" Background="{DynamicResource PrimaryBlueColor}">

<TextBlock Text="Число мест в автомобиле" Foreground="White" FontSize="14" FontWeight="Regular" HorizontalAlignment="Center" VerticalAlignment="Center" Margin="10 5 10 5"/>

</Border>

</ToolTip>

</Border.ToolTip>

</Border>

<TextBlock Grid.Row="1" Grid.Column="1" Text="{Binding Seat, ElementName=this, FallbackValue = 6}" FontSize="15" TextAlignment="Center" Foreground="{DynamicResource TextSecundaryColor}" HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Grid>

</Grid>

</Grid>

</Border>

</UserControl>

using System.Windows;

using System.Windows.Controls;

using System.Windows.Controls.Primitives;

using System.Windows.Input;

namespace OneWay.Controls

{

public partial class ShortCarItem : UserControl

{

public ShortCarItem()

{

InitializeComponent();

}

public string CarName

{

get { return (string)GetValue(CarNameProperty); }

set { SetValue(CarNameProperty, value); }

}

public static readonly DependencyProperty CarNameProperty =

DependencyProperty.Register("CarName", typeof(string), typeof(ShortCarItem));

public string Volume

{

get { return (string)GetValue(VolumeProperty); }

set { SetValue(VolumeProperty, value); }

}

public static readonly DependencyProperty VolumeProperty =

DependencyProperty.Register("Volume", typeof(string), typeof(ShortCarItem));

public string Body

{

get { return (string)GetValue(BodyProperty); }

set { SetValue(BodyProperty, value); }

}

public static readonly DependencyProperty BodyProperty =

DependencyProperty.Register("Body", typeof(string), typeof(ShortCarItem));

public string Transmission

{

get { return (string)GetValue(TransmissionProperty); }

set { SetValue(TransmissionProperty, value); }

}

public static readonly DependencyProperty TransmissionProperty =

DependencyProperty.Register("Transmission", typeof(string), typeof(ShortCarItem));

public string Seat

{

get { return (string)GetValue(SeatProperty); }

set { SetValue(SeatProperty, value); }

}

public static readonly DependencyProperty SeatProperty =

DependencyProperty.Register("Seat", typeof(string), typeof(ShortCarItem));

}

}

<UserControl x:Class="OneWay.Controls.ShortTripItem"

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:OneWay.Controls"

mc:Ignorable="d" Name="this">

<Border Height="100" Width="auto" BorderThickness="2" BorderBrush="{DynamicResource TextSecundaryColor}" Margin="10" CornerRadius="10" Background="Transparent">

<Grid>

<Grid.RowDefinitions>

<RowDefinition/>

<RowDefinition/>

</Grid.RowDefinitions>

<WrapPanel Grid.Row="0" HorizontalAlignment="Center">

<TextBlock Text="{Binding Trip, ElementName=this, FallbackValue = Минск - Брест}" FontSize="23" Foreground="{DynamicResource TextSecundaryColor}" Margin="10"/>

</WrapPanel>

<Grid Grid.Row="1">

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<StackPanel Grid.Column="0" HorizontalAlignment="Center">

<TextBlock Text="Дистанция" FontSize="15" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 10 10"/>

<TextBlock Text="{Binding Distanse, ElementName=this, FallbackValue = 152 км}" FontSize="15" TextAlignment="Center" Foreground="{DynamicResource TextSecundaryColor}" Margin=" 10 -5 10 10"/>

</StackPanel>

<StackPanel Grid.Column="1" HorizontalAlignment="Center">

<TextBlock Text="Время в пути" FontSize="15" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 10 10"/>

<TextBlock Text="{Binding Time, ElementName=this, FallbackValue = 12:02 ч}" FontSize="15" TextAlignment="Center" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 -5 10 10"/>

</StackPanel>

<StackPanel Grid.Column="2" HorizontalAlignment="Center">

<TextBlock Text="Автомобиль" FontSize="15" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 10 10"/>

<TextBlock Text="{Binding Car, ElementName=this, FallbackValue = BMW}" FontSize="15" TextAlignment="Center" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 -5 10 10"/>

</StackPanel>

</Grid>

</Grid>

</Border>

</UserControl>

using System.Windows;

using System.Windows.Controls;

namespace OneWay.Controls

{

public partial class ShortTripItem : UserControl

{

public ShortTripItem()

{

InitializeComponent();

}

public string Trip

{

get { return (string)GetValue(TripProperty); }

set { SetValue(TripProperty, value); }

}

public static readonly DependencyProperty TripProperty =

DependencyProperty.Register("Trip", typeof(string), typeof(ShortTripItem));

public string Car

{

get { return (string)GetValue(CarProperty); }

set { SetValue(CarProperty, value); }

}

public static readonly DependencyProperty CarProperty =

DependencyProperty.Register("Car", typeof(string), typeof(ShortTripItem));

public string Distanse

{

get { return (string)GetValue(DistanseProperty); }

set { SetValue(DistanseProperty, value); }

}

public static readonly DependencyProperty DistanseProperty =

DependencyProperty.Register("Distanse", typeof(string), typeof(ShortTripItem));

public string Time

{

get { return (string)GetValue(TimeProperty); }

set { SetValue(TimeProperty, value); }

}

public static readonly DependencyProperty TimeProperty =

DependencyProperty.Register("Time", typeof(string), typeof(ShortTripItem));

}

}

<UserControl x:Class="OneWay.Controls.TripItem"

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:OneWay.Controls"

xmlns:iconPacks="http://metro.mahapps.com/winfx/xaml/iconpacks"

mc:Ignorable="d" x:Name="this" Loaded="TripItem\_Loaded">

<Border BorderBrush="{DynamicResource TextSecundaryColor}" BorderThickness="1" CornerRadius="5" Height="100" Margin="10">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="2\*"/>

<ColumnDefinition Width="174"/>

</Grid.ColumnDefinitions>

<Grid Grid.Column="1">

<Grid.RowDefinitions>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<Button x:Name="PrintButton" Grid.Column="0" Height="30" Width="160" VerticalAlignment="Center" HorizontalAlignment="Center" Click="ReportButton\_Click">

<Button.Template>

<ControlTemplate TargetType="Button">

<Grid>

<Border Background="Transparent" CornerRadius="10" BorderThickness="1" BorderBrush="{DynamicResource TextSecundaryColor}">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

<TextBlock Text="Отчет" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" VerticalAlignment="Center" HorizontalAlignment="Center"/>

</Grid>

</ControlTemplate>

</Button.Template>

</Button>

<Grid Grid.Row="1">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<Button x:Name="InfoButton" Grid.Column="0" BorderThickness="2" Height="30" HorizontalAlignment="Right" VerticalAlignment="Center" Margin="0 -10 0 0" Click="InfoButton\_Click">

<Button.Template>

<ControlTemplate TargetType="Button">

<Grid>

<Border Background="Transparent" CornerRadius="10">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

<iconPacks:PackIconUnicons Kind="InfoCircle" Foreground="{DynamicResource TextSecundaryColor}" VerticalAlignment="Center" />

<TextBlock Text="Информация" FontSize ="10" Foreground="{DynamicResource TextSecundaryColor}" Margin="20 0 0 0" VerticalAlignment="Center"/>

</Grid>

</ControlTemplate>

</Button.Template>

</Button>

<Button x:Name="DeleteButton" Grid.Column="1" BorderThickness="0" Height="30" HorizontalAlignment="Center" VerticalAlignment="Center" Margin="10 -10 0 0" Click="DeleteButton\_Click">

<Button.Template>

<ControlTemplate TargetType="Button">

<Grid>

<Border Background="Transparent" CornerRadius="10">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

<iconPacks:PackIconUnicons Kind="TrashAlt" Foreground="{DynamicResource TextSecundaryColor}" VerticalAlignment="Center" />

<TextBlock Text="Удалить" FontSize ="10" Foreground="{DynamicResource TextSecundaryColor}" Margin="20 0 0 0" VerticalAlignment="Center"/>

</Grid>

</ControlTemplate>

</Button.Template>

</Button>

</Grid>

</Grid>

<Grid Grid.Column="0">

<Grid.RowDefinitions>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<Grid Grid.Column="0">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="2\*"/>

</Grid.ColumnDefinitions>

<StackPanel Orientation="Horizontal" Grid.Column="0">

<TextBlock Text="{Binding TripName, ElementName=this, FallbackValue = Минск-Брест}" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 10 0 10"/>

<iconPacks:PackIconUnicons x:Name="PointIcon" Kind="MapMarkerInfo" VerticalAlignment="Center" HorizontalAlignment="Center" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" Margin="10 0 0 0" Visibility="Visible">

<iconPacks:PackIconUnicons.ToolTip>

<ToolTip Background="Transparent" BorderThickness="0" Placement="Bottom" HorizontalOffset="90">

<Border Height="auto" Width="auto" CornerRadius="5" Background="{DynamicResource PrimaryBlueColor}">

<TextBlock Text="{Binding Point}" x:Name="ToolTipTextBlock" Foreground="White" FontSize="14" FontWeight="Regular" HorizontalAlignment="Center" VerticalAlignment="Center" Margin="10 5 10 5" TextWrapping="Wrap"/>

<!-- Text="Привет&#x0a;Питербург"-->

</Border>

</ToolTip>

</iconPacks:PackIconUnicons.ToolTip>

</iconPacks:PackIconUnicons>

</StackPanel>

</Grid>

<Grid Grid.Row="1">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<Grid Grid.Column="0">

<Grid.RowDefinitions>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<TextBlock Text="Расстояние" Grid.Row="0" FontSize ="15" Foreground="{DynamicResource TextSecundaryColor}" VerticalAlignment="Bottom" Margin="10 0 0 0"/>

<TextBlock Text="{Binding Distance, ElementName=this, FallbackValue = 220 км}" Grid.Row="1" FontSize ="15" Foreground="{DynamicResource TextSecundaryColor}" VerticalAlignment="Top" Margin="10 0 0 0"/>

</Grid>

<Grid Grid.Column="1">

<Grid.RowDefinitions>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<TextBlock Text="Время в пути" Grid.Row="0" FontSize ="15" Foreground="{DynamicResource TextSecundaryColor}" VerticalAlignment="Bottom" Margin="10 0 0 0"/>

<TextBlock Text="{Binding Time, ElementName=this, FallbackValue = 2.30 ч}" Grid.Row="1" FontSize ="15" Foreground="{DynamicResource TextSecundaryColor}" VerticalAlignment="Top" Margin="10 0 0 0"/>

</Grid>

<Grid Grid.Column="2">

<Grid.RowDefinitions>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<TextBlock Text="Автомобиль" Grid.Row="0" FontSize ="15" Foreground="{DynamicResource TextSecundaryColor}" VerticalAlignment="Bottom" Margin="10 0 0 0"/>

<TextBlock x:Name="CarName" Text="{Binding Car, ElementName=this, FallbackValue = Mercedes}" FontSize ="15" Foreground="{DynamicResource TextSecundaryColor}" VerticalAlignment="Top" Margin="10,24,0,0" HorizontalAlignment="Left" Grid.RowSpan="2" Width="auto">

<TextBlock.ToolTip>

<ToolTip Background="Transparent" BorderThickness="0" Placement="Left" VerticalOffset="-6" HorizontalOffset="10">

<Border Height="Auto" Width="Auto" CornerRadius="5" Background="{DynamicResource PrimaryBlueColor}">

<TextBlock Text="{Binding CarModel}" Foreground="White" FontSize="14" FontWeight="Regular" HorizontalAlignment="Center" VerticalAlignment="Center" Margin="10 5 10 5" TextWrapping="Wrap"/>

</Border>

</ToolTip>

</TextBlock.ToolTip>

</TextBlock>

</Grid>

</Grid>

</Grid>

</Grid>

</Border>

</UserControl>

using System;

using System.Windows;

using System.Windows.Controls;

namespace OneWay.Controls

{

public partial class TripItem : UserControl

{

public event EventHandler ReportButtonClicked;

public event EventHandler InfoButtonClicked;

public event EventHandler DeleteButtonClicked;

public TripItem()

{

InitializeComponent();

DataContext = this;

}

private void TripItem\_Loaded(object sender, RoutedEventArgs e)

{

if (Point.ToString().Length == 0)

{

PointIcon.Visibility = Visibility.Collapsed;

}

}

private void ReportButton\_Click(object sender, RoutedEventArgs e)

{

ReportButtonClicked?.Invoke(this, EventArgs.Empty);

}

private void InfoButton\_Click(object sender, RoutedEventArgs e)

{

InfoButtonClicked?.Invoke(this, EventArgs.Empty);

}

private void DeleteButton\_Click(object sender, RoutedEventArgs e)

{

DeleteButtonClicked?.Invoke(this, EventArgs.Empty);

}

public string TripName

{

get { return (string)GetValue(TripNameProperty); }

set { SetValue(TripNameProperty, value); }

}

public static readonly DependencyProperty TripNameProperty =

DependencyProperty.Register("TripName", typeof(string), typeof(TripItem));

public string CarModel

{

get { return (string)GetValue(CarModelProperty); }

set { SetValue(CarModelProperty, value); }

}

public static readonly DependencyProperty CarModelProperty =

DependencyProperty.Register("CarModel", typeof(string), typeof(TripItem));

public string Distance

{

get { return (string)GetValue(DistanceProperty); }

set { SetValue(DistanceProperty, value); }

}

public static readonly DependencyProperty DistanceProperty =

DependencyProperty.Register("Distance", typeof(string), typeof(TripItem));

public string Time

{

get { return (string)GetValue(TimeProperty); }

set { SetValue(TimeProperty, value); }

}

public static readonly DependencyProperty TimeProperty =

DependencyProperty.Register("Time", typeof(string), typeof(TripItem));

public string Car

{

get { return (string)GetValue(CarProperty); }

set { SetValue(CarProperty, value); }

}

public static readonly DependencyProperty CarProperty =

DependencyProperty.Register("Car", typeof(string), typeof(TripItem));

public string Point

{

get { return (string)GetValue(PointProperty); }

set { SetValue(PointProperty, value); }

}

public static readonly DependencyProperty PointProperty =

DependencyProperty.Register("Point", typeof(string), typeof(TripItem));

public string Id

{

get { return (string)GetValue(IdProperty); }

set { SetValue(IdProperty, value); }

}

public static readonly DependencyProperty IdProperty =

DependencyProperty.Register("Id", typeof(string), typeof(TripItem));

}

}

<UserControl x:Class="OneWay.Controls.UserPopup"

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:OneWay.Controls"

mc:Ignorable="d">

<UserControl.Style>

<Style>

<Style.Triggers>

<EventTrigger RoutedEvent="FrameworkElement.Loaded">

<BeginStoryboard>

<Storyboard>

<DoubleAnimation Storyboard.TargetProperty="Opacity" Duration="00:00:00.25" From="0" To="1"/>

</Storyboard>

</BeginStoryboard>

</EventTrigger>

<Trigger Property="FrameworkElement.Visibility" Value="Collapsed">

<Setter Property="FrameworkElement.Opacity" Value="0"/>

</Trigger>

</Style.Triggers>

</Style>

</UserControl.Style>

<Grid>

<Border x:Name="Conatiner" CornerRadius="5" Width="Auto" Height="30" Margin="10 0 0 0" Background="{DynamicResource PrimaryBlueColor}" HorizontalAlignment="Center" VerticalAlignment="Center">

<TextBlock x:Name="PopupText" FontSize="14" FontWeight="Regular" Panel.ZIndex="1" HorizontalAlignment="Center" VerticalAlignment="Center" Margin="10 5 10 5"/>

</Border>

</Grid>

</UserControl>

using System.Windows.Controls;

namespace OneWay.Controls

{

public partial class UserPopup : UserControl

{

public UserPopup()

{

InitializeComponent();

}

}

}

<Page x:Class="OneWay.Pages.AddCarPage"

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:OneWay.Pages"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800"

Title="AddCarPage">

<Border CornerRadius="5" Background="{DynamicResource SecundaryBackgroundColor}">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="3\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<TextBlock Text="Новый автомобиль" Grid.Column="0" FontSize ="25" Foreground="{DynamicResource TextSecundaryColor}" Margin="10,10,10,403"/>

<ScrollViewer Grid.Column="0" Margin="0 50 0 0">

<StackPanel x:Name="HistoryPanel" Orientation="Vertical">

<Grid x:Name="FilterGrid">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="1\*"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<!-- Первая строка -->

<TextBlock Grid.Row="0" Grid.Column="0" Text="Марка" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0"/>

<TextBox Grid.Row="0" Grid.Column="1" x:Name="CarBrand" Tag="Марка" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="10 0 0 10" Visibility="Visible"/>

<ComboBox Grid.Row="0" Grid.Column="1" x:Name="ComboBoxCarBrand" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="10,0,0,5" BorderBrush="{DynamicResource TextSecundaryColor}" Visibility="Collapsed" SelectionChanged="ComboBoxCarBrand\_SelectionChanged"/>

<!-- Вторая строка -->

<TextBlock Grid.Row="1" Grid.Column="0" Text="Модель" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0"/>

<TextBox Grid.Row="1" Grid.Column="1" Name="CarModel" Tag="Модель" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="10 0 0 10" Visibility="Visible"/>

<ComboBox Grid.Row="1" Grid.Column="1" x:Name="ComboBoxCarModel" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="10,0,0,5" BorderBrush="{DynamicResource TextSecundaryColor}" Visibility="Collapsed" SelectionChanged="ComboBoxCarModel\_SelectionChanged"/>

<!-- Третья строка -->

<TextBlock Grid.Row="2" Grid.Column="0" Text="Поколение" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0"/>

<TextBox Grid.Row="2" Grid.Column="1" x:Name="CarGeneration" Tag="Поколение" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="10 0 0 10" Visibility="Visible"/>

<ComboBox Grid.Row="2" Grid.Column="1" x:Name="ComboBoxCarGeneration" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="10,0,0,5" BorderBrush="{DynamicResource TextSecundaryColor}" Visibility="Collapsed" SelectionChanged="ComboBoxCarGeneration\_SelectionChanged"/>

<!-- Четвертая строка -->

<TextBlock Grid.Row="3" Grid.Column="0" Text="Комплектация" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0"/>

<TextBox Grid.Row="3" Grid.Column="1" x:Name="CarEquipment" Tag="Комплектация" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="10 0 0 10" Visibility="Visible"/>

<ComboBox Grid.Row="3" Grid.Column="1" x:Name="ComboBoxCarEquipment" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="10,0,0,5" BorderBrush="{DynamicResource TextSecundaryColor}" Visibility="Collapsed" SelectionChanged="ComboBoxCarEquipment\_SelectionChanged"/>

<!-- Пятая строка -->

<TextBlock Grid.Row="4" Grid.Column="0" Text="Год выпуска" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0"/>

<TextBox Grid.Row="4" Grid.Column="1" x:Name="Year" Tag="Год выпуска" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="10 0 0 10" IsReadOnly="False" PreviewTextInput="Year\_PreviewTextInput"/>

<!-- Шестая строка -->

<TextBlock Grid.Row="5" Grid.Column="0" Text="Тип кузова" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0"/>

<ComboBox Grid.Row="5" Grid.Column="1" x:Name="Body" FontSize="18" Foreground="{DynamicResource TextSecundaryColor}" Margin="10,0,5,6">

<ComboBoxItem Content="Седан"/>

<ComboBoxItem Content="Внедорожник"/>

<ComboBoxItem Content="Универсал"/>

<ComboBoxItem Content="Хэтчбек"/>

<ComboBoxItem Content="Лифтбек"/>

<ComboBoxItem Content="Минивэн"/>

<ComboBoxItem Content="Купе"/>

<ComboBoxItem Content="Пикап"/>

<ComboBoxItem Content="Кабриолет"/>

</ComboBox>

<!-- Седьмая строка -->

<TextBlock Grid.Row="6" Grid.Column="0" Text="Тип привода" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0"/>

<ComboBox Grid.Row="6" Grid.Column="1" x:Name="Drive" FontSize="18" Margin="10 0 5 6" Foreground="{DynamicResource TextSecundaryColor}">

<ComboBoxItem Content="Передний"/>

<ComboBoxItem Content="Задний"/>

<ComboBoxItem Content="Полный"/>

</ComboBox>

<!-- Восьмая строка -->

<TextBlock Grid.Row="7" Grid.Column="0" Text="Тип трансмиссии" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 10"/>

<ComboBox Grid.Row="7" Grid.Column="1" x:Name="GearBox" FontSize="18" Height="35" Margin="10 0 5 6" Foreground="{DynamicResource TextSecundaryColor}">

<ComboBoxItem Content="АКПП"/>

<ComboBoxItem Content="МКПП"/>

</ComboBox>

<!-- Девятая строка -->

<TextBlock Grid.Row="8" Grid.Column="0" Text="Число мест" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0"/>

<TextBox Grid.Row="8" Grid.Column="1" x:Name="Seats" Tag="Число мест" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="10 0 0 10" PreviewTextInput="Seats\_PreviewTextInput"/>

<!-- Десятая строка -->

<TextBlock Grid.Row="9" Grid.Column="0" Text="Число дверей" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0"/>

<TextBox Grid.Row="9" Grid.Column="1" x:Name="Doors" Tag="Число дверей" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="10 0 0 10" PreviewTextInput="Doord\_PreviewTextInput"/>

<!-- Одиннадцатая строка -->

<TextBlock Grid.Row="10" Grid.Column="0" Text="Тип двигателя" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0"/>

<ComboBox Grid.Row="10" Grid.Column="1" x:Name="EngineType" FontSize="18" Height="35" Margin="10 0 5 6" Foreground="{DynamicResource TextSecundaryColor}" SelectionChanged="EngineType\_SelectionChanged">

<ComboBoxItem Content="ДВС"/>

<ComboBoxItem Content="Электрический"/>

<ComboBoxItem Content="Гибрид"/>

</ComboBox>

<!-- Двенадцатая строка -->

<TextBlock Grid.Row="11" Grid.Column="0" x:Name="TextVolume" Text="Объем двигателя, куб.см" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0" Visibility="Visible"/>

<TextBox Grid.Row="11" Grid.Column="1" x:Name="Volume" Tag="Объем двигателя" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="10 0 0 10" Visibility="Visible" PreviewTextInput="Volume\_PreviewTextInput"/>

<!-- Тринадцатая строка -->

<TextBlock Grid.Row="12" Grid.Column="0" Text="Потребляемое топливо" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0"/>

<ComboBox Grid.Row="12" Grid.Column="1" x:Name="Fuel" FontSize="18" Height="35" Margin="10 0 5 6" Foreground="{DynamicResource TextSecundaryColor}" SelectionChanged="Fuel\_SelectionChanged">

<ComboBoxItem x:Name="Petrol" Content="Бензин" Visibility="Visible"/>

<ComboBoxItem x:Name= "Diesel" Content = "Дизельное топливо" Visibility="Visible"/>

<ComboBoxItem x:Name="Electro" Content="Электричество" Visibility="Visible"/>

<ComboBoxItem x:Name="Gaz" Content="Газ"/>

</ComboBox>

<!-- Четырнадцатая строка -->

<TextBlock Grid.Row="13" Grid.Column="0" x:Name="TextOctaneNumber" Text="Октановое число" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0" Visibility="Visible"/>

<ComboBox Grid.Row="13" Grid.Column="1" x:Name="OctaneNumber" FontSize="18" Height="35" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 5 6" Visibility="Visible">

<ComboBoxItem Content="АИ-95"/>

<ComboBoxItem Content="АИ-98"/>

<ComboBoxItem Content="ДТ"/>

</ComboBox>

<!-- Пятнадцатая строка -->

<TextBlock Grid.Row="14" Grid.Column="0" x:Name="TextFuelConsumption" Text="Расход топлива в смешанном цикле, л/100 км" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 6" TextWrapping="WrapWithOverflow"/>

<TextBox Grid.Row="14" Grid.Column="1" x:Name="FuelConsumption" Tag="Потребление топлива" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="10 0 0 10"/>

<!-- Шеснадацатая строка -->

<TextBlock Grid.Row="15" Grid.Column="0" Text="Максимальная скорость, км/ч" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0"/>

<TextBox Grid.Row="15" Grid.Column="1" x:Name="MaxSpeed" Tag="Максимальная скорость" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="10 0 0 10"/>

<!-- Семнадацатая строка -->

<TextBlock Grid.Row="16" Grid.Column="0" Text="Кондиционер" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 10"/>

<ComboBox Grid.Row="16" Grid.Column="1" x:Name="Conditioner" FontSize="18" Height="35" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 5 6" >

<ComboBoxItem Content="Нет"/>

<ComboBoxItem Content="Есть"/>

</ComboBox>

<!-- Восемнадцатая строка-->

<TextBlock Grid.Row="17" Grid.Column="0" x:Name="TextBlockRangePerCharge" Text="Запас хода на электротяге, км" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0" Visibility="Visible"/>

<TextBox Grid.Row="17" Grid.Column="1" x:Name="RangePerCharge" Tag="Запас хода на электротяге" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="10 0 0 10" Visibility="Visible"/>

<!-- Девятнадцатая строка -->

<TextBlock Grid.Row="18" Grid.Column="0" x:Name="TextBlockBatteryCapacity" Text="Ёмкость батареи, кВт\*ч" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0" Visibility="Visible"/>

<TextBox Grid.Row="18" Grid.Column="1" x:Name="BatteryCapacity" Tag="Объем батареи" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="10 0 0 10" Visibility="Visible"/>

<!-- Двадцатая строка -->

<Button Grid.Row="19" Grid.Column="1" x:Name="AddButton" Content="Добавить автомобиль" FontSize="25" Foreground="{DynamicResource SecundaryBackgroundColor}" Margin="10 5 10 5" BorderThickness="0" Click="Button\_Click">

<Button.Template>

<ControlTemplate TargetType="Button">

<Grid>

<Border x:Name="border" Background="{DynamicResource TextSecundaryColor}" CornerRadius="5">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

</Grid>

<ControlTemplate.Triggers>

<Trigger Property="IsMouseOver" Value="True">

<Setter TargetName="border" Property="Background" Value="#ADD8E6"/>

</Trigger>

</ControlTemplate.Triggers>

</ControlTemplate>

</Button.Template>

</Button>

<Button Grid.Row="19" Grid.Column="0" x:Name="Button\_Web" Content="Посетить сайт Drom.ru" FontSize="25" Foreground="{DynamicResource SecundaryBackgroundColor}" Margin="10 5 10 5" BorderThickness="0" Click="Button\_Web\_Click">

<Button.Template>

<ControlTemplate TargetType="Button">

<Grid>

<Border x:Name="border" Background="{DynamicResource TextSecundaryColor}" CornerRadius="5">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

</Grid>

<ControlTemplate.Triggers>

<Trigger Property="IsMouseOver" Value="True">

<Setter TargetName="border" Property="Background" Value="#ADD8E6"/>

</Trigger>

</ControlTemplate.Triggers>

</ControlTemplate>

</Button.Template>

</Button>

</Grid>

</StackPanel>

</ScrollViewer>

<StackPanel Grid.Column="1">

<TextBlock Text="Настройки" FontSize ="25" Foreground="{DynamicResource TextSecundaryColor}" Margin="10,10,10,0"/>

<TextBlock Text="Заполнение полей" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10,10,10,0"/>

<ComboBox Grid.Row="16" Grid.Column="1" x:Name="Fill" FontSize="18" Height="35" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 5 6" SelectionChanged="Fill\_SelectionChanged" >

<ComboBoxItem Content="Ручное"/>

<ComboBoxItem Content="Автоматическое"/>

</ComboBox>

</StackPanel>

</Grid>

</Border>

</Page>

using OneWay.Class;

using System.Collections.Generic;

using System;

using System.Linq;

using System.Windows;

using System.Windows.Controls;

using HtmlAgilityPack;

using System.Threading.Tasks;

using System.Text.RegularExpressions;

using System.Diagnostics;

using OneWay.Controls;

using System.Net.NetworkInformation;

namespace OneWay.Pages

{

public partial class AddCarPage : Page

{

public int IdUser;

Car newCar = new Car();

public string selectUrl;

public string carNameUrl;

public string carModelUrl;

public string carEquipmentUrl;

DataBase db = new DataBase("OneWayDB.db");

public List<(int id, string octaneNumber)> octaneNumber;

public List<Tuple<string, string>> carName;

public List<Tuple<string, string>> carModel;

public List<Tuple<string, string>> carGeneration;

public List<Tuple<string, string>> carEquipment;

public Dictionary<string, string> CountryTranslations = new Dictionary<string, string>()

{

{ "no", "Для мира" },

{ "russia", "Для России" },

{ "europe", "Для Европы" },

{ "japan", "Для Японии" },

{ "usa", "Для США" },

{ "uae", "Для ОАЭ" },

{ "south-east-asia", "Для Азии" },

{ "south-korea", "Для Южной Кореи" },

{ "china", "Для Китая" }

};

public AddCarPage(int IdUser)

{

InitializeComponent();

Body.SelectedIndex = -1;

Drive.SelectedIndex = -1;

GearBox.SelectedIndex = -1;

EngineType.SelectedIndex = -1;

Fuel.SelectedIndex = -1;

OctaneNumber.SelectedIndex = -1;

Conditioner.SelectedIndex = -1;

Fill.SelectedIndex = 0;

this.IdUser = IdUser;

}

private async void Fill\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

if (Fill.SelectedIndex == 1) // авто

{

ClearField();

newCar = new Car();

bool isConnected = await CheckInternetConnectionAsync();

if (isConnected)

{

BlockField();

ComboBoxCarBrand.Visibility = Visibility.Visible;

CarBrand.Visibility = Visibility.Collapsed;

ComboBoxCarModel.Visibility = Visibility.Visible;

CarModel.Visibility = Visibility.Collapsed;

ComboBoxCarGeneration.Visibility = Visibility.Visible;

CarGeneration.Visibility = Visibility.Collapsed;

ComboBoxCarEquipment.Visibility = Visibility.Visible;

CarEquipment.Visibility = Visibility.Collapsed;

Button\_Web.Visibility = Visibility.Visible;

carName = await GetCarData("https://www.drom.ru/catalog/", "//div[@class='css-1dk948p ehmqafe0']");////

carName = carName.OrderBy(t => t.Item1).ToList();

if(carName.Count != 0)

{

await Dispatcher.InvokeAsync(() =>

{

foreach (Tuple<string, string> car in carName)

{

ComboBoxCarBrand.Items.Add(car.Item1);

}

});

}

else

{

Fill.SelectedIndex = 0;

}

}

else

{

Fill.SelectedIndex = 0;

var error = CustomMessageBox.Show("Уведомление", $"Проверьте интернет - соединение и повторите попытку.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

}

}

else // ручное

{

ClearField();

newCar = new Car();

UnBlockField();

ComboBoxCarBrand.Visibility = Visibility.Collapsed;

CarBrand.Visibility = Visibility.Visible;

ComboBoxCarModel.Visibility = Visibility.Collapsed;

CarModel.Visibility = Visibility.Visible;

ComboBoxCarGeneration.Visibility = Visibility.Collapsed;

CarGeneration.Visibility = Visibility.Visible;

ComboBoxCarEquipment.Visibility = Visibility.Collapsed;

CarEquipment.Visibility = Visibility.Visible;

Button\_Web.Visibility = Visibility.Collapsed;

}

}

private void EngineType\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

if (EngineType.SelectedIndex == 0)

{

Volume.Visibility = Visibility.Visible;

TextVolume.Visibility = Visibility.Visible;

TextBlockBatteryCapacity.Visibility = Visibility.Collapsed;

TextBlockRangePerCharge.Visibility = Visibility.Collapsed;

BatteryCapacity.Visibility = Visibility.Collapsed;

RangePerCharge.Visibility = Visibility.Collapsed;

Petrol.Visibility = Visibility.Visible;

Gaz.Visibility = Visibility.Visible;

Diesel.Visibility = Visibility.Visible;

Electro.Visibility = Visibility.Collapsed;

}

else if(EngineType.SelectedIndex == 1)

{

Volume.Visibility = Visibility.Collapsed;

TextVolume.Visibility = Visibility.Collapsed;

TextBlockBatteryCapacity.Visibility = Visibility.Visible;

TextBlockRangePerCharge.Visibility = Visibility.Visible;

BatteryCapacity.Visibility = Visibility.Visible;

RangePerCharge.Visibility = Visibility.Visible;

Fuel.SelectedIndex = 2;

Petrol.Visibility = Visibility.Collapsed;

Gaz.Visibility = Visibility.Collapsed;

Diesel.Visibility = Visibility.Collapsed;

Electro.Visibility = Visibility.Visible;

}

else if(EngineType.SelectedIndex == 2)

{

Volume.Visibility = Visibility.Visible;

TextVolume.Visibility = Visibility.Visible;

TextBlockBatteryCapacity.Visibility = Visibility.Visible;

TextBlockRangePerCharge.Visibility = Visibility.Visible;

BatteryCapacity.Visibility = Visibility.Visible;

RangePerCharge.Visibility = Visibility.Visible;

Petrol.Visibility = Visibility.Visible;

Gaz.Visibility = Visibility.Visible;

Diesel.Visibility = Visibility.Visible;

Electro.Visibility = Visibility.Collapsed;

}

OctaneNumber.SelectedIndex = -1;

Fuel.SelectedIndex = -1;

Volume.Text = "";

}

private async void ComboBoxCarBrand\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

int index = this.ComboBoxCarBrand.SelectedIndex;

if (index != -1)

{

ClearField();

this.ComboBoxCarModel.Items.Clear();

this.ComboBoxCarGeneration.Items.Clear();

this.ComboBoxCarEquipment.Items.Clear();

selectUrl = carName[index].Item2;

carModel = await GetCarData(selectUrl, "//div[@class='css-1dk948p ehmqafe0']");

if (carModel.Count != 0)

{

carModel = carModel.OrderBy(t => t.Item1).ToList();

foreach (Tuple<string, string> car in carModel)

{

this.ComboBoxCarModel.Items.Add(car.Item1);

}

}

else

{

Fill.SelectedIndex = 0;

}

}

}

private async void ComboBoxCarModel\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

int index = this.ComboBoxCarModel.SelectedIndex;

if (index != -1)

{

ClearField();

this.ComboBoxCarGeneration.Items.Clear();

this.ComboBoxCarEquipment.Items.Clear();

selectUrl = carModel[index].Item2;

carGeneration = await GetCarGeneration(selectUrl, "//div[@class='css-10ib5jr']");

if (carGeneration.Count != 0)

{

carGeneration = carGeneration.OrderBy(t => t.Item1).ToList();

foreach (Tuple<string, string> car in carGeneration)

{

this.ComboBoxCarGeneration.Items.Add(car.Item1);

}

}

else

{

Fill.SelectedIndex = 0;

}

}

}

private async void ComboBoxCarGeneration\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

int index = this.ComboBoxCarGeneration.SelectedIndex;

if (index != -1)

{

ClearField();

this.ComboBoxCarEquipment.Items.Clear();

string url = selectUrl + carGeneration[index].Item2;

carEquipment = await GetCarEquipment(url, "//table[@class='b-table b-table\_text-left']");

carEquipment.RemoveAll(t => string.IsNullOrEmpty(t.Item1) | t.Item2.Contains("engine") | t.Item2.Contains("#"));

if (carEquipment.Count == 0)

{

CustomMessageBox.Show("Уведомление","Извините, произошла ошибка. Информация о данном автомобиле не найдена или автомобиль не был выпущен.", MessageBoxButton.OK,CustomMessageBox.MessageBoxImage.Information);

}

foreach (Tuple<string, string> car in carEquipment)

{

this.ComboBoxCarEquipment.Items.Add(car.Item1);

}

}

}

private void ComboBoxCarEquipment\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

int index = this.ComboBoxCarEquipment.SelectedIndex;

if (index != -1)

{

ClearField();

string url = $"https://www.drom.ru/{carEquipment[index].Item2}";

GetAllData(url, "//div[@class='b-media-cont b-media-cont\_margin\_b-size-m bm-catCompTableWrap']");

}

}

public async Task<List<Tuple<string, string>>> GetCarData(string url, string xpath)

{

List<Tuple<string, string>> data = new List<Tuple<string, string>>();

int retryCount = 0;

while (data.Count == 0 && retryCount < 3)

{

try

{

if (!(await CheckInternetConnectionAsync()))

{

break;

}

HtmlWeb webGet = new HtmlWeb();

HtmlDocument document = await webGet.LoadFromWebAsync(url);

HtmlNode parentDivNode = document.DocumentNode.SelectSingleNode(xpath);

if (parentDivNode != null)

{

HtmlNodeCollection anchorNodes = parentDivNode.SelectNodes(".//a");

if (anchorNodes != null)

{

foreach (HtmlNode anchorNode in anchorNodes)

{

string linkText = anchorNode.InnerText.Trim();

string href = anchorNode.GetAttributeValue("href", "");

if (!string.IsNullOrEmpty(href))

{

data.Add(Tuple.Create(linkText, href));

}

}

}

}

if (data.Count > 0)

{

break;

}

}

catch (Exception ex)

{

await Application.Current.Dispatcher.InvokeAsync(() =>

{

CustomMessageBox.Show("Уведомление", $"Произошла ошибка:{ex.Message}", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

});

}

retryCount++;

}

if (data.Count == 0)

{

await Application.Current.Dispatcher.InvokeAsync(() =>

{

CustomMessageBox.Show("Уведомление", $"Данные не удалось получить. Проверьте интернет соединение и повторите попытку позже.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

});

}

return data;

}

public async Task<List<Tuple<string, string>>> GetCarGeneration(string url, string xpath)

{

List<Tuple<string, string>> carDataList = new List<Tuple<string, string>>();

int attempts = 0;

while (carDataList.Count == 0 && attempts < 3)

{

try

{

if (!(await CheckInternetConnectionAsync()))

{

break;

}

HtmlWeb webGet = new HtmlWeb();

HtmlDocument document = await webGet.LoadFromWebAsync(url);

HtmlNode parentDivNode = document.DocumentNode.SelectSingleNode(xpath);

if (parentDivNode != null)

{

HtmlNodeCollection countryDivs = parentDivNode.SelectNodes("div[@class = 'css-pyemnz e1ei9t6a4']");

foreach (HtmlNode countryDivNode in countryDivs)

{

HtmlNodeCollection countryDivTexts = countryDivNode.SelectNodes("div[@class = 'css-112idg0 e1ei9t6a3']");

string countryCode = countryDivTexts[0].GetAttributeValue("id", "");

string countryName = CountryTranslations.ContainsKey(countryCode) ? CountryTranslations[countryCode] : "";

HtmlNodeCollection carDivs = countryDivNode.SelectNodes("div[@class = 'css-kumxje e1naoij51']");

foreach (HtmlNode BlockWithCar in carDivs)

{

HtmlNodeCollection selectCar = BlockWithCar.SelectNodes("div[@class = 'css-0 e1ei9t6a0']");

foreach (HtmlNode car in selectCar)

{

HtmlNodeCollection anchorNodes = car.SelectNodes(".//a");

if (anchorNodes != null && anchorNodes.Count > 0)

{

string href = anchorNodes[0].GetAttributeValue("href", "");

HtmlNode spanNode = car.SelectSingleNode(".//span[@class='css-1089mxj e1ei9t6a2']");

string spanText = spanNode != null ? spanNode.InnerText.Trim() : "";

HtmlNodeCollection divNodes = car.SelectNodes(".//div[@data-ftid='component\_article\_extended-info']/div[@class='css-1qzcj5h e1rlzkvp0'][2]");

string divText = "";

if (divNodes != null)

{

foreach (HtmlNode divNode in divNodes)

{

divText = divNode.InnerText.Trim();

}

}

string resultText = $"{spanText}\n{divText}\n{countryName}";

if (!string.IsNullOrEmpty(resultText))

{

carDataList.Add(Tuple.Create(resultText, href));

}

}

}

}

}

}

}

catch (Exception ex)

{

CustomMessageBox.Show("Уведомление", $"Произошла ошибка:{ex.Message}", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

}

attempts++;

if (carDataList.Count == 0)

{

CustomMessageBox.Show("Уведомление", $"Данные не удалось получить. Проверьте интернет соединение и повторите попытку позже.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

}

}

return carDataList;

}

public async Task<List<Tuple<string, string>>> GetCarEquipment(string url, string xpath)

{

List<Tuple<string, string>> data = new List<Tuple<string, string>>();

int attempts = 0;

while (attempts < 3)

{

try

{

if (!(await CheckInternetConnectionAsync()))

{

break;

}

HtmlWeb webGet = new HtmlWeb();

HtmlDocument document = await webGet.LoadFromWebAsync(url);

HtmlNode parentDivNode = document.DocumentNode.SelectSingleNode(xpath);

if (parentDivNode == null)

{

return data;

}

HtmlNodeCollection anchorNodes = parentDivNode.SelectNodes(".//a");

if (anchorNodes != null)

{

foreach (HtmlNode anchorNode in anchorNodes)

{

string linkText = anchorNode.InnerText.Trim();

string href = anchorNode.GetAttributeValue("href", "");

if (!string.IsNullOrEmpty(href))

{

data.Add(Tuple.Create(linkText, href));

}

}

}

break;

}

catch (Exception ex)

{

CustomMessageBox.Show("Уведомление", $"Произошла ошибка:{ex.Message}", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

}

attempts++;

}

if (data.Count == 0)

{

CustomMessageBox.Show("Уведомление", $"Данные не удалось получить. Проверьте интернет соединение и повторите попытку позже.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

}

return data;

}

public string GetAllData(string url, string xpath)

{

string data = "";

List<Tuple<string, string>> info = new List<Tuple<string, string>>();

try

{

HtmlWeb webGet = new HtmlWeb();

HtmlDocument document = webGet.Load(url);

HtmlNode parentDivNode = document.DocumentNode.SelectSingleNode(xpath);

if (parentDivNode == null)

{

return "";

}

HtmlNodeCollection anchorNodes = parentDivNode.SelectNodes(".//td");

if (anchorNodes != null)

{

bool skipFirst = true;

string key = "";

foreach (HtmlNode tdNode in anchorNodes)

{

string text = tdNode.InnerText.Trim();

string svg = tdNode.InnerHtml.Trim();

if (svg.Contains("svg") && text.Length == 0)

{

text = "Есть";

}

if (!IsIgnoredText(text))

{

data += text;

if (skipFirst)

{

key = text;

skipFirst = false;

}

else

{

info.Add(new Tuple<string, string>(key, text));

skipFirst = true;

}

}

}

}

}

catch (Exception ex)

{

MessageBox.Show("Произошла ошибка: " + ex.Message);

}

var filteredItems = info.Where(x => x.Item1 == "Название комплектации" ||

x.Item1 == "Период выпуска" ||

x.Item1 == "Тип привода" ||

x.Item1 == "Тип кузова" ||

x.Item1 == "Тип трансмиссии" ||

x.Item1 == "Максимальная скорость, км/ч" ||

x.Item1 == "Объем двигателя, куб.см" ||

x.Item1 == "Число дверей" ||

x.Item1 == "Число мест" ||

x.Item1 == "Используемое топливо" ||

x.Item1 == "Кондиционер" ||

x.Item1 == "Расход топлива в смешанном цикле, л/100 км" ||

x.Item1 == "Запас хода на электротяге в км" ||

x.Item1 == "Ёмкость батареи, кВт\*ч"

).ToList();

string textInfo = "";

if (filteredItems.Any(x => x.Item1 == "Ёмкость батареи, кВт\*ч") & filteredItems.Any(x => x.Item1 == "Используемое топливо" && x.Item2 == "Электричество"))

{

ClearField();

newCar = new Car();

Conditioner.SelectedIndex = 0;

EngineType.SelectedIndex = 1;

FuelConsumption.Text = null;

newCar.Brand = ComboBoxCarBrand.SelectedItem.ToString();

newCar.Model = ComboBoxCarModel.SelectedItem.ToString();

newCar.Generation = ComboBoxCarGeneration.SelectedItem.ToString().Split('\r')[0];

newCar.Equipment = ComboBoxCarEquipment.SelectedItem.ToString();

newCar.EngineType = "Электрический";

EngineType.SelectedIndex = 1;

TextVolume.Visibility = Visibility.Collapsed;

Volume.Visibility = Visibility.Collapsed;

TextFuelConsumption.Text = "Расход батареии, кВт\*ч/км";

foreach (var item in filteredItems)

{

switch (item.Item1)

{

case "Период выпуска":

string patternNumber = @"\b\d{4}\b";

Regex regexNumber = new Regex(patternNumber);

MatchCollection matchesNumber = regexNumber.Matches(item.Item2);

if (matchesNumber.Count == 2)

{

Year.Text = $"{matchesNumber[0].Value} - {matchesNumber[1].Value}";

newCar.Year = $"{matchesNumber[0].Value} - {matchesNumber[1].Value}";

}

else

{

Year.Text = $"{matchesNumber[0].Value} - н.в.";

newCar.Year = $"{matchesNumber[0].Value} - н.в.";

}

break;

case "Тип привода":

SelectComboBoxItemByText(Drive, item.Item2.Split(' ')[0]);

newCar.Drive = item.Item2.Split(' ')[0];

break;

case "Тип кузова":

string body = "";

if (item.Item2 == "SUV")

{

body = "Внедорожник";

}

else if (item.Item2 == "Открытый кузов")

{

body = "Кабриолет";

}

else

{

body = item.Item2.Split(' ')[0];

}

SelectComboBoxItemByText(Body, body);

newCar.Body = body;

break;

case "Тип трансмиссии":

string gearBox = item.Item2.Split(' ')[0];

if (gearBox == "РКПП" | gearBox == "АКПП" | gearBox == "Вариатор" | gearBox == "Редуктор")

{

GearBox.SelectedIndex = 0;

gearBox = "АКПП";

}

else if (gearBox == "МКПП")

{

GearBox.SelectedIndex = 1;

gearBox = "МКПП";

}

newCar.GearBox = gearBox;

break;

case "Максимальная скорость, км/ч":

MaxSpeed.Text = item.Item2;

newCar.MaxSpeed = Convert.ToInt32(item.Item2);

break;

case "Число дверей":

Doors.Text = item.Item2;

newCar.Doors = Convert.ToInt32(item.Item2);

break;

case "Число мест":

Seats.Text = item.Item2;

newCar.Seats = Convert.ToInt32(item.Item2);

break;

case "Используемое топливо":

TextOctaneNumber.Visibility = Visibility.Collapsed;

OctaneNumber.Visibility = Visibility.Collapsed;

Fuel.SelectedIndex = 2;

OctaneNumber.SelectedIndex = -1;

newCar.Fuel = "Электричество";

break;

case "Кондиционер":

if (item.Item2 == "Есть")

{

Conditioner.SelectedIndex = 1;

newCar.Conditioner = "Есть";

}

else

{

Conditioner.SelectedIndex = 0;

newCar.Conditioner = "Нет";

}

break;

case "Расход топлива в смешанном цикле, л/100 км":

FuelConsumption.Text = item.Item2;

newCar.FuelConsumption = Convert.ToDouble(item.Item2.ToString());

break;

case "Ёмкость батареи, кВт\*ч":

BatteryCapacity.Text = item.Item2;

newCar.BatteryCapacity = Convert.ToDouble(item.Item2.Replace('.',','));

break;

case "Запас хода на электротяге в км":

TextBlockRangePerCharge.Visibility = Visibility.Visible;

RangePerCharge.Visibility = Visibility.Visible;

RangePerCharge.Text = item.Item2;

newCar.RangePerCharge = Convert.ToDouble(item.Item2);

break;

}

}

if (newCar.MaxSpeed == null)

{

MaxSpeed.Text = 150.ToString();

newCar.MaxSpeed = 150;

}

double fuelConsumption = Math.Round(Convert.ToDouble(BatteryCapacity.Text.Replace('.',',')) / Convert.ToDouble(RangePerCharge.Text), 2);

FuelConsumption.Text = fuelConsumption.ToString();

newCar.FuelConsumption = fuelConsumption;

}

if ((!filteredItems.Any(x => x.Item1 == "Ёмкость батареи, кВт\*ч")) & filteredItems.Any(x => x.Item1 == "Используемое топливо" && (x.Item2.Contains("Бензин") | x.Item2.Contains("Дизельное топливо") | x.Item2.Contains("Природный газ") | x.Item2.Contains("Газ/бензин"))))

{

ClearField();

newCar = new Car();

Conditioner.SelectedIndex = 0;

EngineType.SelectedIndex = 0;

FuelConsumption.Text = null;

TextFuelConsumption.Text = "Расход топлива в смешанном цикле, л/100 км";

newCar.Brand = ComboBoxCarBrand.SelectedItem.ToString();

newCar.Model = ComboBoxCarModel.SelectedItem.ToString();

newCar.Generation = ComboBoxCarGeneration.SelectedItem.ToString().Split('\r')[0];

newCar.Equipment = ComboBoxCarEquipment.SelectedItem.ToString();

newCar.EngineType = "ДВС";

foreach (var item in filteredItems)

{

switch (item.Item1)

{

case "Период выпуска":

string patternNumber = @"\b\d{4}\b";

Regex regexNumber = new Regex(patternNumber);

MatchCollection matchesNumber = regexNumber.Matches(item.Item2);

if(matchesNumber.Count == 2)

{

Year.Text = $"{matchesNumber[0].Value} - {matchesNumber[1].Value}";

newCar.Year = $"{matchesNumber[0].Value} - {matchesNumber[1].Value}";

}

else

{

Year.Text = $"{matchesNumber[0].Value} - н.в.";

newCar.Year = $"{matchesNumber[0].Value} - н.в.";

}

break;

case "Тип привода":

SelectComboBoxItemByText(Drive, item.Item2.Split(' ')[0]);

newCar.Drive = item.Item2.Split(' ')[0];

break;

case "Тип кузова":

string body = "";

if( item.Item2 == "SUV")

{

body = "Внедорожник";

} else if( item.Item2 == "Открытый кузов")

{

body = "Кабриолет";

}

else

{

body = item.Item2.Split(' ')[0];

}

SelectComboBoxItemByText(Body, body);

newCar.Body = body;

break;

case "Тип трансмиссии":

string gearBox = item.Item2.Split(' ')[0];

if (gearBox == "РКПП" | gearBox == "АКПП" | gearBox == "Вариатор" | gearBox == "Редуктор")

{

GearBox.SelectedIndex = 0;

gearBox = "АКПП";

}

else if (gearBox == "МКПП")

{

GearBox.SelectedIndex = 1;

gearBox = "МКПП";

}

newCar.GearBox = gearBox;

break;

case "Максимальная скорость, км/ч":

MaxSpeed.Text = item.Item2;

newCar.MaxSpeed = Convert.ToInt32(item.Item2);

break;

case "Объем двигателя, куб.см":

TextVolume.Visibility = Visibility.Visible;

Volume.Visibility = Visibility.Visible;

Volume.Text = item.Item2;

newCar.Volume = Convert.ToInt32(item.Item2);

break;

case "Число дверей":

Doors.Text = item.Item2;

newCar.Doors = Convert.ToInt32(item.Item2);

break;

case "Число мест":

Seats.Text = item.Item2;

newCar.Seats = Convert.ToInt32(item.Item2);

break;

case "Используемое топливо":

string[] fuelParts = item.Item2.Split(' ');

if (item.Item2.Contains("Бензин"))

{

newCar.Fuel = "Бензин";

Fuel.SelectedIndex = 0;

if (fuelParts.Length >= 2)

{

string patternOctaneNumber = @"АИ-\d+";

Regex regexOctaneNumber = new Regex(patternOctaneNumber);

MatchCollection matches = regexOctaneNumber.Matches(item.Item2);

if (matches.Count != 0)

{

int index = octaneNumber.FindIndex(x => x.octaneNumber == matches[0].ToString());

OctaneNumber.SelectedIndex = index;

newCar.FuelOctane = octaneNumber[index].id;

}

else

{

int index = octaneNumber.FindIndex(x => x.octaneNumber == "АИ-95");

OctaneNumber.SelectedIndex = index;

newCar.FuelOctane = octaneNumber[index].id;

}

}

else

{

int index = octaneNumber.FindIndex(x => x.octaneNumber == "АИ-95");

OctaneNumber.SelectedIndex = index;

newCar.FuelOctane = octaneNumber[index].id;

}

}

else if (item.Item2.Contains("Дизельное топливо"))

{

TextOctaneNumber.Visibility = Visibility.Collapsed;

OctaneNumber.Visibility = Visibility.Collapsed;

Fuel.SelectedIndex = 1;

OctaneNumber.SelectedIndex = -1;

newCar.Fuel = "Дизельное топливо";

newCar.FuelOctane = null;

}

else if (item.Item2.Contains("Природный газ") | item.Item2.Contains("Газ/бензин"))

{

TextOctaneNumber.Visibility = Visibility.Collapsed;

OctaneNumber.Visibility = Visibility.Collapsed;

Fuel.SelectedIndex = 3;

OctaneNumber.SelectedIndex = 0;

newCar.Fuel = "Газ";

newCar.FuelOctane = null;

}

break;

case "Кондиционер":

if (item.Item2 == "Есть")

{

Conditioner.SelectedIndex = 1;

newCar.Conditioner = "Есть";

}

else

{

Conditioner.SelectedIndex = 0;

newCar.Conditioner = "Нет";

}

break;

case "Расход топлива в смешанном цикле, л/100 км":

FuelConsumption.Text = item.Item2;

newCar.FuelConsumption = Convert.ToDouble(item.Item2.ToString());

break;

}

}

if (newCar.MaxSpeed == null)

{

MaxSpeed.Text = 150.ToString();

newCar.MaxSpeed = 150;

}

}

if (filteredItems.Any(x => x.Item1 == "Ёмкость батареи, кВт\*ч") & filteredItems.Any(x => x.Item1 == "Используемое топливо" && (x.Item2.Contains("Бензин") | x.Item2.Contains("Дизельное топливо"))))

{

ClearField();

newCar = new Car();

Conditioner.SelectedIndex = 0;

EngineType.SelectedIndex = 2;

FuelConsumption.Text = null;

TextFuelConsumption.Text = "Расход топлива в смешанном цикле, л/100 км";

newCar.Brand = ComboBoxCarBrand.SelectedItem.ToString();

newCar.Model = ComboBoxCarModel.SelectedItem.ToString();

newCar.Generation = ComboBoxCarGeneration.SelectedItem.ToString().Split('\r')[0];

newCar.Equipment = ComboBoxCarEquipment.SelectedItem.ToString();

newCar.EngineType = "Гибрид";

bool rangeFound = false;

foreach (var item in filteredItems)

{

switch (item.Item1)

{

case "Период выпуска":

string patternNumber = @"\b\d{4}\b";

Regex regexNumber = new Regex(patternNumber);

MatchCollection matchesNumber = regexNumber.Matches(item.Item2);

if (matchesNumber.Count == 2)

{

Year.Text = $"{matchesNumber[0].Value} - {matchesNumber[1].Value}";

newCar.Year = $"{matchesNumber[0].Value} - {matchesNumber[1].Value}";

}

else

{

Year.Text = $"{matchesNumber[0].Value} - н.в.";

newCar.Year = $"{matchesNumber[0].Value} - н.в.";

}

break;

case "Тип привода":

SelectComboBoxItemByText(Drive, item.Item2.Split(' ')[0]);

newCar.Drive = item.Item2.Split(' ')[0];

break;

case "Тип кузова":

string body = "";

if (item.Item2 == "SUV")

{

body = "Внедорожник";

}

else if (item.Item2 == "Открытый кузов")

{

body = "Кабриолет";

}

else

{

body = item.Item2.Split(' ')[0];

}

SelectComboBoxItemByText(Body, body);

newCar.Body = body;

break;

case "Тип трансмиссии":

string gearBox = item.Item2.Split(' ')[0];

if (gearBox == "РКПП" | gearBox == "АКПП" | gearBox == "Вариатор" | gearBox == "Редуктор")

{

GearBox.SelectedIndex = 0;

gearBox = "AКПП";

}

else if (gearBox == "МКПП")

{

GearBox.SelectedIndex = 1;

gearBox = "МКПП";

}

newCar.GearBox = gearBox;

break;

case "Максимальная скорость, км/ч":

MaxSpeed.Text = item.Item2;

newCar.MaxSpeed = Convert.ToInt32(item.Item2);

break;

case "Объем двигателя, куб.см":

TextVolume.Visibility = Visibility.Visible;

Volume.Visibility = Visibility.Visible;

Volume.Text = item.Item2;

newCar.Volume = Convert.ToInt32(item.Item2);

break;

case "Число дверей":

Doors.Text = item.Item2;

newCar.Doors = Convert.ToInt32(item.Item2);

break;

case "Число мест":

Seats.Text = item.Item2;

newCar.Seats = Convert.ToInt32(item.Item2);

break;

case "Используемое топливо":

string[] fuelParts = item.Item2.Split(' ');

if (item.Item2.Contains("Бензин"))

{

newCar.Fuel = "Бензин";

Fuel.SelectedIndex = 0;

if (fuelParts.Length >= 2)

{

string patternOctaneNumber = @"АИ-\d+";

Regex regexOctaneNumber = new Regex(patternOctaneNumber);

MatchCollection matches = regexOctaneNumber.Matches(item.Item2);

if (matches.Count != 0)

{

int index = octaneNumber.FindIndex(x => x.octaneNumber == matches[0].ToString());

OctaneNumber.SelectedIndex = index;

newCar.FuelOctane = octaneNumber[index].id;

}

else

{

int index = octaneNumber.FindIndex(x => x.octaneNumber == "АИ-95");

OctaneNumber.SelectedIndex = index;

newCar.FuelOctane = octaneNumber[index].id;

}

}

else

{

int index = octaneNumber.FindIndex(x => x.octaneNumber == "АИ-95");

OctaneNumber.SelectedIndex = index;

newCar.FuelOctane = octaneNumber[index].id;

}

}

else if (item.Item2.Contains("Дизельное топливо"))

{

TextOctaneNumber.Visibility = Visibility.Collapsed;

OctaneNumber.Visibility = Visibility.Collapsed;

Fuel.SelectedIndex = 1;

OctaneNumber.SelectedIndex = -1;

newCar.Fuel = "Дизельное топливо";

newCar.FuelOctane = null;

}

else if (item.Item2.Contains("Природный газ") | item.Item2.Contains("Газ/бензин"))

{

TextOctaneNumber.Visibility = Visibility.Collapsed;

OctaneNumber.Visibility = Visibility.Collapsed;

Fuel.SelectedIndex = 3;

OctaneNumber.SelectedIndex = 0;

newCar.Fuel = "Газ";

newCar.FuelOctane = null;

}

break;

case "Кондиционер":

if (item.Item2 == "Есть")

{

Conditioner.SelectedIndex = 1;

newCar.Conditioner = "Есть";

}

else

{

Conditioner.SelectedIndex = 0;

newCar.Conditioner = "Нет";

}

break;

case "Расход топлива в смешанном цикле, л/100 км":

FuelConsumption.Text = item.Item2;

newCar.FuelConsumption = Convert.ToDouble(item.Item2.ToString());

break;

case "Ёмкость батареи, кВт\*ч":

BatteryCapacity.Text = item.Item2;

newCar.BatteryCapacity = Convert.ToDouble(item.Item2);

break;

case "Запас хода на электротяге в км"://гибрид

TextBlockRangePerCharge.Visibility = Visibility.Visible;

RangePerCharge.Visibility = Visibility.Visible;

RangePerCharge.Text = item.Item2;

newCar.RangePerCharge = Convert.ToDouble(item.Item2);

rangeFound = true;

break;

}

}

if (!rangeFound)

{

TextBlockRangePerCharge.Text = "";

TextBlockRangePerCharge.Visibility = Visibility.Collapsed;

RangePerCharge.Visibility = Visibility.Collapsed;

}

if (newCar.MaxSpeed == null)

{

MaxSpeed.Text = 150.ToString();

newCar.MaxSpeed = 150;

}

}

return textInfo;

}

private bool IsIgnoredText(string text)

{

string[] ignoredTexts = {

"Основные параметры", "Размеры", "Размеры кузова", "Размеры салона",

"Размерности ходовой части", "Вес и допустимые нагрузки", "Объёмы", "Дополнительно",

"Двигатель, коробка передач и рулевое управление", "Двигатель", "Электротяга",

"Расход топлива", "Коробка передач", "Рулевое управление", "Подвеска / Ходовая часть",

"Подвеска", "Диски", "Шины", "Тормоза", "Экстерьер и внешнее оснащение", "Фары",

"Боковые зеркала", "Двери", "Стёкла", "Крыша", "Аксессуары", "Внутреннее оснащение",

"Руль и центральная панель", "Исполнение салона", "Сиденья", "Электропакет салона",

"Доп. оборудование салона", "Активные и пассивные системы безопасности",

"Подушки безопасности", "Детская безопасность", "Активная и пассивная безопасность",

"Электронные системы безопасности и контроля движения", "Удобства", "Микроклимат салона",

"Аудио системы", "Навигационное и видео оборудование", "Дополнительные удобства",

"Сохранность автомобиля", "Комплект водителя","Окраска кузова","Прочее"

};

return ignoredTexts.Contains(text);

}

public void SelectComboBoxItemByText(ComboBox comboBox, string searchText)

{

foreach (ComboBoxItem item in comboBox.Items)

{

if (item.Content.ToString() == searchText)

{

comboBox.SelectedItem = item;

break;

}

}

}

private void Fuel\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

if (Fuel.SelectedIndex == 2)

{

OctaneNumber.Items.Clear();

TextOctaneNumber.Visibility = Visibility.Collapsed;

OctaneNumber.Visibility = Visibility.Collapsed;

}

else if(Fuel.SelectedIndex == 3)

{

OctaneNumber.Items.Clear();

TextOctaneNumber.Visibility = Visibility.Visible;

OctaneNumber.Visibility = Visibility.Visible;

octaneNumber = db.GetOctaneNumber("Газ");

foreach (var item in octaneNumber)

{

string octane = item.octaneNumber.ToString();

OctaneNumber.Items.Add(octane);

}

}

else if (Fuel.SelectedIndex != -1)

{

OctaneNumber.Items.Clear();

TextOctaneNumber.Visibility = Visibility.Visible;

OctaneNumber.Visibility = Visibility.Visible;

octaneNumber = db.GetOctaneNumber(((ComboBoxItem)Fuel.SelectedItem).Content.ToString());

foreach (var item in octaneNumber)

{

string octane = item.octaneNumber.ToString();

OctaneNumber.Items.Add(octane);

}

}

OctaneNumber.SelectedIndex = -1;

}

private void Seats\_PreviewTextInput(object sender, System.Windows.Input.TextCompositionEventArgs e)

{

if (!char.IsDigit(e.Text, e.Text.Length - 1))

{

e.Handled = true;

return;

}

string currentText = Seats.Text.Insert(Seats.SelectionStart, e.Text); // Вставляем новый символ в текущий текст

if (!int.TryParse(currentText, out int seat))

{

e.Handled = true;

return;

}

if (seat <= 0)

{

var messageBoxResultTwo = CustomMessageBox.Show("Уведомление", $"Число мест должно быть больше 0", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

e.Handled = true;

}

else if (seat > 10)

{

CustomMessageBox.Show("Уведомление", $"Число мест должно быть меньше или равно 10", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

e.Handled = true;

}

}

private void Doord\_PreviewTextInput(object sender, System.Windows.Input.TextCompositionEventArgs e)

{

string currentText = Doors.Text.Insert(Doors.SelectionStart, e.Text);

if (!int.TryParse(currentText, out int seat))

{

e.Handled = true;

return;

}

if (seat <= 0)

{

CustomMessageBox.Show("Уведомление", $"Число дверей должно быть больше 0", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

e.Handled = true;

}

else if (seat > 5)

{

CustomMessageBox.Show("Уведомление", $"Число дверей должно меньше или равно 5", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

e.Handled = true;

}

}

private void Volume\_PreviewTextInput(object sender, System.Windows.Input.TextCompositionEventArgs e)

{

if (!char.IsDigit(e.Text, e.Text.Length - 1))

{

e.Handled = true;

return;

}

string currentText = Volume.Text.Insert(Volume.SelectionStart, e.Text);

if (!int.TryParse(currentText, out int volume))

{

e.Handled = true;

return;

}

if (volume <= 0)

{

CustomMessageBox.Show("Уведомление", $"Объем двигателя должен быть больше 0", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

e.Handled = true;

}

else if (volume > 10000)

{

CustomMessageBox.Show("Уведомление", $"Объем двигателя должен быть меньше 10000", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

e.Handled = true;

}

}

private void Year\_PreviewTextInput(object sender, System.Windows.Input.TextCompositionEventArgs e)

{

if (!char.IsDigit(e.Text, e.Text.Length - 1))

{

e.Handled = true;

return;

}

string currentText = Year.Text.Insert(Year.SelectionStart, e.Text);

if (!int.TryParse(currentText, out int volume))

{

e.Handled = true;

return;

}

if (volume > 2024)

{

CustomMessageBox.Show("Уведомление", $"Год должен быть меньше 2024", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

e.Handled = true;

}

}

public void ClearField()

{

foreach (UIElement element in FilterGrid.Children)

{

if (element is TextBox textBox)

{

textBox.Clear();

}

else if (element is ComboBox comboBox)

{

if (comboBox.Name != "ComboBoxCarBrand" & comboBox.Name != "ComboBoxCarModel" & comboBox.Name != "ComboBoxCarGeneration" & comboBox.Name != "ComboBoxCarEquipment")

{

comboBox.SelectedIndex = -1;

}

}

}

}

private void BlockField()

{

foreach (UIElement element in FilterGrid.Children)

{

if (element is TextBox textBox)

{

textBox.IsReadOnly = true;

}

else if (element is ComboBox comboBox)

{

if (comboBox.Name != "ComboBoxCarBrand" & comboBox.Name != "ComboBoxCarModel" & comboBox.Name != "ComboBoxCarGeneration" & comboBox.Name != "ComboBoxCarEquipment")

{

comboBox.IsEnabled = false;

}

}

}

}

private void UnBlockField()

{

foreach (UIElement element in FilterGrid.Children)

{

if (element is TextBox textBox)

{

textBox.IsReadOnly = false;

}

else if (element is ComboBox comboBox)

{

comboBox.IsEnabled = true;

}

}

}

private void Button\_Click(object sender, RoutedEventArgs e)

{

if(Fill.SelectedIndex == 1)

{

if(CarCheck(newCar)) // проверка автомобиля на заполенность

{

newCar.IdUser = IdUser;

if(db.InsertCar(newCar))

{

CustomMessageBox.Show("Уведомление", $"Автомобиль {newCar.Brand} {newCar.Model} был добавлен в систему.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

}

else

{

CustomMessageBox.Show("Уведомление", $"Автомобиль {newCar.Brand} {newCar.Model} не был добавлен в систему.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

}

}

else

{

CustomMessageBox.Show("Уведомление", $"Недостаточно информации об автомобиле {newCar.Brand} {newCar.Model}. Выберите другой автомобиль.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

}

}

else

{

if (string.IsNullOrEmpty(CarBrand.Text) || string.IsNullOrEmpty(CarModel.Text) || string.IsNullOrEmpty(CarGeneration.Text)

|| string.IsNullOrEmpty(CarEquipment.Text) || string.IsNullOrEmpty(Year.Text) || string.IsNullOrEmpty(Seats.Text)

|| string.IsNullOrEmpty(Doors.Text) || string.IsNullOrEmpty(FuelConsumption.Text))

{

CustomMessageBox.Show("Уведомление", $"Пожалуйста, заполните все обязательные поля.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

return;

}

int seats, doors;

double volume = 0 ,fuelConsumption, rangePerCharge = 0, batteryCapacity = 0;

if (!int.TryParse(Seats.Text, out seats) || seats <= 0)

{

CustomMessageBox.Show("Предупреждение", $"Пожалуйста, введите корректное значение для количества сидений.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

return;

}

if (!int.TryParse(Doors.Text, out doors) || doors <= 0)

{

CustomMessageBox.Show("Предупреждение", $"Пожалуйста, введите корректное значение для количества дверей.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

return;

}

int? maxSpeed = null;

if (MaxSpeed.Text != "")

{

if (!int.TryParse(MaxSpeed.Text, out int parsedMaxSpeed) || parsedMaxSpeed <= 0)

{

CustomMessageBox.Show("Предупреждение", $"Пожалуйста, введите корректное значение для максимальной скорости.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

return;

}

maxSpeed = parsedMaxSpeed;

}

if (Volume.Visibility == Visibility.Visible)

{

if (!double.TryParse(Volume.Text, out volume) || volume <= 0)

{

CustomMessageBox.Show("Предупреждение", $"Пожалуйста, введите корректное значение для объема двигателя.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

return;

}

}

if (!double.TryParse(FuelConsumption.Text, out fuelConsumption) || fuelConsumption <= 0)

{

CustomMessageBox.Show("Предупреждение", $"Пожалуйста, введите корректное значение для расхода топлива.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

return;

}

if (RangePerCharge.Visibility == Visibility.Visible)

{

if (!double.TryParse(RangePerCharge.Text, out rangePerCharge) || rangePerCharge <= 0)

{

CustomMessageBox.Show("Предупреждение", $"Пожалуйста, введите корректное значение для пробега на одном заряде.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

return;

}

}

if (RangePerCharge.Visibility == Visibility.Visible)

{

if (!double.TryParse(BatteryCapacity.Text, out batteryCapacity) || batteryCapacity <= 0)

{

CustomMessageBox.Show("Предупреждение", $"Пожалуйста, введите корректное значение для емкости батареи.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

return;

}

}

if (Body.SelectedItem == null || Drive.SelectedItem == null || GearBox.SelectedItem == null

|| EngineType.SelectedItem == null || Fuel.SelectedItem == null || Conditioner.SelectedItem == null)

{

CustomMessageBox.Show("Предупреждение", $"Пожалуйста, выберите значение для всех параметров.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

return;

}

Car newCar = new Car()

{

Brand = CarBrand.Text,

Model = CarModel.Text,

Generation = CarGeneration.Text,

Equipment = CarEquipment.Text,

Year = Year.Text,

Body = ((ComboBoxItem)Body.SelectedItem).Content.ToString(),

Drive = ((ComboBoxItem)Drive.SelectedItem).Content.ToString(),

GearBox = ((ComboBoxItem)GearBox.SelectedItem).Content.ToString(),

Seats = seats,

Doors = doors,

MaxSpeed = MaxSpeed.Text != "" ? maxSpeed : (int?)null,

EngineType = ((ComboBoxItem)EngineType.SelectedItem).Content.ToString(),

Volume = Volume.Visibility == Visibility.Visible ? volume : (double?)null,

Fuel = ((ComboBoxItem)Fuel.SelectedItem).Content.ToString(),

FuelOctane = OctaneNumber.Visibility == Visibility.Visible ? OctaneNumber.SelectedIndex : (int?)null,

Conditioner = ((ComboBoxItem)Conditioner.SelectedItem).Content.ToString(),

FuelConsumption = fuelConsumption,

RangePerCharge = RangePerCharge.Visibility == Visibility.Visible ? rangePerCharge : (double?)null,

BatteryCapacity = BatteryCapacity.Visibility == Visibility.Visible ? batteryCapacity : (double?)null

};

if (CarCheck(newCar))

{

newCar.IdUser = IdUser;

if (db.InsertCar(newCar))

{

CustomMessageBox.Show("Уведомление", $"Автомобиль {newCar.Brand} {newCar.Model} был добавлен в систему.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

}

else

{

CustomMessageBox.Show("Уведомление", $"Автомобиль {newCar.Brand} {newCar.Model} не был добавлен в систему.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

}

}

else

{

CustomMessageBox.Show("Уведомление", $"Недостаточно информации об автомобиле {newCar.Brand} {newCar.Model}. Выберите другой автомобиль.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

}

}

}

private bool CarCheck(Car car)

{

if((car.Model != "" && car.Brand != "" && car.Generation != "" && car.Year != "" && car.Body != "" && car.Drive != "" && car.GearBox != "" && car.Seats != 0 && car.Doors != 0 && (car.EngineType == "ДВС" | car.EngineType == "Гибрид") && car.Fuel != "" && car.FuelConsumption > 0)

|(car.Model != "" && car.Brand != "" && car.Generation != "" && car.Year != "" && car.Body != "" && car.Drive != "" && car.GearBox != "" && car.Seats != 0 && car.Doors != 0 && car.EngineType == "Электрический" && car.Fuel != "" && car.FuelConsumption > 0 && car.BatteryCapacity > 0 && car.RangePerCharge > 0))

{

return true;

}

return false;

}

private void Button\_Web\_Click(object sender, RoutedEventArgs e)

{

if (Fill.SelectedIndex == 1)

{

if (ComboBoxCarEquipment.SelectedIndex != -1)

{

Process.Start($"https://www.drom.ru/{carEquipment[ComboBoxCarEquipment.SelectedIndex].Item2}");

}

else

{

CustomMessageBox.Show("Уведомление", $"Выберите автмобиль и повторите попытку", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

}

}

}

public async Task<bool> CheckInternetConnectionAsync()

{

try

{

Ping ping = new Ping();

PingReply reply = await ping.SendPingAsync("www.google.com");

return reply.Status == IPStatus.Success;

}

catch (PingException)

{

return false;

}

}

}

}

<Page x:Class="OneWay.Pages.AddRoutePage"

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:iconPacks="http://metro.mahapps.com/winfx/xaml/iconpacks"

xmlns:local="clr-namespace:OneWay.Pages"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800"

Title="Route">

<Border CornerRadius="5" Background="{DynamicResource SecundaryBackgroundColor}">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="3\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<TextBlock Text="Маршрут" Grid.Column="0" FontSize ="25" Foreground="{DynamicResource TextSecundaryColor}" Margin="10,10,10,10"/>

<Grid Grid.Column="0" Margin="0 50 0 0" >

<Grid.RowDefinitions>

<RowDefinition Height="50"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="60"/>

<RowDefinition Height="50"/>

<RowDefinition Height="50"/>

</Grid.RowDefinitions>

<Grid Grid.Row="0">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="auto"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<TextBlock Grid.Column="0" Text="Точка отправки" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10,10,10,10"/>

<TextBox Grid.Column="1" x:Name="StartPoint" Height="26.6" Tag="Город" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0 0 0 1.5" LostFocus="StartPoint\_LostFocus" Margin="5,12,5,12"/>

<ComboBox Grid.Column="2" x:Name="comboBoxStartPoint" FontSize="18" Margin="10,12,10,11"/>

</Grid>

<ScrollViewer Grid.Row="1" Margin="10 0 0 0">

<StackPanel x:Name="Points">

</StackPanel>

</ScrollViewer>

<Button Grid.Row="2" x:Name="AddCarButton" Content="Добавить промежуточную точку" FontSize="20" Height="30" Margin="10" Click="CreatePointButton\_Click">

<Button.Template>

<ControlTemplate TargetType="Button">

<Grid>

<Border Background="{DynamicResource TextSecundaryColor}" CornerRadius="10">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

</Grid>

</ControlTemplate>

</Button.Template>

</Button>

<Grid Grid.Row="3">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="auto"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<TextBlock Grid.Column="0" Text="Точка прибытия" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10,10,10,10"/>

<TextBox Grid.Column="1" x:Name="FinishPoint" Height="26.6" Tag="Город" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0 0 0 1.5" LostFocus="FinishPoint\_LostFocus" Margin="5,12,5,12"/>

<TextBox Grid.Column="2" x:Name="Distance" Height="26.6" Tag="Общее расстояние" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0 0 0 1.5" Margin="5,12,5,12" Visibility="Hidden"/>

<ComboBox Grid.Column="2" x:Name="comboBoxLastPointInfo" FontSize="18" Margin="10,12,10,11"/>

</Grid>

<StackPanel Grid.Row="4" Orientation="Horizontal">

<Button x:Name="CreateRouteButton" Width="150" Content="Создать" FontSize="20" Height="30" Margin="10" Click="CreateRouteButton\_Click">

<Button.Template>

<ControlTemplate TargetType="Button">

<Grid>

<Border Background="{DynamicResource TextSecundaryColor}" CornerRadius="10">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

</Grid>

</ControlTemplate>

</Button.Template>

</Button>

</StackPanel>

</Grid>

<StackPanel Grid.Column="1">

<TextBlock Text="Настройки" FontSize ="25" Foreground="{DynamicResource TextSecundaryColor}" Margin="10,10,10,0"/>

<TextBlock Text="Заполение полей" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10,10,10,0"/>

<ComboBox Grid.Row="16" Grid.Column="1" x:Name="Fill" FontSize="18" Height="35" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 5 6" SelectionChanged="Fill\_SelectionChanged" >

<ComboBoxItem Content="Ручное"/>

<ComboBoxItem Content="Автоматическое"/>

</ComboBox>

</StackPanel>

<StackPanel Grid.Column="1" x:Name="TripInfo" Margin="0,135,0,0" />

</Grid>

</Border>

</Page>

using MahApps.Metro.IconPacks;

using Newtonsoft.Json.Linq;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net.Http;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Input;

using System.Windows.Media;

using OneWay.Controls;

using OneWay.Class;

using System.Net.NetworkInformation;

namespace OneWay.Pages

{

public partial class AddRoutePage : Page

{

public static string selectToken;

public static string x = null;

public static string y = null;

public int fill = 0;

int IdUser;

DataBase db = new DataBase("OneWayDB.db");

Route selectRoute = new Route();

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

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

List<(double distance, int time)> info = new List<(double distance, int time)>();

public static List<Tuple<string, string>> AllCity = new List<Tuple<string, string>>();

public AddRoutePage(int idUser)

{

InitializeComponent();

Fill.SelectedIndex = 0;

IdUser = idUser;

}

private async void StartPoint\_LostFocus(object sender, RoutedEventArgs e)

{

if (Fill.SelectedIndex != 0)

{

comboBoxStartPoint.Items.Clear();

comboBoxStartPoint.SelectedIndex = -1;

if (StartPoint.Text.Length != 0)

{

List<Tuple<string, string>> route = await GetCityAsync(StartPoint.Text);

AllCity.AddRange(route);

foreach (var item in route)

{

comboBoxStartPoint.Items.Add(item.Item1);

}

}

}

}

private async void FinishPoint\_LostFocus(object sender, RoutedEventArgs e)

{

if(Fill.SelectedIndex != 0)

{

comboBoxLastPointInfo.Items.Clear();

comboBoxLastPointInfo.SelectedIndex = -1;

if (FinishPoint.Text.Length != 0)

{

List<Tuple<string, string>> route = await GetCityAsync(FinishPoint.Text);

AllCity.AddRange(route);

foreach (var item in route)

{

comboBoxLastPointInfo.Items.Add(item.Item1);

}

}

}

}

public async Task<List<Tuple<string, string>>> GetCityAsync(string city)

{

string url = $"https://geocode-maps.yandex.ru/1.x/?apikey=810e6f73-e4e6-4d63-850a-3a87c1c662ef&geocode={city}&format=json";

using (HttpClient client = new HttpClient())

{

try

{

HttpResponseMessage response = await client.GetAsync(url);

if (response.IsSuccessStatusCode)

{

// Получаем содержимое ответа в виде строки

string jsonResponse = await response.Content.ReadAsStringAsync();

JObject jsonObject = JObject.Parse(jsonResponse);

List<Tuple<string, string>> posKeys = GetPosKeys(jsonObject);

if (posKeys.Count == 0)

{

CustomMessageBox.Show("Уведомление", $"Введенный город не найден. Повторите попытку.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

return new List<Tuple<string, string>>();

}

return posKeys;

}

else

{

CustomMessageBox.Show("Уведомление", $"Ошибка: {response.StatusCode}.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

}

}

catch (Exception ex)

{

CustomMessageBox.Show("Уведомление", $"Ошибка: {ex.Message}.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

}

}

return new List<Tuple<string, string>>();

}

public static List<Tuple<string, string>> GetPosKeys(JToken token)

{

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

List<Tuple<string, string>> data = new List<Tuple<string, string>>();

if (token.Type == JTokenType.Object)

{

foreach (JProperty property in token.Children<JProperty>())

{

JToken currentObject = property.Parent;

if (property.Name == "text" && (property.Parent?.SelectToken("kind")?.Value<string>() == "locality" || property.Parent?.SelectToken("kind")?.Value<string>() == "province" || property.Parent?.SelectToken("kind")?.Value<string>() == "house"))

{

x = property.Value.ToString();

selectToken = GetParentPathUntil(currentObject, "GeoObject");

}

if (property.Name == "pos")

{

string objectName = GetParentPathUntil(currentObject, "GeoObject");

if (selectToken == objectName)

{

y = ReverseString(property.Value.ToString());

}

}

data.AddRange(GetPosKeys(property.Value));

if (!string.IsNullOrEmpty(x) && !string.IsNullOrEmpty(y))

{

data.Add(Tuple.Create(x, y));

x = null;

y = null;

}

}

}

else if (token.Type == JTokenType.Array)

{

foreach (JToken arrayItem in token.Children())

{

data.AddRange(GetPosKeys(arrayItem));

}

}

return data;

}

private static string GetParentPathUntil(JToken token, string stopAt)

{

var path = token.Path;

var index = path.LastIndexOf(stopAt, StringComparison.Ordinal);

return index >= 0 ? path.Substring(0, index + stopAt.Length) : path;

}

public static string ReverseString(string input)

{

string[] parts = input.Split(' ');

return $"{parts[1]} {parts[0]}";

}

private void CreatePointButton\_Click(object sender, RoutedEventArgs e)

{

AddPoint();

}

private void AddPoint()

{

Grid grid = new Grid();

grid.ColumnDefinitions.Add(new ColumnDefinition() { Width = new GridLength(155) });

grid.ColumnDefinitions.Add(new ColumnDefinition() { Width = new GridLength(1, GridUnitType.Star) });

grid.ColumnDefinitions.Add(new ColumnDefinition() { Width = new GridLength(1, GridUnitType.Star) });

grid.ColumnDefinitions.Add(new ColumnDefinition() { Width = new GridLength(80) });

TextBlock textCount = new TextBlock()

{

Text = "Новая точка",

Width = 130,

FontSize = 20,

Foreground = (Brush)FindResource("TextSecundaryColor"),

TextAlignment = TextAlignment.Left,

HorizontalAlignment = HorizontalAlignment.Left,

VerticalAlignment = VerticalAlignment.Center,

};

Grid.SetColumn(textCount, 0);

TextBox newTextBox = new TextBox()

{

Height = 26.6,

Tag = "Город",

FontSize = 20,

Foreground = (Brush)FindResource("TextSecundaryColor"),

Background = Brushes.Transparent,

BorderThickness = new Thickness(0, 0, 0, 1),

Margin = new Thickness(5, 0, 5, 0),

VerticalAlignment = VerticalAlignment.Center,

};

if(fill == 1)

{

newTextBox.LostFocus += Point\_LostFocus;

}

Grid.SetColumn(newTextBox, 1);

if(fill == 1 )

{

ComboBox comboBox = new ComboBox()

{

Height = 30,

FontSize = 18,

Margin = new Thickness(5, 0, 10, 0),

};

Grid.SetColumn(comboBox, 2);

grid.Children.Add(comboBox);

}

Button deleteButton = new Button();

deleteButton.Width = 42;

deleteButton.Height = 34;

deleteButton.Margin = new Thickness(10);

deleteButton.BorderThickness = new Thickness(0);

deleteButton.Click += DeleteButton\_Click;

ControlTemplate buttonTemplate = new ControlTemplate(typeof(Button));

FrameworkElementFactory gridFactory = new FrameworkElementFactory(typeof(Grid));

FrameworkElementFactory borderFactory = new FrameworkElementFactory(typeof(Border));

borderFactory.SetValue(Border.BackgroundProperty, Brushes.Red);

borderFactory.SetValue(Border.CornerRadiusProperty, new CornerRadius(10));

FrameworkElementFactory contentPresenterFactory = new FrameworkElementFactory(typeof(ContentPresenter));

contentPresenterFactory.SetValue(HorizontalAlignmentProperty, HorizontalAlignment.Center);

contentPresenterFactory.SetValue(VerticalAlignmentProperty, VerticalAlignment.Center);

borderFactory.AppendChild(contentPresenterFactory);

FrameworkElementFactory iconFactory = new FrameworkElementFactory(typeof(PackIconUnicons));

iconFactory.SetValue(PackIconUnicons.KindProperty, PackIconUniconsKind.Times);

iconFactory.SetValue(Control.ForegroundProperty, (Brush)FindResource("TextSecundaryColor"));

iconFactory.SetValue(HorizontalAlignmentProperty, HorizontalAlignment.Center);

iconFactory.SetValue(VerticalAlignmentProperty, VerticalAlignment.Center);

gridFactory.AppendChild(borderFactory);

gridFactory.AppendChild(iconFactory);

buttonTemplate.VisualTree = gridFactory;

deleteButton.Template = buttonTemplate;

deleteButton.Click += DeleteButton\_Click;

Grid.SetColumn(deleteButton, 3);

grid.Children.Add(textCount);

grid.Children.Add(newTextBox);

grid.Children.Add(deleteButton);

Points.Children.Add(grid);

}

private async void Point\_LostFocus(object sender, RoutedEventArgs e)

{

TextBox textBox = sender as TextBox;

if (textBox != null)

{

Grid parentPanel = textBox.Parent as Grid;

if (parentPanel != null)

{

int index = parentPanel.Children.IndexOf(textBox);

if (index < parentPanel.Children.Count - 1)

{

ComboBox comboBox = parentPanel.Children[0] as ComboBox;

if (comboBox != null)

{

comboBox.Items.Clear();

if (textBox.Text.Length != 0)

{

List<Tuple<string, string>> route = await GetCityAsync(textBox.Text);

AllCity.AddRange(route);

foreach (var item in route)

{

comboBox.Items.Add(item.Item1);

}

}

}

}

}

}

}

private void DeleteButton\_Click(object sender, RoutedEventArgs e)

{

Button button = sender as Button;

if (button != null)

{

Grid parentPanel = button.Parent as Grid;

if (parentPanel != null)

{

Points.Children.Remove(parentPanel);

}

}

}

private async void CreateRouteButton\_Click(object sender, RoutedEventArgs e)

{

if(Fill.SelectedIndex == 1) // проверка типа заполения автоматический

{

additionalPoints = new List<string>();

if (comboBoxStartPoint.SelectedIndex != -1 || comboBoxLastPointInfo.SelectedIndex != -1)

{

string startPoint = this.comboBoxStartPoint.SelectedItem?.ToString();

string lastPoint = this.comboBoxLastPointInfo.SelectedItem?.ToString();

string points = "";

additionalPoints.Add(startPoint);

foreach (var item in Points.Children) // цикл для получения всех дополниетельных точек

{

if (item is Grid)

{

Grid block = item as Grid;

foreach (var element in block.Children)

{

if (element is ComboBox)

{

ComboBox comboBox = element as ComboBox;

if (comboBox.SelectedItem != null)

{

additionalPoints.Add(comboBox.SelectedItem.ToString());

pointCity.Add(comboBox.SelectedItem.ToString().Split(' ')[comboBox.SelectedItem.ToString().Split(' ').Length -1]);

}

else

{

CustomMessageBox.Show("Уведомление", $"Заполните все поля или удалите ненужные и повторите попытку.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

return;

}

}

}

}

}

additionalPoints.Add(lastPoint);

if (!string.IsNullOrEmpty(startPoint) && !string.IsNullOrEmpty(lastPoint))

{

foreach (var item in additionalPoints)

{

if (!string.IsNullOrEmpty(item.ToString()))

{

points += $"point={AllCity.FirstOrDefault(x => x.Item1 == item)?.Item2.Replace(' ', ',')}&"; // создание строки с координатами для api

}

else

{

CustomMessageBox.Show("Уведомление", $"Заполните все поля или удалите ненужные и повторите попытку.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

return;

}

}

}

else

{

CustomMessageBox.Show("Уведомление", $"Заполните все поля или удалите ненужные и повторите попытку.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

return;

}

var client = new HttpClient();

HttpResponseMessage request;

if (points.Split(',').Length == 3) // проверка на кол-во точек в маршруте

{

request = await client.GetAsync($"https://graphhopper.com/api/1/route?{points}profile=car&calc\_points=false&algorithm=alternative\_route&key=7bd83501-9f94-4196-bc57-9af92a81344d");

}

else

{

request = await client.GetAsync($"https://graphhopper.com/api/1/route?{points}profile=car&calc\_points=false&key=7bd83501-9f94-4196-bc57-9af92a81344d");

}

var responseContent = await request.Content.ReadAsStringAsync();

var jsonResponse = JObject.Parse(responseContent);

var paths = jsonResponse["paths"];

info = new List<(double distance, int time)>();

if(paths != null)

{

foreach (var path in paths)

{

var distanceInKm = Math.Round((double)path["distance"] / 1000, 2);

var timeInMinutes = (int)path["time"] / 60000;

info.Add((distanceInKm, timeInMinutes));

}

CreateTripItem(info);

CustomMessageBox.Show("Уведомление", $"Выберите маршрут в правой колонке для просмотра информации о выбранном маршруте.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

}

else

{

CustomMessageBox.Show("Уведомление", $"Данный машрут не найден. Выберите другие города и повторите попытку.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information); }

}

else

{

CustomMessageBox.Show("Уведомление", $"Заполните все поля и повторите попытку.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

}

}

else // ручно ввод данных

{

if (StartPoint.Text != null || FinishPoint.Text != null || Distance.Text != null)

{

additionalPoints = new List<string>();

string startPoint = StartPoint.Text;

string lastPoint = FinishPoint.Text;

Class.Route newRoute = new Class.Route();

db.InsertPoint(startPoint, "", IdUser);

db.InsertPoint(lastPoint, "", IdUser);

newRoute.Name = $"{startPoint} - {lastPoint}";

newRoute.PointOne = db.GetCityIdByName(IdUser,startPoint);

newRoute.PointTwo = db.GetCityIdByName(IdUser,lastPoint);

newRoute.AdditionPoint = "";

foreach (var item in Points.Children)

{

if (item is Grid)

{

Grid block = item as Grid;

foreach (var element in block.Children)

{

if (element is TextBox)

{

TextBox TextBox = element as TextBox;

if (TextBox.Text != null)

{

additionalPoints.Add(TextBox.Text.ToString());

}

else

{

CustomMessageBox.Show("Уведомление", $"Заполните все поля или удалите ненужные и повторите попытку.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

return;

}

}

}

}

}

newRoute.AdditionPoint = string.Join("#", pointCity);

double distance = 0;

if (!double.TryParse(Distance.Text, out distance) || distance <= 0)

{

CustomMessageBox.Show( "Предупреждение", "Пожалуйста, введите корректное расстояние.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

return;

}

newRoute.Distance = distance;

newRoute.TravelTime = Math.Round(distance / 100 \* 60,0).ToString();

newRoute.IdUser = IdUser;

if(db.InsertRoute(newRoute))

{

CustomMessageBox.Show("Уведомление", $"Маршрут {newRoute.Name} добавлен в систему.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

}

else

{

CustomMessageBox.Show("Уведомление", $"Маршрут {newRoute.Name} не добавлен в систему.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Warning);

}

}

else

{

CustomMessageBox.Show("Уведомление", $"Заполните все поля и повторите попытку.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

}

}

}

public void CreateTripItem(List<(double distance, int time)> values)

{

TripInfo.Children.Clear();

int count = 1;

foreach (var value in values)

{

Border border = new Border()

{

Margin = new Thickness(10),

Background = Brushes.Transparent,

BorderBrush = (Brush)FindResource("TextSecundaryColor"),

BorderThickness = new Thickness(1),

CornerRadius = new CornerRadius(5),

};

border.MouseLeftButtonDown += CreateTripClick;

Grid grid = new Grid();

grid.RowDefinitions.Add(new RowDefinition() { Height = GridLength.Auto });

grid.RowDefinitions.Add(new RowDefinition() { Height = new GridLength(1, GridUnitType.Star) });

grid.RowDefinitions.Add(new RowDefinition() { Height = GridLength.Auto });

TextBlock textBlock1 = new TextBlock()

{

Text = $"Маршрут {count}",

FontSize = 25,

Foreground = (Brush)FindResource("TextSecundaryColor"),

HorizontalAlignment = HorizontalAlignment.Center,

Margin = new Thickness(10)

};

Grid.SetRow(textBlock1, 0);

TimeSpan date = TimeSpan.FromMinutes(value.time);

if(date.Days >= 1)

{

if (date.Hours == 0)

{

TextBlock textBlock2 = new TextBlock()

{

Text = $"{date.Days} д {date.Minutes} мин",

FontSize = 20,

Foreground = (Brush)FindResource("TextSecundaryColor"),

HorizontalAlignment = HorizontalAlignment.Center,

Margin = new Thickness(10, 0, 10, 0)

};

Grid.SetRow(textBlock2, 1);

grid.Children.Add(textBlock2);

}

else

{

TextBlock textBlock2 = new TextBlock()

{

Text = $"{date.Days} д {date.Hours} ч {date.Minutes} мин",

FontSize = 20,

Foreground = (Brush)FindResource("TextSecundaryColor"),

HorizontalAlignment = HorizontalAlignment.Center,

Margin = new Thickness(10, 0, 10, 0)

};

Grid.SetRow(textBlock2, 1);

grid.Children.Add(textBlock2);

}

}

else if(date.Hours != 0)

{

TextBlock textBlock2 = new TextBlock()

{

Text = $"{date.Hours} ч {date.Minutes} мин",

FontSize = 20,

Foreground = (Brush)FindResource("TextSecundaryColor"),

HorizontalAlignment = HorizontalAlignment.Center,

Margin = new Thickness(10, 0, 10, 0)

};

Grid.SetRow(textBlock2, 1);

grid.Children.Add(textBlock2);

}

else

{

TextBlock textBlock2 = new TextBlock()

{

Text = $"{date.Minutes} мин",

FontSize = 20,

Foreground = (Brush)FindResource("TextSecundaryColor"),

HorizontalAlignment = HorizontalAlignment.Center,

Margin = new Thickness(10, 0, 10, 0)

};

Grid.SetRow(textBlock2, 1);

grid.Children.Add(textBlock2);

}

TextBlock textBlock3 = new TextBlock()

{

Text = $"{value.distance} км",

FontSize = 20,

Foreground = (Brush)FindResource("TextSecundaryColor"),

HorizontalAlignment = HorizontalAlignment.Center,

Margin = new Thickness(10, 5, 10, 10)

};

Grid.SetRow(textBlock3, 2);

grid.Children.Add(textBlock1);

grid.Children.Add(textBlock3);

border.Child = grid;

TripInfo.Children.Add(border);

count++;

}

}

private void CreateTripClick(object sender, MouseButtonEventArgs e)

{

bool flagOne = false;

bool flagTwo = false;

Border clickedBorder = sender as Border;

if (clickedBorder != null)

{

Grid grid = clickedBorder.Child as Grid;

if (grid != null && grid.Children.Count == 3)

{

TextBlock textBlock1 = grid.Children[0] as TextBlock;

TextBlock textBlock2 = grid.Children[1] as TextBlock;

TextBlock textBlock3 = grid.Children[2] as TextBlock;

if (textBlock1 != null && textBlock2 != null && textBlock3 != null)

{

string routeName = textBlock2.Text;

string distance = textBlock3.Text;

string time = textBlock1.Text;

var result = CustomMessageBox.Show("Уведомление", $"Вы выбрали маршрут:\nНазвание: {routeName}\nДистанция: {distance}\nВремя: {time}\nЖелаете сохранить данный маршрут в системе?", MessageBoxButton.YesNo, CustomMessageBox.MessageBoxImage.Information);

if (result == MessageBoxResult.Yes)

{

if (comboBoxStartPoint.SelectedIndex != -1 || comboBoxLastPointInfo.SelectedIndex !=1)

{

string startPoint = this.comboBoxStartPoint.SelectedItem?.ToString();

string lastPoint = this.comboBoxLastPointInfo.SelectedItem?.ToString();

if (startPoint.Contains("улица"))

{

string[] arr = this.comboBoxStartPoint.SelectedItem?.ToString().Split(',');

startPoint = arr[arr.Length - 2].Trim() + " " + arr[arr.Length - 1].Trim();

flagOne = true;

}

else

{

startPoint = this.comboBoxStartPoint.SelectedItem?.ToString().Split(' ')[comboBoxStartPoint.SelectedItem.ToString().Split(' ').Length - 1];

}

if (lastPoint.Contains("улица"))

{

string[] arr = this.comboBoxLastPointInfo.SelectedItem?.ToString().Split(',');

lastPoint = arr[arr.Length - 2].Trim() + " " + arr[arr.Length - 1].Trim();

flagTwo = true;

}

else

{

lastPoint = this.comboBoxLastPointInfo.SelectedItem?.ToString().Split(' ')[comboBoxLastPointInfo.SelectedItem.ToString().Split(' ').Length - 1];

}

string addPoint;

if (pointCity != null)

{

addPoint = string.Join("#", pointCity);

}

else

{

addPoint = null;

}

string coordinatesOne = "";

string coordinatesTwo = "";

foreach (var item in AllCity)

{

if(flagOne)

{

string[] arr = item.Item1.ToString().Split(',');

string path = arr[arr.Length - 2].Trim() + " " + arr[arr.Length - 1].Trim();

if (path.Trim() == startPoint.Trim())

{

coordinatesOne = item.Item2.Replace(' ', ',');

}

}

else

{

string path = item.Item1.Split(',')[item.Item1.Split(',').Length - 1];

if (path.Trim() == startPoint.Trim())

{

coordinatesOne = item.Item2.Replace(' ', ',');

}

}

if (flagTwo)

{

string[] arr = item.Item1.ToString().Split(',');

string path = arr[arr.Length - 2].Trim() + " " + arr[arr.Length - 1].Trim();

if (path.Trim() == lastPoint.Trim())

{

coordinatesTwo = item.Item2.Replace(' ', ',');

}

}

else

{

string path = item.Item1.Split(',')[item.Item1.Split(',').Length - 1];

if (path.Trim() == lastPoint.Trim())

{

coordinatesTwo = item.Item2.Replace(' ', ',');

}

}

}

db.InsertPoint(startPoint, coordinatesOne, IdUser);

db.InsertPoint(lastPoint, coordinatesTwo, IdUser);

Class.Route route = new Class.Route();

route.Name = $"{startPoint} - {lastPoint}";

route.PointOne = db.GetCityIdByName(IdUser, startPoint);

route.PointTwo = db.GetCityIdByName(IdUser, lastPoint);

route.AdditionPoint = addPoint;

route.Distance = Convert.ToDouble(distance.Split(' ')[0]);

int timeInt = info[Convert.ToInt32(routeName.Split(' ')[1]) -1].time;

route.TravelTime = timeInt.ToString();

route.IdUser = IdUser;

selectRoute = route;

if (db.InsertRoute(route))

{

CustomMessageBox.Show("Уведомление", $"Машрут {route.Name} сохранен в систему.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

}

else

{

CustomMessageBox.Show("Уведомление", $"Машрут {route.Name} не был сохранен в систему, так как уже есть в базе данных.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

}

}

}

}

}

}

}

private async void Fill\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

Points.Children.Clear();

bool internet = await CheckInternetConnectionAsync();

if (Fill.SelectedIndex == 1)

{

if(internet)

{

comboBoxStartPoint.Visibility = Visibility.Visible;

comboBoxLastPointInfo.Visibility = Visibility.Visible;

Distance.Visibility = Visibility.Collapsed;

fill = 1;

}

else

{

await Application.Current.Dispatcher.InvokeAsync(() =>

{

CustomMessageBox.Show("Уведомление", $"Проверьте интернет соединение и повторите попытку позже.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

});

Fill.SelectedIndex = 0;

}

}

else

{

comboBoxStartPoint.Visibility = Visibility.Collapsed;

comboBoxLastPointInfo.Visibility = Visibility.Collapsed;

Distance.Visibility = Visibility.Visible;

fill = 0;

}

}

public async Task<bool> CheckInternetConnectionAsync()

{

try

{

Ping ping = new Ping();

PingReply reply = await ping.SendPingAsync("www.google.com");

return reply.Status == IPStatus.Success;

}

catch (PingException)

{

return false;

}

}

}

}<Page x:Class="OneWay.Pages.AddTripPage"

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:iconPacks="http://metro.mahapps.com/winfx/xaml/iconpacks"

xmlns:local="clr-namespace:OneWay.Pages"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800"

Title="Trip">

<Border CornerRadius="5" Background="{DynamicResource SecundaryBackgroundColor}" Loaded="Border\_Loaded">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="3.5\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<TextBlock Text="Создание поездки" Grid.Column="0" FontSize ="25" Foreground="{DynamicResource TextSecundaryColor}" Margin="10,10,10,10"/>

<Grid Grid.Column="0" Margin="0 50 0 0" >

<Grid.RowDefinitions>

<RowDefinition Height="50"/>

<RowDefinition Height="50"/>

<RowDefinition Height="50"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="50"/>

<RowDefinition Height="50"/>

<RowDefinition Height="50"/>

</Grid.RowDefinitions>

<Grid Grid.Row="0">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="165"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<TextBlock Grid.Column="0" Text="Автомобиль" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10,10,10,10"/>

<ComboBox Grid.Column="1" x:Name="comboBoxCar" FontSize="18" Margin="10,12,10,11" SelectionChanged="comboBoxCar\_SelectionChanged" />

</Grid>

<Grid Grid.Row="1">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="165"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<TextBlock Grid.Column="0" Text="Пассажиров" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10,10,10,10"/>

<TextBox Grid.Column="1" x:Name="People" Tag="Число человек" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="10,12,10,11" PreviewTextInput="People\_PreviewTextInput"/>

<ComboBox Grid.Column="2" x:Name="FuelOctane" FontSize="18" Margin="10,12,10,11" Visibility="Hidden"/>

</Grid>

<Grid Grid.Row="2">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="auto"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<TextBlock Grid.Column="0" Text="Точка отправки" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10,10,10,10"/>

<ComboBox Grid.Column="1" x:Name="comboBoxStartPoint" FontSize="18" Margin="10,12,10,11" SelectionChanged="comboBoxStartPoint\_SelectionChanged"/>

</Grid>

<ScrollViewer Grid.Row="3" Margin="10,0,0,0">

<StackPanel x:Name="Points">

</StackPanel>

</ScrollViewer>

<Grid Grid.Row="4">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="auto"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<TextBlock Grid.Column="0" Text="Точка прибытия" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10,10,10,10"/>

<ComboBox Grid.Column="1" x:Name="comboBoxLastPoint" FontSize="18" Margin="10,12,10,11" SelectionChanged="comboBoxLastPointInfo\_SelectionChanged"/>

</Grid>

<Grid Grid.Row="5">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="auto"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="auto"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<TextBlock Grid.Column="0" Text="Стоимость поездки, руб" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10,10,10,10"/>

<TextBox Grid.Column="1" x:Name="Price" TextAlignment="Right" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="10,12,10,11" IsReadOnly="True"/>

<TextBlock Grid.Column="2" x:Name="TextFuelUse" Text="Затрачено топлива, л" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10,10,10,10"/>

<TextBox Grid.Column="3" x:Name="FuelUse" TextAlignment="Right" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="10,12,10,11" IsReadOnly="True"/>

</Grid>

<Button Grid.Row="6" x:Name="CreateRouteButton" Content="Сохранить поездку" FontSize="20" Height="30" Margin="10,10,355,10" Click="CreateRouteButton\_Click">

<Button.Template>

<ControlTemplate TargetType="Button">

<Grid>

<Border Background="{DynamicResource TextSecundaryColor}" CornerRadius="10">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

</Grid>

</ControlTemplate>

</Button.Template>

</Button>

</Grid>

<Grid Grid.Column="1">

<Grid.RowDefinitions>

<RowDefinition Height="\*"/>

<RowDefinition Height="3\*"/>

</Grid.RowDefinitions>

</Grid>

<StackPanel Grid.Column="1"/>

<StackPanel Grid.Column="1" x:Name="TripInfo" />

</Grid>

</Border>

</Page>

using System;

using System.Collections.Generic;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Input;

using System.Windows.Media;

using OneWay.Controls;

using OneWay.Class;

using System.Data;

using System.Linq;

namespace OneWay.Pages

{

public partial class AddTripPage : Page

{

public int IdUser;

DataBase db = new DataBase("OneWayDB.db");

public List<(int id, string octaneNumber)> octaneNumber;

List<Car> cars = new List<Car> { };

List<Route> routes = new List<Route> { };

List<Points> points = new List<Points> { };

public static List<Tuple<string, string>> AllCity = new List<Tuple<string, string>>();

public List<string> number = new List<string>();

private Trip selectTrip = null;

public AddTripPage(int IdUser)

{

InitializeComponent();

this.IdUser = IdUser;

routes = db.GetAllRoutes(IdUser);

points = db.GetAllPoints(IdUser);

comboBoxStartPoint.Items.Clear();

foreach (var item in points)

{

comboBoxStartPoint.Items.Add(item.Name);

comboBoxLastPoint.Items.Add(item.Name);

}

}

private void Border\_Loaded(object sender, RoutedEventArgs e)

{

comboBoxCar.Items.Clear();

cars = db.GetAllCars(IdUser);

foreach (Car car in cars)

{

ComboBoxItem item = new ComboBoxItem();

item.Content = $"{car.Brand} {car.Model} {car.Equipment}";

if (db.CheckFavoriteCar(IdUser, car.IdCar) == 1)

{

item.Content += " ⭐";

item.Foreground = new SolidColorBrush(Colors.Gold);

}

comboBoxCar.Items.Add(item);

}

}

private void AddPoint(string point)

{

Points.Children.Clear();

foreach (var item in point.Split('#'))

{

Grid grid = new Grid();

grid.ColumnDefinitions.Add(new ColumnDefinition() { Width = GridLength.Auto });

grid.ColumnDefinitions.Add(new ColumnDefinition() { Width = new GridLength(1, GridUnitType.Star) });

TextBlock textCount = new TextBlock()

{

Text = "Точка остановки",

FontSize = 20,

Foreground = (Brush)FindResource("TextSecundaryColor"),

TextAlignment = TextAlignment.Left,

HorizontalAlignment = HorizontalAlignment.Left,

VerticalAlignment = VerticalAlignment.Center,

};

Grid.SetColumn(textCount, 0);

TextBox newTextBox = new TextBox()

{

Height = 26.6,

Tag = "Город",

Text = item.ToString(),

FontSize = 20,

Foreground = (Brush)FindResource("TextSecundaryColor"),

Background = Brushes.Transparent,

BorderThickness = new Thickness(0, 0, 0, 1),

Margin = new Thickness(5, 0, 5, 0),

VerticalAlignment = VerticalAlignment.Center,

IsReadOnly = true,

};

Grid.SetColumn(newTextBox, 1);

grid.Children.Add(textCount);

grid.Children.Add(newTextBox);

Points.Children.Add(grid);

}

}

void CreateTripItem(List<Route> values)

{

number.Clear();

TripInfo.Children.Clear();

int count = 1;

foreach (var value in values)

{

string name = value.IdRoute.ToString();

number.Add(name);

Border border = new Border()

{

Margin = new Thickness(10),

Background = Brushes.Transparent,

BorderBrush = (Brush)FindResource("TextSecundaryColor"),

BorderThickness = new Thickness(1),

CornerRadius = new CornerRadius(5),

};

border.MouseLeftButtonDown += CreateTripClick;

Grid grid = new Grid();

grid.RowDefinitions.Add(new RowDefinition() { Height = GridLength.Auto });

grid.RowDefinitions.Add(new RowDefinition() { Height = new GridLength(1, GridUnitType.Star) });

grid.RowDefinitions.Add(new RowDefinition() { Height = GridLength.Auto });

TextBlock textBlock1 = new TextBlock()

{

Text = $"Маршрут {count}",

FontSize = 25,

Foreground = (Brush)FindResource("TextSecundaryColor"),

HorizontalAlignment = HorizontalAlignment.Center,

Margin = new Thickness(10)

};

if(db.CheckFavoriteRoute(IdUser, value.IdRoute) == 1)

{

textBlock1.Foreground = new SolidColorBrush(Colors.Gold);

}

Grid.SetRow(textBlock1, 0);

TimeSpan date = TimeSpan.FromMinutes(Convert.ToDouble(value.TravelTime));

if (date.Days >= 1)

{

if (date.Hours == 0)

{

TextBlock textBlock2 = new TextBlock()

{

Text = $"{date.Days} д {date.Minutes} мин",

FontSize = 20,

Foreground = (Brush)FindResource("TextSecundaryColor"),

HorizontalAlignment = HorizontalAlignment.Center,

Margin = new Thickness(10, 0, 10, 0)

};

Grid.SetRow(textBlock2, 1);

grid.Children.Add(textBlock2);

}

else

{

TextBlock textBlock2 = new TextBlock()

{

Text = $"{date.Days} д {date.Hours} ч {date.Minutes} мин",

FontSize = 20,

Foreground = (Brush)FindResource("TextSecundaryColor"),

HorizontalAlignment = HorizontalAlignment.Center,

Margin = new Thickness(10, 0, 10, 0)

};

Grid.SetRow(textBlock2, 1);

grid.Children.Add(textBlock2);

}

}

else if (date.Hours != 0)

{

TextBlock textBlock2 = new TextBlock()

{

Text = $"{date.Hours} ч {date.Minutes} мин",

FontSize = 20,

Foreground = (Brush)FindResource("TextSecundaryColor"),

HorizontalAlignment = HorizontalAlignment.Center,

Margin = new Thickness(10, 0, 10, 0)

};

Grid.SetRow(textBlock2, 1);

grid.Children.Add(textBlock2);

}

else

{

TextBlock textBlock2 = new TextBlock()

{

Text = $"{date.Minutes} мин",

FontSize = 20,

Foreground = (Brush)FindResource("TextSecundaryColor"),

HorizontalAlignment = HorizontalAlignment.Center,

Margin = new Thickness(10, 0, 10, 0)

};

Grid.SetRow(textBlock2, 1);

grid.Children.Add(textBlock2);

}

TextBlock textBlock3 = new TextBlock()

{

Text = $"{value.Distance} км",

FontSize = 20,

Foreground = (Brush)FindResource("TextSecundaryColor"),

HorizontalAlignment = HorizontalAlignment.Center,

Margin = new Thickness(10, 5, 10, 10)

};

Grid.SetRow(textBlock3, 2);

grid.Children.Add(textBlock1);

grid.Children.Add(textBlock3);

border.Child = grid;

TripInfo.Children.Add(border);

count++;

}

}

private void CreateRouteButton\_Click(object sender, RoutedEventArgs e)

{

if(selectTrip != null ) // проверка выбрана ли поездка

{

if (comboBoxCar.SelectedIndex != -1 )

{

Car selectedCar = null;

string selectItemCar = (comboBoxCar.SelectedItem as ComboBoxItem)?.Content.ToString();

if (selectItemCar.Contains(" ⭐"))

{

selectItemCar = selectItemCar.Substring(0, selectItemCar.Length - 2);

}

foreach (var item in cars)

{

if ($"{item.Brand} {item.Model} {item.Equipment}" == selectItemCar)

{

selectedCar = item;

break;

}

}

if (selectedCar != null) // выбран ли автомобиль

{

if (string.IsNullOrWhiteSpace(People.Text))

{

CustomMessageBox.Show("Уведомление", $"Введите число пассажиров.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

return;

}

int numberOfPeople;

if (!int.TryParse(People.Text, out numberOfPeople))

{

CustomMessageBox.Show("Уведомление", $"Введите корректное число пассажиров.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

return;

}

if (numberOfPeople > selectedCar.Seats)

{

CustomMessageBox.Show("Уведомление", $"Число пассажиров больше, чем число мест в машине.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

return;

}

selectTrip.IdCar = selectedCar.IdCar;

selectTrip.People = numberOfPeople;

double distance = db.GetDistance(selectTrip.IdRoute);

double usedFuel;

double usedMoney;

if (selectedCar.Fuel == "Электричество")

{

usedFuel = Math.Round(distance \* selectedCar.FuelConsumption, 2);

int fuelId = octaneNumber[FuelOctane.SelectedIndex].id;

usedMoney = Math.Round(usedFuel \* db.GetFuelLastPrice(fuelId), 2);

TextFuelUse.Text = "Затрачено кВт/ч";

}

else if (selectedCar.Fuel == "Бензин")

{

usedFuel = Math.Round((distance / 100) \* selectedCar.FuelConsumption, 2);

usedMoney = Math.Round(usedFuel \* db.GetFuelLastPrice(selectedCar.FuelOctane), 2);

TextFuelUse.Text = "Затрачено топлива, л";

}

else if (selectedCar.Fuel == "Газ")

{

usedFuel = Math.Round((distance / 100) \* selectedCar.FuelConsumption, 2);

usedMoney = Math.Round(usedFuel \* db.GetFuelLastPrice(9), 2);

TextFuelUse.Text = "Затрачено топлива, л";

}

else

{

if (FuelOctane.SelectedIndex != -1)

{

usedFuel = Math.Round((distance / 100) \* selectedCar.FuelConsumption, 2);

int fuelId = octaneNumber[FuelOctane.SelectedIndex].id;

usedMoney = Math.Round(usedFuel \* db.GetFuelLastPrice(fuelId), 2);

TextFuelUse.Text = "Затрачено топлива, л";

}

else

{

CustomMessageBox.Show("Уведомление", $"Выберите тип топлива", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

return;

}

}

selectTrip.UsedFuel = usedFuel;

selectTrip.UsedMoney = usedMoney;

selectTrip.StartTime = DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss");

int minutes = Convert.ToInt32(db.GetTime(selectTrip.IdRoute));

DateTime finishTime = DateTime.Now.AddMinutes(minutes);

selectTrip.FinishTime = finishTime.ToString("yyyy-MM-dd HH:mm:ss");

FuelUse.Text = selectTrip.UsedFuel.ToString();

FuelUse.BorderBrush = Brushes.Green;

FuelUse.BorderThickness = new Thickness(0, 0, 0, 2);

Price.BorderBrush = Brushes.Green;

Price.BorderThickness = new Thickness(0, 0, 0, 2);

Price.Text = selectTrip.UsedMoney.ToString();

var message = CustomMessageBox.Show("Уведомление", $"Сохранить поездку на автомобиле {comboBoxCar.SelectedItem.ToString()} по машруту {comboBoxStartPoint.SelectedItem} - {comboBoxLastPoint.SelectedItem}", MessageBoxButton.YesNo, CustomMessageBox.MessageBoxImage.Information);

if(message == MessageBoxResult.Yes)

{

if (db.InsertTrip(selectTrip)) // добавление поездки в бд

{

CustomMessageBox.Show("Уведомление", $"Поездка сохранена в систему.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

}

}

}

else

{

CustomMessageBox.Show("Уведомление", $"Выберите автомобиль из списка.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

}

}

else

{

CustomMessageBox.Show("Уведомление", $"Выберите автомобиль из списка.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

}

}

else

{

CustomMessageBox.Show("Уведомление", $"Выберите маршрут.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

}

}

private void CreateTripClick(object sender, MouseButtonEventArgs e)

{

FuelUse.Text = "";

Price.Text = "";

FuelUse.BorderBrush = Brushes.Gray;

FuelUse.BorderThickness = new Thickness(0);

Price.BorderBrush = Brushes.Gray;

Price.BorderThickness = new Thickness(0);

foreach (Border border in TripInfo.Children)

{

border.BorderBrush = Brushes.Gray;

}

Border clickedBorder = sender as Border;

if (clickedBorder != null)

{

clickedBorder.BorderBrush = Brushes.Green;

Grid grid = clickedBorder.Child as Grid;

if (grid != null && grid.Children.Count == 3)

{

TextBlock textBlock1 = grid.Children[0] as TextBlock;

TextBlock textBlock2 = grid.Children[1] as TextBlock;

TextBlock textBlock3 = grid.Children[2] as TextBlock;

if (textBlock1 != null && textBlock2 != null && textBlock3 != null)

{

string routeName = textBlock2.Text;

string distance = textBlock3.Text;

string time = textBlock1.Text;

var result = CustomMessageBox.Show("Уведомление", $"Вы выбрали маршрут:\nНазвание: {routeName}\nДистанция: {distance}\nВремя: {time}\nЖелаете просмотреть информацию о машруте?", MessageBoxButton.YesNo, CustomMessageBox.MessageBoxImage.Information);

if (result == MessageBoxResult.Yes)

{

Points.Children.Clear();

Trip newTrip = new Trip();

newTrip.IdUser = IdUser;

int id = Convert.ToInt32(routeName.Split(' ')[1].Trim()) - 1;

newTrip.IdRoute = Convert.ToInt32(number[id]);

string points = "";

foreach (var item in routes)

{

if (newTrip.IdRoute == item.IdRoute)

{

points = item.AdditionPoint;

}

}

if (points != "") // проверка на наличие точек остановки

{

AddPoint(points);

}

if (comboBoxCar.SelectedIndex != -1)

{

Car selectedCar = null;

string selectItemCar = (comboBoxCar.SelectedItem as ComboBoxItem)?.Content.ToString();

if (selectItemCar.Contains(" ⭐"))

{

selectItemCar = selectItemCar.Substring(0, selectItemCar.Length - 2);

}

foreach (var item in cars)

{

if ($"{item.Brand} {item.Model} {item.Equipment}" == selectItemCar)

{

selectedCar = item;

break;

}

}

if (selectedCar != null)

{

selectTrip = newTrip;

selectTrip.IdCar = selectedCar.IdCar;

double distanceTwo = db.GetDistance(newTrip.IdRoute);

double usedFuel;

double usedMoney;

if (selectedCar.Fuel == "Электричество")

{

usedFuel = Math.Round(distanceTwo \* selectedCar.FuelConsumption, 2);

int fuelId = octaneNumber[FuelOctane.SelectedIndex].id;

usedMoney = Math.Round(usedFuel \* db.GetFuelLastPrice(fuelId), 2);

TextFuelUse.Text = "Затрачено кВт/ч";

}

else if(selectedCar.Fuel == "Бензин")

{

usedFuel = Math.Round((distanceTwo / 100) \* selectedCar.FuelConsumption, 2);

usedMoney = Math.Round(usedFuel \* db.GetFuelLastPrice(selectedCar.FuelOctane), 2);

TextFuelUse.Text = "Затрачено топлива, л";

}

else if (selectedCar.Fuel == "Газ")

{

usedFuel = Math.Round((distanceTwo / 100) \* selectedCar.FuelConsumption, 2);

usedMoney = Math.Round(usedFuel \* db.GetFuelLastPrice(9), 2);

TextFuelUse.Text = "Затрачено топлива, л";

}

else

{

if (FuelOctane.SelectedIndex != -1)

{

usedFuel = Math.Round((distanceTwo / 100) \* selectedCar.FuelConsumption,2);

int fuelId = octaneNumber[FuelOctane.SelectedIndex].id;

usedMoney = Math.Round(usedFuel \* db.GetFuelLastPrice(fuelId), 2);

TextFuelUse.Text = "Затрачено топлива, л";

}

else

{

CustomMessageBox.Show("Уведомление", $"Выберите тип топлива", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

return;

}

}

FuelUse.Text = usedFuel.ToString();

FuelUse.BorderBrush = Brushes.Green;

FuelUse.BorderThickness = new Thickness(0, 0, 0, 2);

Price.BorderBrush = Brushes.Green;

Price.BorderThickness = new Thickness(0, 0, 0, 2);

Price.Text = usedMoney.ToString();

}

else

{

CustomMessageBox.Show("Уведомление", $"Для отображения информации выберите автомобиль из списка и повторите попытку.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

}

}

else

{

CustomMessageBox.Show("Уведомление", $"Для отображения информации выберите автомобиль из списка и повторите попытку.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

clickedBorder.BorderBrush = Brushes.Gray;

}

}

else

{

clickedBorder.BorderBrush = Brushes.Gray;

}

}

}

}

}

private void People\_PreviewTextInput(object sender, TextCompositionEventArgs e)

{

if (!char.IsDigit(e.Text, e.Text.Length - 1))

{

e.Handled = true;

return;

}

string currentText = People.Text.Insert(People.SelectionStart, e.Text);

if (!int.TryParse(currentText, out int seat))

{

e.Handled = true;

return;

}

if (seat <= 0)

{

CustomMessageBox.Show("Уведомление", $"Число пассажиров должно быть больше 0", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

e.Handled = true;

}

else if (seat > 10)

{

CustomMessageBox.Show("Уведомление", $"Число пассажиров должно быть меньше или равно 10", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

e.Handled = true;

}

}

private void comboBoxCar\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

FuelOctane.Items.Clear();

Car car = cars[comboBoxCar.SelectedIndex];

if (car.Fuel == "Электричество")

{

FuelOctane.Visibility = Visibility.Visible;

octaneNumber = db.GetOctaneNumber("Электричество");

}

else if (car.Fuel == "Дизельное топливо")

{

FuelOctane.Visibility = Visibility.Visible;

octaneNumber = db.GetOctaneNumber("Дизельное топливо");

}

else

{

FuelOctane.Visibility = Visibility.Collapsed;

return;

}

foreach (var item in octaneNumber)

{

FuelOctane.Items.Add(item.octaneNumber);

}

FuelOctane.SelectedIndex = 0;

}

private void comboBoxStartPoint\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

if (comboBoxStartPoint.SelectedIndex != -1 && comboBoxLastPoint.SelectedIndex != -1)

{

selectTrip = null;

string startPoint = this.comboBoxStartPoint.SelectedItem?.ToString();

string lastPoint = this.comboBoxLastPoint.SelectedItem?.ToString();

List<Route> selectRoute = db.SelectRoute($"{startPoint} - {lastPoint}", IdUser);

if (selectRoute.Count > 0)

{

CreateTripItem(selectRoute);

}

else

{

CustomMessageBox.Show("Уведомление", $"Маршрут не найден.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

number.Clear();

TripInfo.Children.Clear();

}

}

if(comboBoxLastPoint.SelectedIndex == -1)

{

List<Route> allRoute = db.GetAllRoutes(IdUser);

List<Route> selectRoute = new List<Route>();

List<Points> FinishPoints = new List<Points>();

string startPoint = this.comboBoxStartPoint.SelectedItem?.ToString();

int index = db.GetCityIdByName(IdUser, startPoint);

foreach (var item in allRoute)

{

if(item.PointOne == index)

{

selectRoute.Add(item);

}

}

comboBoxLastPoint.Items.Clear();

foreach(var route in selectRoute)

{

int q = (int)route.PointTwo;

foreach(var item in points)

{

if(item.IdPoint == q)

{

FinishPoints.Add(item);

break;

}

}

}

List<Points> filterPoint = FinishPoints.GroupBy(p => p.Name)

.Select(g => g.First())

.ToList();

foreach (var item in filterPoint)

{

comboBoxLastPoint.Items.Add(item.Name);

}

}

else

{

comboBoxLastPoint.Items.Clear();

}

}

private void comboBoxLastPointInfo\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

if(comboBoxStartPoint.SelectedIndex != -1 && comboBoxLastPoint.SelectedIndex != -1)

{

selectTrip = null;

string startPoint = this.comboBoxStartPoint.SelectedItem?.ToString();

string lastPoint = this.comboBoxLastPoint.SelectedItem?.ToString();

List<Route> selectRoute = db.SelectRoute($"{startPoint} - {lastPoint}",IdUser);

if(selectRoute.Count > 0)

{

CreateTripItem(selectRoute);

}

else

{

CustomMessageBox.Show("Уведомление", $"Маршрут не найден.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

number.Clear();

TripInfo.Children.Clear();

}

}

}

}

}

<Page x:Class="OneWay.Pages.AllCarsPage"

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:iconPacks="http://metro.mahapps.com/winfx/xaml/iconpacks"

xmlns:ctls="clr-namespace:OneWay.Controls"

xmlns:local="clr-namespace:OneWay.Pages"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800"

Title="Cars" Loaded="Page\_Loaded">

<Page.Resources>

<Style TargetType="TextBox" BasedOn="{StaticResource {x:Type TextBox}}">

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="{x:Type TextBox}">

<Grid>

<Border Background="{TemplateBinding Background}"

BorderBrush="{TemplateBinding BorderBrush}"

BorderThickness="{TemplateBinding BorderThickness}"/>

<TextBlock x:Name="Placeholder"

Text="{TemplateBinding Tag}"

Foreground="Gray"

Visibility="Collapsed"/>

<ScrollViewer x:Name="PART\_ContentHost"

Margin="{TemplateBinding Padding}"/>

</Grid>

<ControlTemplate.Triggers>

<MultiTrigger>

<MultiTrigger.Conditions>

<Condition Property="Text" Value=""/>

</MultiTrigger.Conditions>

<Setter TargetName="Placeholder" Property="Visibility" Value="Visible"/>

</MultiTrigger>

</ControlTemplate.Triggers>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

<Style x:Key="ScrollBarTrackThumb"

TargetType="{x:Type Thumb}">

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="{x:Type Thumb}">

<Grid x:Name="Grid">

<Rectangle HorizontalAlignment="Stretch"

VerticalAlignment="Stretch"

Width="Auto"

Height="Auto"

Fill="Transparent" />

<Border x:Name="CornerScrollBarRectangle"

CornerRadius="5"

HorizontalAlignment="Stretch"

VerticalAlignment="Stretch"

Width="Auto"

Height="Auto"

Margin="0,1,0,1"

Background="{TemplateBinding Background}" />

</Grid>

<ControlTemplate.Triggers>

<Trigger Property="Tag"

Value="Horizontal">

<Setter TargetName="CornerScrollBarRectangle"

Property="Width"

Value="Auto" />

<Setter TargetName="CornerScrollBarRectangle"

Property="Height"

Value="6" />

</Trigger>

</ControlTemplate.Triggers>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

<Style x:Key="CustomCheckBoxStyle" TargetType="CheckBox">

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="CheckBox">

<Grid Background="Transparent" Margin="5 0 0 5">

<Ellipse x:Name="CheckBoxEllipse" Fill="#B0B7C3" StrokeThickness="1" Stroke="#B0B7C3" Height="15" Width="15" HorizontalAlignment="Left" VerticalAlignment="Center"/>

<Ellipse x:Name="CheckboxCheckMark" Fill="#1d2335" Margin ="2.5 0 0 0" Height="10" Width="10" HorizontalAlignment="Left" VerticalAlignment="Center"/>

<ContentPresenter HorizontalAlignment="Center" Margin="30 0 0 0" VerticalAlignment="Center" Width="210"/>

</Grid>

<ControlTemplate.Triggers>

<Trigger Property="IsChecked" Value="True">

<Setter TargetName="CheckboxCheckMark" Property="Visibility" Value="Visible"/>

</Trigger>

<Trigger Property="IsChecked" Value="False">

<Setter TargetName="CheckboxCheckMark" Property="Visibility" Value="Collapsed"/>

</Trigger>

</ControlTemplate.Triggers>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

</Page.Resources>

<Border CornerRadius="5" Background="{DynamicResource SecundaryBackgroundColor}">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="auto"/>

</Grid.ColumnDefinitions>

<TextBlock Text="Мои Автомобили" Grid.Column="0" FontSize ="25" Foreground="{DynamicResource TextSecundaryColor}" Margin="10,10,10,10"/>

<ScrollViewer Grid.Row="0" Margin="0 50 0 0">

<WrapPanel x:Name="CarPanel">

<!--<ctls:CarItem Id="1" CarName="Mercedes-Benz" Model="SLR McLaren" Volume="5.4 / Бензин" Transmission="АКПП" Body="Купе" Seat="2" FavoriteCar="0" FavoriteButtonClicked="FavoriteButton\_Click"/>-->

<!--<ctls:CarItem Id="2" CarName="Mercedes-Benz" Model="SLR McLaren" Volume="5.4 / Бензин" Transmission="АКПП" Body="Купе" Seat="2"/>-->

</WrapPanel>

</ScrollViewer>

<Grid Grid.Column="1">

<Grid.RowDefinitions>

<RowDefinition Height="auto"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<Grid Grid.Row="0">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<TextBlock Text="Фильтры" Grid.Column="0" FontSize ="25" Foreground="{DynamicResource TextSecundaryColor}" Margin="5 10 10 10"/>

<Button Grid.Column="1" Margin="10" BorderThickness="0" Click="AcceptFilter\_Click" Width="42" Height="34">

<Button.Template>

<ControlTemplate TargetType="Button">

<Grid>

<Border Background="#006400" CornerRadius="10">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

<iconPacks:PackIconUnicons Kind="Search" Foreground="{DynamicResource TextSecundaryColor}" HorizontalAlignment="Center" VerticalAlignment="Center" />

</Grid>

</ControlTemplate>

</Button.Template>

</Button>

<Button x:Name="ClearButton" Grid.Column="2" Margin="10" BorderThickness="0" Width="42" Height="34" Click="ClearButton\_Click">

<Button.Template>

<ControlTemplate TargetType="Button">

<Grid>

<Border Background="#FFCA1E1E" CornerRadius="10">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

<iconPacks:PackIconUnicons Kind="Times" Foreground="{DynamicResource TextSecundaryColor}" HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Grid>

</ControlTemplate>

</Button.Template>

</Button>

</Grid>

<ScrollViewer Grid.Row="1">

<StackPanel x:Name="FilterPanel">

<TextBlock Text="Марка" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="5 0 0 0"/>

<TextBox x:Name="CarBrand" Tag="Марка" FontSize="15" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="5 0 0 0"/>

<TextBlock Text="Модель" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="5 0 0 0"/>

<TextBox x:Name="CarModel" Tag="Модель" FontSize="15" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="5 0 0 0"/>

<TextBlock Text="Количество мест" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="5 0 0 0"/>

<StackPanel Orientation="Horizontal">

<TextBox x:Name="SeatFrom" Tag="c" FontSize="15" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="5 0 0 0" Width="100" PreviewTextInput="SeatFrom\_PreviewTextInput" />

<TextBlock Text="-" FontSize ="15" Foreground="Gray" HorizontalAlignment="Center"/>

<TextBox x:Name="SeatTo" Tag="до" FontSize="15" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="5 0 0 0" Width="100" PreviewTextInput="SeatTo\_PreviewTextInput"/>

</StackPanel>

<TextBlock Text="Тип двигателя" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="5 0 0 0"/>

<StackPanel>

<CheckBox x:Name="Gasoline" Style="{StaticResource CustomCheckBoxStyle}">

<TextBlock Text="Бензиновый" Grid.Column="0" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" HorizontalAlignment="Left"/>

</CheckBox>

<CheckBox x:Name="Diesel" Style="{StaticResource CustomCheckBoxStyle}">

<TextBlock Text="Дизельный" Grid.Column="0" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" HorizontalAlignment="Left"/>

</CheckBox>

<CheckBox x:Name="Electro" Style="{StaticResource CustomCheckBoxStyle}">

<TextBlock Text="Электрический" Grid.Column="0" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" HorizontalAlignment="Left"/>

</CheckBox>

<CheckBox x:Name="Gas" Style="{StaticResource CustomCheckBoxStyle}">

<TextBlock Text="Гибридный" Grid.Column="0" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" HorizontalAlignment="Left"/>

</CheckBox>

</StackPanel>

<TextBlock Text="Коробка передач" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="5 0 0 0"/>

<StackPanel>

<CheckBox x:Name="Auto" Style="{StaticResource CustomCheckBoxStyle}">

<TextBlock Text="Автоматическая" Grid.Column="0" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" HorizontalAlignment="Left"/>

</CheckBox>

<CheckBox x:Name="Mech" Style="{StaticResource CustomCheckBoxStyle}">

<TextBlock Text="Механическая" Grid.Column="0" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" HorizontalAlignment="Left"/>

</CheckBox>

</StackPanel>

<TextBlock Text="Тип кузова" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="5 0 0 0"/>

<StackPanel>

<CheckBox x:Name="Sedan" Style="{StaticResource CustomCheckBoxStyle}">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="2\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<TextBlock Text="Седан" Grid.Column="0" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" HorizontalAlignment="Left"/>

<Image Source="/Resources/Body/sedan.png" Grid.Column="1" Height="25" HorizontalAlignment="Right"/>

</Grid>

</CheckBox>

<CheckBox x:Name="Jeep" Style="{StaticResource CustomCheckBoxStyle}">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="2\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<TextBlock Text="Внедорожник" Grid.Column="0" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" HorizontalAlignment="Left"/>

<Image Source="/Resources/Body/suv.png" Grid.Column="1" Height="25" HorizontalAlignment="Right"/>

</Grid>

</CheckBox>

<CheckBox x:Name="Universal" Style="{StaticResource CustomCheckBoxStyle}">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="2\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<TextBlock Text="Универсал" Grid.Column="0" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" HorizontalAlignment="Left"/>

<Image Source="/Resources/Body/universal.png" Grid.Column="1" Height="25" HorizontalAlignment="Right"/>

</Grid>

</CheckBox>

<CheckBox x:Name="Hatchback" Style="{StaticResource CustomCheckBoxStyle}">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="2\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<TextBlock Text="Хетчбэк" Grid.Column="0" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" HorizontalAlignment="Left"/>

<Image Source="/Resources/Body/hatchback.png" Grid.Column="1" Height="25" HorizontalAlignment="Right"/>

</Grid>

</CheckBox>

<CheckBox x:Name="Liftback" Style="{StaticResource CustomCheckBoxStyle}">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="2\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<TextBlock Text="Лифтбэк" Grid.Column="0" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" HorizontalAlignment="Left"/>

<Image Source="/Resources/Body/liftback.png" Grid.Column="1" Height="25" HorizontalAlignment="Right"/>

</Grid>

</CheckBox>

<CheckBox x:Name="Minivan" Style="{StaticResource CustomCheckBoxStyle}">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="2\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<TextBlock Text="Минивэн" Grid.Column="0" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" HorizontalAlignment="Left"/>

<Image Source="/Resources/Body/minivan.png" Grid.Column="1" Height="25" HorizontalAlignment="Right"/>

</Grid>

</CheckBox>

<!--<CheckBox x:Name="Minibus" Style="{StaticResource CustomCheckBoxStyle}">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="2\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<TextBlock Text="Микроавтобус" Grid.Column="0" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" HorizontalAlignment="Left"/>

<Image Source="/Resources/Body/minibus.png" Grid.Column="1" Height="25" HorizontalAlignment="Right"/>

</Grid>

</CheckBox>-->

<CheckBox x:Name="Coupe" Style="{StaticResource CustomCheckBoxStyle}">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="2\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<TextBlock Text="Купе" Grid.Column="0" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" HorizontalAlignment="Left"/>

<Image Source="/Resources/Body/coupe.png" Grid.Column="1" Height="25" HorizontalAlignment="Right"/>

</Grid>

</CheckBox>

<!--<CheckBox x:Name="Van" Style="{StaticResource CustomCheckBoxStyle}">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="2\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<TextBlock Text="Фургон" Grid.Column="0" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" HorizontalAlignment="Left"/>

<Image Source="/Resources/Body/van.png" Grid.Column="1" Height="25" HorizontalAlignment="Right"/>

</Grid>

</CheckBox>-->

<CheckBox x:Name="Pickup" Style="{StaticResource CustomCheckBoxStyle}">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="2\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<TextBlock Text="Пикап" Grid.Column="0" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" HorizontalAlignment="Left"/>

<Image Source="/Resources/Body/pickup.png" Grid.Column="1" Height="25" HorizontalAlignment="Right"/>

</Grid>

</CheckBox>

<CheckBox x:Name="Cabriolet" Style="{StaticResource CustomCheckBoxStyle}">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="2\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<TextBlock Text="Кабриолет" Grid.Column="0" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" HorizontalAlignment="Left"/>

<Image Source="/Resources/Body/cabriolet.png" Grid.Column="1" Height="25" HorizontalAlignment="Right"/>

</Grid>

</CheckBox>

</StackPanel>

</StackPanel>

</ScrollViewer>

</Grid>

<Button x:Name="AddCarButton" Margin="0,10,18,0" VerticalAlignment="Top" Height="34" HorizontalAlignment="Right" Width="34" Click="AddCarButton\_Click">

<Button.Template>

<ControlTemplate TargetType="Button">

<Grid>

<Border Background="{DynamicResource TextSecundaryColor}" CornerRadius="10">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

<iconPacks:PackIconUnicons Kind="Plus" Foreground="{DynamicResource PrimaryBackgroundColor}" HorizontalAlignment="Center" VerticalAlignment="Center" />

</Grid>

</ControlTemplate>

</Button.Template>

</Button>

</Grid>

</Border>

</Page>

using OneWay.Class;

using OneWay.Controls;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Input;

namespace OneWay.Pages

{

public partial class AllCarsPage : Page

{

public int IdUser;

DataBase db = new DataBase("OneWayDB.db");

public AllCarsPage(int IdUser)

{

InitializeComponent();

this.IdUser = IdUser;

}

private void AcceptFilter\_Click(object sender, RoutedEventArgs e)

{

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

if (CarBrand.Text != "")

{

filter.Add($"Brand = '{CarBrand.Text}'");

}

if (CarModel.Text != "")

{

filter.Add($"Model = '{CarModel.Text}'");

}

if (SeatFrom.Text != "" && SeatTo.Text != "")

{

filter.Add($"Seats BETWEEN {SeatFrom.Text} AND {SeatTo.Text}");

}

else

{

if(SeatFrom.Text != "")

{

filter.Add($"Seats >= {SeatFrom.Text}");

}

if(SeatTo.Text != "")

{

filter.Add($"Seats <= {SeatTo.Text}");

}

}

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

if (Gasoline.IsChecked == true)

{

fuel.Add("Бензин");

}

if (Diesel.IsChecked == true)

{

fuel.Add("Дизельное топливо");

}

if (Electro.IsChecked == true)

{

fuel.Add("Электричество");

}

if (Gas.IsChecked == true)

{

fuel.Add("Газ");

}

string fuelString = "Fuel = ";

if (fuel.Count != 0)

{

if (fuel.Count >= 2)

{

fuelString += string.Join(" OR Fuel = ", fuel.Select(f => $"'{f}'"));

}

else

{

fuelString += $"'{fuel[0]}'";

}

}

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

if (Auto.IsChecked == true)

{

gearBox.Add("АКПП");

}

if (Mech.IsChecked == true)

{

gearBox.Add("МКПП");

}

string gearBoxString = "GearBox = ";

if (gearBox.Count != 0)

{

if (gearBox.Count >= 2)

{

gearBoxString += string.Join(" OR GearBox = ", gearBox.Select(f => $"'{f}'"));

}

else

{

gearBoxString += $"'{gearBox[0]}'";

}

}

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

if (Sedan.IsChecked == true)

{

Body.Add("Седан");

}

if (Jeep.IsChecked == true)

{

Body.Add("Внедорожник");

}

if (Universal.IsChecked == true)

{

Body.Add("Униварсал");

}

if (Hatchback.IsChecked == true)

{

Body.Add("Хетчбек");

}

if (Liftback.IsChecked == true)

{

Body.Add("Лифтбек");

}

if (Minivan.IsChecked == true)

{

Body.Add("Минивэн");

}

if (Coupe.IsChecked == true)

{

Body.Add("Купе");

}

if (Pickup.IsChecked == true)

{

Body.Add("Купе");

}

string bodyString = "Body = ";

if (Body.Count != 0)

{

if (Body.Count >= 2)

{

bodyString += string.Join(" OR Body = ", Body.Select(f => $"'{f}'"));

}

else

{

bodyString += $"'{Body[0]}'";

}

}

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

if (filter.Count > 0)

{

allFilters.AddRange(filter);

}

if (!string.IsNullOrEmpty(fuelString) && fuelString.Length > 7)

{

allFilters.Add($"({fuelString})");

}

if (!string.IsNullOrEmpty(gearBoxString) && gearBoxString.Length > 10)

{

allFilters.Add($"({gearBoxString})");

}

if (!string.IsNullOrEmpty(bodyString) && bodyString.Length > 7)

{

allFilters.Add($"({bodyString})");

}

string resultString = string.Join(" AND ", allFilters);

List<Car> allCar = db.GetFilterCars(resultString, IdUser);

if(allCar.Count == 0)

{

CarPanel.Children.Clear();

CustomMessageBox.Show("Уведомление", "Автомобиль с такими характеристикми не найден.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

}

else

{

CarPanel.Children.Clear();

AddCarInPanel(allCar);

}

}

private void ClearButton\_Click(object sender, RoutedEventArgs e)

{

foreach (var child in FilterPanel.Children)

{

if (child is CheckBox checkBox)

{

checkBox.IsChecked = false;

}

if (child is TextBox textBox && !string.IsNullOrEmpty(textBox.Name))

{

textBox.Text = string.Empty;

}

if (child is StackPanel stackPanel)

{

foreach (var childTwo in stackPanel.Children)

{

if (childTwo is TextBox textBoxTwo && !string.IsNullOrEmpty(textBoxTwo.Name))

{

textBoxTwo.Text = string.Empty;

}

if (childTwo is CheckBox checkBoxTwo)

{

checkBoxTwo.IsChecked = false;

}

}

}

}

}

private void DateFrom\_PreviewTextInput(object sender, TextCompositionEventArgs e)

{

if (!char.IsDigit(e.Text, e.Text.Length - 1))

{

e.Handled = true;

}

}

private void DateBefore\_PreviewTextInput(object sender, TextCompositionEventArgs e)

{

if (!char.IsDigit(e.Text, e.Text.Length - 1))

{

e.Handled = true;

return;

}

}

private void SeatTo\_PreviewTextInput(object sender, TextCompositionEventArgs e)

{

if (!char.IsDigit(e.Text, e.Text.Length - 1))

{

e.Handled = true;

}

}

private void SeatFrom\_PreviewTextInput(object sender, TextCompositionEventArgs e)

{

if (!char.IsDigit(e.Text, e.Text.Length - 1))

{

e.Handled = true;

}

}

private void Page\_Loaded(object sender, RoutedEventArgs e)

{

List<Car> allCar = db.GetAllCars(IdUser);

AddCarInPanel(allCar);

}

private void AddCarInPanel(List<Car> cars)

{

CarPanel.Children.Clear();

foreach (var item in cars)

{

CarItem newCar = new CarItem();

newCar.Id = item.IdCar.ToString();

newCar.CarName = item.Brand;

newCar.Model = item.Model.ToString();

newCar.Transmission = item.GearBox;

newCar.Seat = item.Seats.ToString();

newCar.Body = item.Body;

newCar.FavoriteCar = db.CheckFavoriteCar(IdUser, item.IdCar).ToString();

if (item.Volume.ToString() == "0")

{

newCar.Volume = $"{item.EngineType}";

newCar.FuelConsumption = $"{item.FuelConsumption} кВт\*ч / км";

}

else

{

newCar.Volume = $"{Math.Round(item.Volume.Value / 1000, 2)} / {item.Fuel}";

newCar.FuelConsumption = $"{item.FuelConsumption} л / 100 км";

}

newCar.FavoriteButtonClicked += FavoriteButton\_Click;

newCar.DeleteButtonClicked += DeleteButton\_Click;

newCar.GridClicked += GridButton\_Click;

CarPanel.Children.Add(newCar);

}

}

private void AddCarButton\_Click(object sender, RoutedEventArgs e)

{

AddCarPage addCarPage = new AddCarPage(IdUser);

Window parentWindow = Window.GetWindow(this);

if (parentWindow != null && parentWindow is MainWindow mainWindow && mainWindow.fContainer != null)

{

mainWindow.fContainer.Content = addCarPage;

}

}

public void FavoriteButton\_Click(object sender, EventArgs e)

{

if (sender is CarItem carItem)

{

if (carItem.FavoriteCar == "0")

{

var result = CustomMessageBox.Show("Уведомление", $"Желаете добавить автомобиль в избранные автомобили?", MessageBoxButton.YesNo, CustomMessageBox.MessageBoxImage.Question);

if (result == MessageBoxResult.Yes)

{

if(db.AddFavoriteCar(IdUser, Convert.ToInt32(carItem.Id)))

{

carItem.FavoriteCar = "1";

CustomMessageBox.Show("Уведомление", $"Автомобиль успешно добавлен в избранные автомобили.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

}

else

{

CustomMessageBox.Show("Уведомление", $"Автомобиль не был добавлен в избранные автомобили.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

}

}

}

else

{

var result = CustomMessageBox.Show("Уведомление", $"Желаете удалить автомобиль из избранных автомобилей?", MessageBoxButton.YesNo, CustomMessageBox.MessageBoxImage.Question);

if (result == MessageBoxResult.Yes)

{

if (db.DeleteFavoriteCar(IdUser, Convert.ToInt32(carItem.Id)))

{

carItem.FavoriteCar = "0";

CustomMessageBox.Show("Уведомление", $"Автомобиль успешно удален из избранных автомобилей.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

}

else

{

CustomMessageBox.Show("Уведомление", $"Автомобиль не удален из избранных автомобилей.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

}

}

}

}

}

public void DeleteButton\_Click(object sender, EventArgs e)

{

if (sender is CarItem carItem)

{

var result = CustomMessageBox.Show("Уведомление", $"Желаете удалить данный автомобиль и поездки совершенные на нем?", MessageBoxButton.YesNo, CustomMessageBox.MessageBoxImage.Question);

if (result == MessageBoxResult.Yes)

{

if (db.DeleteCar(Convert.ToInt32(carItem.Id)))

{

CustomMessageBox.Show("Уведомление", $"Автомобиль успешно удален.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

List<Car> allCar = db.GetAllCars(IdUser);

AddCarInPanel(allCar);

}

else

{

CustomMessageBox.Show("Уведомление", $"Автомобиль не удален удален.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

}

}

}

}

public void GridButton\_Click(object sender, EventArgs e)

{

if (sender is CarItem carItem)

{

CarInfoPage carInfoPage = new CarInfoPage(IdUser,Convert.ToInt32(carItem.Id));

Window parentWindow = Window.GetWindow(this);

if (parentWindow != null && parentWindow is MainWindow mainWindow && mainWindow.fContainer != null)

{

mainWindow.fContainer.Content = carInfoPage;

}

}

}

}

}

<Page xmlns:Controls="clr-namespace:OneWay.Controls" x:Class="OneWay.Pages.AllFuel"

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:OneWay.Pages"

xmlns:iconPacks="http://metro.mahapps.com/winfx/xaml/iconpacks"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800"

Title="Fuel" Loaded="Page\_Loaded">

<Page.Resources>

<Style x:Key="CustomCheckBoxStyle" TargetType="CheckBox">

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="CheckBox">

<Grid Background="Transparent" Margin="5 0 0 5">

<Ellipse x:Name="CheckBoxEllipse" Fill="#B0B7C3" StrokeThickness="1" Stroke="#B0B7C3" Height="15" Width="15" HorizontalAlignment="Left" VerticalAlignment="Center"/>

<Ellipse x:Name="CheckboxCheckMark" Fill="#1d2335" Margin ="2.5 0 0 0" Height="10" Width="10" HorizontalAlignment="Left" VerticalAlignment="Center"/>

<ContentPresenter HorizontalAlignment="Center" Margin="30 0 0 0" VerticalAlignment="Center" Width="210"/>

</Grid>

<ControlTemplate.Triggers>

<Trigger Property="IsChecked" Value="True">

<Setter TargetName="CheckboxCheckMark" Property="Visibility" Value="Visible"/>

</Trigger>

<Trigger Property="IsChecked" Value="False">

<Setter TargetName="CheckboxCheckMark" Property="Visibility" Value="Collapsed"/>

</Trigger>

</ControlTemplate.Triggers>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

</Page.Resources>

<Border CornerRadius="5" Background="{DynamicResource SecundaryBackgroundColor}">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="auto"/>

</Grid.ColumnDefinitions>

<TextBlock Text="Цены на топливо" Grid.Column="0" FontSize ="25" Foreground="{DynamicResource TextSecundaryColor}" Margin="10,10,10,10"/>

<ScrollViewer Grid.Row="0" Margin="0 50 0 0">

<WrapPanel x:Name="FuelPanel"/>

</ScrollViewer>

<Grid Grid.Column="1">

<Grid.RowDefinitions>

<RowDefinition Height="auto"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="50"/>

</Grid.RowDefinitions>

<Grid Grid.Row="0">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<TextBlock Text="Фильтры" Grid.Column="0" FontSize ="25" Foreground="{DynamicResource TextSecundaryColor}" Margin="5 10 10 10"/>

<Button Grid.Column="1" Margin="10" BorderThickness="0" Click="AcceptFilter\_Click" Width="42" Height="34">

<Button.Template>

<ControlTemplate TargetType="Button">

<Grid>

<Border Background="#006400" CornerRadius="10">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

<iconPacks:PackIconUnicons Kind="Search" Foreground="{DynamicResource TextSecundaryColor}" HorizontalAlignment="Center" VerticalAlignment="Center" />

</Grid>

</ControlTemplate>

</Button.Template>

</Button>

<Button x:Name="ClearButton" Grid.Column="2" Margin="10" BorderThickness="0" Width="42" Height="34" Click="ClearButton\_Click">

<Button.Template>

<ControlTemplate TargetType="Button">

<Grid>

<Border Background="#FFCA1E1E" CornerRadius="10">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

<iconPacks:PackIconUnicons Kind="Times" Foreground="{DynamicResource TextSecundaryColor}" HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Grid>

</ControlTemplate>

</Button.Template>

</Button>

</Grid>

<ScrollViewer Grid.Row="1">

<StackPanel x:Name="FilterPanel">

<TextBlock Text="Вид топлива" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="5 0 0 0"/>

<StackPanel>

<CheckBox x:Name="Gasoline" Style="{StaticResource CustomCheckBoxStyle}" >

<TextBlock Text="Бензин" Grid.Column="0" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" HorizontalAlignment="Left"/>

</CheckBox>

<CheckBox x:Name="Diesel" Style="{StaticResource CustomCheckBoxStyle}">

<TextBlock Text="Дизельное топливо" Grid.Column="0" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" HorizontalAlignment="Left"/>

</CheckBox>

<CheckBox x:Name="Electro" Style="{StaticResource CustomCheckBoxStyle}">

<TextBlock Text="Электричество" Grid.Column="0" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" HorizontalAlignment="Left"/>

</CheckBox>

</StackPanel>

</StackPanel>

</ScrollViewer>

<TextBlock Grid.Row="2" x:Name="info" Width="262" Text="Данные взяты c сайта Белорусьнефть на 22.04.2024" FontSize ="15" TextAlignment="Left" Foreground="{DynamicResource TextSecundaryColor}" TextWrapping="WrapWithOverflow" MouseLeftButtonDown="TextBlock\_MouseLeftButtonDown"/>

</Grid>

</Grid>

</Border>

</Page>

using OneWay.Class;

using OneWay.Controls;

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.Windows;

using System.Windows.Controls;

namespace OneWay.Pages

{

public partial class AllFuel : Page

{

public int IdUser;

DataBase db = new DataBase("OneWayDB.db");

public AllFuel(int idUser)

{

InitializeComponent();

IdUser = idUser;

info.Text = $"Данные взяты c сайта Белорусьнефть на {db.GetLastDate()}";

}

private void AcceptFilter\_Click(object sender, RoutedEventArgs e)

{

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

if (Gasoline.IsChecked == true)

{

selectedFuelTypes.Add("Бензин");

}

if (Diesel.IsChecked == true)

{

selectedFuelTypes.Add("Дизельное топливо");

}

if (Electro.IsChecked == true)

{

selectedFuelTypes.Add("Электричество");

}

FuelPanel.Children.Clear();

DataBase db = new DataBase("OneWayDB.db");

List<Fuel> allFuel = db.GetFilterFuelTypes(selectedFuelTypes);

foreach (var item in allFuel)

{

FuelItem fuelItem = new FuelItem();

fuelItem.FuelType = item.FuelType;

fuelItem.OctanNumber = item.OctaneNumber;

fuelItem.PriceRub = item.Price.ToString() + " руб";

FuelPanel.Children.Add(fuelItem);

}

}

private void ClearButton\_Click(object sender, RoutedEventArgs e)

{

foreach (var child in FilterPanel.Children)

{

if (child is CheckBox checkBox)

{

checkBox.IsChecked = false;

}

if (child is StackPanel stackPanel)

{

foreach (var childTwo in stackPanel.Children)

{

if (childTwo is TextBox textBoxTwo && !string.IsNullOrEmpty(textBoxTwo.Name))

{

textBoxTwo.Text = string.Empty;

}

if (childTwo is CheckBox checkBoxTwo)

{

checkBoxTwo.IsChecked = false;

}

}

}

}

}

private void Page\_Loaded(object sender, RoutedEventArgs e)

{

DataBase db = new DataBase("OneWayDB.db");

List<Fuel> allFuel = db.GetFuelTypes(IdUser);

foreach(var item in allFuel)

{

FuelItem fuelItem = new FuelItem();

fuelItem.FuelType = item.FuelType;

fuelItem.OctanNumber = item.OctaneNumber;

fuelItem.PriceRub = item.Price.ToString() + " руб";

fuelItem.IdUser = IdUser.ToString();

fuelItem.GridClicked += Grid\_Click;

FuelPanel.Children.Add(fuelItem);

}

}

public void Grid\_Click(object sender, EventArgs e)

{

if (sender is FuelItem fuelItem)

{

FuelPriceEditPage fuelEditPage = new FuelPriceEditPage(IdUser, db.GetFuelId(fuelItem.OctanNumber));

Window parentWindow = Window.GetWindow(this);

if (parentWindow != null && parentWindow is MainWindow mainWindow && mainWindow.fContainer != null)

{

mainWindow.fContainer.Content = fuelEditPage;

}

}

}

private void TextBlock\_MouseLeftButtonDown(object sender, System.Windows.Input.MouseButtonEventArgs e)

{

Process.Start($"https://azs.belorusneft.by/sitebeloil/ru/center/azs/center/fuelandService/price/");

}

}

}

<Page x:Class="OneWay.Pages.AllRoutes"

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:OneWay.Pages"

xmlns:iconPacks="http://metro.mahapps.com/winfx/xaml/iconpacks"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800"

Title="AllRoutes">

<Page.Resources>

<Style x:Key="CustomCheckBoxStyle" TargetType="CheckBox">

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="CheckBox">

<Grid Background="Transparent" Margin="5 0 0 5">

<Ellipse x:Name="CheckBoxEllipse" Fill="#B0B7C3" StrokeThickness="1" Stroke="#B0B7C3" Height="15" Width="15" HorizontalAlignment="Left" VerticalAlignment="Center"/>

<Ellipse x:Name="CheckboxCheckMark" Fill="#1d2335" Margin ="2.5 0 0 0" Height="10" Width="10" HorizontalAlignment="Left" VerticalAlignment="Center"/>

<ContentPresenter HorizontalAlignment="Center" Margin="30 0 0 0" VerticalAlignment="Center" Width="210"/>

</Grid>

<ControlTemplate.Triggers>

<Trigger Property="IsChecked" Value="True">

<Setter TargetName="CheckboxCheckMark" Property="Visibility" Value="Visible"/>

</Trigger>

<Trigger Property="IsChecked" Value="False">

<Setter TargetName="CheckboxCheckMark" Property="Visibility" Value="Collapsed"/>

</Trigger>

</ControlTemplate.Triggers>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

</Page.Resources>

<Grid>

<Border CornerRadius="5" Background="{DynamicResource SecundaryBackgroundColor}">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="auto"/>

</Grid.ColumnDefinitions>

<TextBlock Text="Маршруты" Grid.Column="0" FontSize ="25" Foreground="{DynamicResource TextSecundaryColor}" Margin="10,10,10,10"/>

<Button x:Name="AddRouteButton" Margin="0,10,18,0" VerticalAlignment="Top" Height="34" HorizontalAlignment="Right" Width="34" Click="AddRouteButton\_Click" >

<Button.Template>

<ControlTemplate TargetType="Button">

<Grid>

<Border Background="{DynamicResource TextSecundaryColor}" CornerRadius="10">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

<iconPacks:PackIconUnicons Kind="Plus" Foreground="{DynamicResource PrimaryBackgroundColor}" HorizontalAlignment="Center" VerticalAlignment="Center" />

</Grid>

</ControlTemplate>

</Button.Template>

</Button>

<ScrollViewer Grid.Row="0" Margin="0 50 0 0">

<StackPanel x:Name="RoutePanel">

</StackPanel>

</ScrollViewer>

<Grid Grid.Column="1">

<Grid.RowDefinitions>

<RowDefinition Height="auto"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<Grid Grid.Row="0">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<TextBlock Text="Фильтры" Grid.Column="0" FontSize ="25" Foreground="{DynamicResource TextSecundaryColor}" Margin="5 10 10 10"/>

<Button Grid.Column="1" Margin="10" BorderThickness="0" Width="42" Height="34" Click="Button\_Click">

<Button.Template>

<ControlTemplate TargetType="Button">

<Grid>

<Border Background="#006400" CornerRadius="10">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

<iconPacks:PackIconUnicons Kind="Search" Foreground="{DynamicResource TextSecundaryColor}" HorizontalAlignment="Center" VerticalAlignment="Center" />

</Grid>

</ControlTemplate>

</Button.Template>

</Button>

<Button x:Name="ClearButton" Grid.Column="2" Margin="10" BorderThickness="0" Width="42" Height="34" Click="ClearButton\_Click">

<Button.Template>

<ControlTemplate TargetType="Button">

<Grid>

<Border Background="#FFCA1E1E" CornerRadius="10">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

<iconPacks:PackIconUnicons Kind="Times" Foreground="{DynamicResource TextSecundaryColor}" HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Grid>

</ControlTemplate>

</Button.Template>

</Button>

</Grid>

<ScrollViewer Grid.Row="1">

<StackPanel x:Name="FilterPanel">

<TextBlock Text="Точка отправки" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="5 0 0 0"/>

<TextBox x:Name="StartPoint" Tag="Город" FontSize="15" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="5 0 0 0"/>

<TextBlock Text="Точка прибытия" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="5 0 0 0"/>

<TextBox x:Name="FinishPoint" Tag="Город" FontSize="15" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="5 0 0 0"/>

</StackPanel>

</ScrollViewer>

</Grid>

</Grid>

</Border>

</Grid>

</Page>

using OneWay.Class;

using OneWay.Controls;

using System;

using System.Collections.Generic;

using System.Windows;

using System.Windows.Controls;

namespace OneWay.Pages

{

public partial class AllRoutes : Page

{

public int IdUser;

DataBase db = new DataBase("OneWayDB.db");

List<Class.Route> allRoute = new List<Class.Route>();

public AllRoutes(int Iduser)

{

InitializeComponent();

this.IdUser = Iduser;

allRoute = db.GetAllRoutes(this.IdUser);

CreateItem();

}

private void CreateItem()

{

RoutePanel.Children.Clear();

foreach (var item in allRoute)

{

var info = db.GetRouteInfoById(item.IdRoute);

string startPoint = info.Item1;

string finishPoint = info.Item2;

double distance = info.Item3;

string travelTime = info.Item4;

string points = info.Item5;

RouteItem route = new RouteItem();

route.Id = item.IdRoute.ToString();

route.TripName = $"{startPoint} - {finishPoint}";

route.Distance = distance + " км";

TimeSpan date = TimeSpan.FromMinutes(Convert.ToDouble(travelTime));

if (date.Days >= 1)

{

if (date.Hours == 0)

{

route.Time = $"{date.Days} д {date.Minutes} мин";

}

else

{

route.Time = $"{date.Days} д {date.Hours} ч {date.Minutes} мин";

}

}

else

{

if (date.Hours != 0)

{

route.Time = $"{date.Hours} ч {date.Minutes} мин";

}

else

{

route.Time = $"{date.Minutes} мин";

}

}

string formattedPoints = "";

if (!string.IsNullOrEmpty(points))

{

string[] point = points.Split('#');

formattedPoints += string.Join("\n", point);

}

route.Point = formattedPoints;

route.FavoriteRoute = db.CheckFavoriteRoute(IdUser, item.IdRoute).ToString();

route.InfoButtonClicked += Route\_InfoButtonClicked;

route.DeleteButtonClicked += Route\_DeleteButtonClicked;

route.FavoriteIconClicked += Route\_FavoriteButtonClicked;

RoutePanel.Children.Add(route);

}

}

private void Route\_FavoriteButtonClicked(object sender, EventArgs e)

{

if (sender is RouteItem routeItem)

{

if (routeItem.FavoriteRoute == "0")

{

var result = CustomMessageBox.Show("Уведомление", $"Желаете добавить маршрут в избранные маршруты?", MessageBoxButton.YesNo, CustomMessageBox.MessageBoxImage.Question);

if (result == MessageBoxResult.Yes)

{

if (db.AddFavoriteRoute(IdUser, Convert.ToInt32(routeItem.Id)))

{

routeItem.FavoriteRoute = "1";

CustomMessageBox.Show("Уведомление", $"Маршрут успешно добавлен в избранные маршруты.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

}

else

{

CustomMessageBox.Show("Уведомление", $"Маршрут не был добавлен в избранные автомобили.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

}

}

}

else

{

var result = CustomMessageBox.Show("Уведомление", $"Желаете удалить маршрут из избранных маршрутов?", MessageBoxButton.YesNo, CustomMessageBox.MessageBoxImage.Question);

if (result == MessageBoxResult.Yes)

{

if (db.DeleteFavoriteRoute(IdUser, Convert.ToInt32(routeItem.Id)))

{

routeItem.FavoriteRoute = "0";

CustomMessageBox.Show("Уведомление", $"Маршрут успешно удален из избранных маршрутов.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

}

else

{

CustomMessageBox.Show("Уведомление", $"Маршрут не удален из избранных маршрутов.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

}

}

}

}

}

private void Route\_InfoButtonClicked(object sender, EventArgs e)

{

if (sender is RouteItem routeItem)

{

var messageBoxResult = CustomMessageBox.Show("Уведомление", $"Желаете посмотреть информаци о маршруте?", MessageBoxButton.YesNo, CustomMessageBox.MessageBoxImage.Information);

if (messageBoxResult == MessageBoxResult.Yes)

{

RouteInfoPage routePage = new RouteInfoPage(IdUser, Convert.ToInt32(routeItem.Id));

Window parentWindow = Window.GetWindow(this);

if (parentWindow != null && parentWindow is MainWindow mainWindow && mainWindow.fContainer != null)

{

mainWindow.fContainer.Content = routePage;

}

}

}

}

private void Route\_DeleteButtonClicked(object sender, EventArgs e)

{

var quequestion = CustomMessageBox.Show("Уведомление", $"Желаете удалить данный маршрут и удалить все совершенные поездки по данному маршруту.", MessageBoxButton.YesNo, CustomMessageBox.MessageBoxImage.Information);

if (quequestion == MessageBoxResult.Yes)

{

if (sender is RouteItem routeItem)

{

int id = Convert.ToInt32(routeItem.Id);

if (db.DeleteRoute(id))

{

CustomMessageBox.Show("Уведомление", $"Данный маршрут был удален.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

allRoute = db.GetAllRoutes(this.IdUser);

CreateItem();

}

else

{

CustomMessageBox.Show("Уведомление", $"Данная поездка не была удалена из истории.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

}

}

}

}

private void ClearButton\_Click(object sender, RoutedEventArgs e)

{

StartPoint.Text = "";

FinishPoint.Text = "";

}

private void AddRouteButton\_Click(object sender, RoutedEventArgs e)

{

AddRoutePage addCarPage = new AddRoutePage(IdUser);

Window parentWindow = Window.GetWindow(this);

if (parentWindow != null && parentWindow is MainWindow mainWindow && mainWindow.fContainer != null)

{

mainWindow.fContainer.Content = addCarPage;

}

}

private void Button\_Click(object sender, RoutedEventArgs e)

{

if(!string.IsNullOrWhiteSpace(StartPoint.Text) | !string.IsNullOrEmpty(FinishPoint.Text))

{

int startId = db.GetCityIdByName(IdUser, StartPoint.Text);

int finishId = db.GetCityIdByName(IdUser, FinishPoint.Text);

if(startId == -1 & !string.IsNullOrWhiteSpace(StartPoint.Text))

{

CustomMessageBox.Show("Уведомление", "Точка отправления не найдена. Проверьте название и повторите попытку", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

return;

}

if (finishId == -1 & !string.IsNullOrWhiteSpace(FinishPoint.Text))

{

CustomMessageBox.Show("Уведомление", "Точка прибытия не найдена. Проверьте название и повторите попытку", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

return;

}

allRoute = db.SelectRoutesByPoints(IdUser, startId, finishId);

CreateItem();

}

else

{

CustomMessageBox.Show("Уведомление", "Отображены все маршруты.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

allRoute = db.GetAllRoutes(this.IdUser);

CreateItem();

}

}

}

}

<Page x:Class="OneWay.Pages.AuthenticationPage"

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:iconPacks="http://metro.mahapps.com/winfx/xaml/iconpacks"

xmlns:local="clr-namespace:OneWay.Pages"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800"

Title="AuthenticationPage">

<Border CornerRadius="5" Background="{DynamicResource SecundaryBackgroundColor}">

<Border Height="370" Width="350" Background="{DynamicResource SecundaryBackgroundColor}" BorderBrush="{DynamicResource TextSecundaryColor}" BorderThickness="2" CornerRadius="10">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="210"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="auto"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<TextBlock Grid.Row="0" Grid.Column="1" Text="Аутентификация" FontSize ="25" Foreground="{DynamicResource TextSecundaryColor}" Margin="10,10,10,10" TextAlignment="Center"/>

<TextBlock Grid.Row="1" Grid.Column="1" Text="логин" FontSize ="25" TextAlignment="Center" Foreground="{DynamicResource TextSecundaryColor}" Margin="10,10,10,10"/>

<TextBox Grid.Row="2" Grid.Column="1" x:Name="Login" Margin="5 10 5 10" Height="30" TextAlignment="Center" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0 0 0 1.5" BorderBrush="{DynamicResource TextSecundaryColor}"/>

<TextBlock Grid.Row="3" Grid.Column="1" Text="пароль" FontSize ="25" TextAlignment="Center" Foreground="{DynamicResource TextSecundaryColor}" Margin="10,10,10,10"/>

<PasswordBox Grid.Row="4" Grid.Column="1" x:Name="Password" Margin="5 10 5 10" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0 0 0 1.5" BorderBrush="{DynamicResource TextSecundaryColor}" Visibility="Visible"/>

<TextBox Grid.Row="4" Grid.Column="1" x:Name="PasswordTextBox" Margin="5 10 5 10" Height="30" TextAlignment="Left" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0 0 0 1.5" BorderBrush="{DynamicResource TextSecundaryColor}" Visibility="Collapsed" TextChanged="PasswordTextBox\_TextChanged"/>

<Button Grid.Row="4" Grid.Column="2" BorderThickness="0" Width="34" Height="34" Click="Button\_Click" HorizontalAlignment="Left" Margin="10,0,0,0">

<Button.Template>

<ControlTemplate TargetType="Button">

<Grid>

<Border Background="{DynamicResource TextSecundaryColor}" CornerRadius="10">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

<iconPacks:PackIconFontAwesome x:Name="icon" Kind="EyeSlashSolid" Foreground="{DynamicResource SecundaryBackgroundColor}" HorizontalAlignment="Center" VerticalAlignment="Center" />

</Grid>

</ControlTemplate>

</Button.Template>

</Button>

<Button Grid.Row="5" Grid.Column="1" x:Name="AuthButton" Content="Войти" FontSize="20" Height="30" Click="AuthButon">

<Button.Template>

<ControlTemplate TargetType="Button">

<Grid>

<Border Background="{DynamicResource TextSecundaryColor}" CornerRadius="10">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

</Grid>

</ControlTemplate>

</Button.Template>

</Button>

<Button Grid.Row="6" Grid.Column="1" x:Name="RegButton" Content="Зарегистрироваться" FontSize="20" Height="30" Click="RegButtonClick">

<Button.Template>

<ControlTemplate TargetType="Button">

<Grid>

<Border Background="{DynamicResource TextSecundaryColor}" CornerRadius="10">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

</Grid>

</ControlTemplate>

</Button.Template>

</Button>

</Grid>

</Border>

</Border>

</Page>

using MahApps.Metro.IconPacks;

using OneWay.Class;

using OneWay.Controls;

using System;

using System.Windows;

using System.Windows.Controls;

namespace OneWay.Pages

{

public partial class AuthenticationPage : Page

{

DataBase db = new DataBase("OneWayDB.db");

public event EventHandler<int> AuthenticationCompleted;

public AuthenticationPage()

{

InitializeComponent();

}

private void AuthButon(object sender, RoutedEventArgs e)

{

if (string.IsNullOrWhiteSpace(Login.Text) ||

string.IsNullOrWhiteSpace(Password.Password))//проверка на заполнение полей

{

CustomMessageBox.Show("Уведомление", $"Заполните все поля и повторите попытку.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

return;

}

int id = db.CheckUser(Login.Text, AesEncryption.EncryptString(Password.Password));//шифрование пароля

if (id != -1)

{

AuthenticationCompleted?.Invoke(this, id);

}

else

{

CustomMessageBox.Show("Уведомление", $"Проверьте логин или пароль.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

}

}

private void RegButtonClick(object sender, RoutedEventArgs e)

{

RegistrationPage regPage = new RegistrationPage();

Window parentWindow = Window.GetWindow(this);

if (parentWindow != null && parentWindow is MainWindow mainWindow && mainWindow.fContainer != null)

{

mainWindow.fContainer.Content = regPage;

}

}

private void Button\_Click(object sender, RoutedEventArgs e)

{

Button button = sender as Button;

if (button != null)

{

ControlTemplate template = button.Template;

PackIconFontAwesome icon = template.FindName("icon", button) as PackIconFontAwesome;

if (icon != null)

{

if (Password.Visibility == Visibility.Visible)

{

PasswordTextBox.Visibility = Visibility.Visible;

Password.Visibility = Visibility.Collapsed;

PasswordTextBox.Text = Password.Password;

icon.Kind = PackIconFontAwesomeKind.EyeSolid;

}

else

{

PasswordTextBox.Visibility = Visibility.Collapsed;

Password.Visibility = Visibility.Visible;

Password.Password = PasswordTextBox.Text;

icon.Kind = PackIconFontAwesomeKind.EyeSlashSolid;

}

}

}

}

private void PasswordTextBox\_TextChanged(object sender, TextChangedEventArgs e)

{

Password.Password = PasswordTextBox.Text;

}

}

}

<Page x:Class="OneWay.Pages.CarInfoPage"

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:OneWay.Pages"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800"

Title="AddCarPage">

<Border CornerRadius="5" Background="{DynamicResource SecundaryBackgroundColor}">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="3\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<TextBlock Text="Информация об автомобиле" Grid.Column="0" FontSize ="25" Foreground="{DynamicResource TextSecundaryColor}" Margin="10,10,10,403"/>

<ScrollViewer Grid.Column="0" Margin="0 50 0 0">

<StackPanel x:Name="HistoryPanel" Orientation="Vertical">

<Grid x:Name="FilterGrid">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="1\*"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<!-- Первая строка -->

<TextBlock Grid.Row="0" Grid.Column="0" Text="Марка" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0"/>

<TextBlock Grid.Row="0" Grid.Column="1" x:Name="CarBrand" Tag="Марка" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" Margin="10 0 0 10" Visibility="Visible"/>

<!-- Вторая строка -->

<TextBlock Grid.Row="1" Grid.Column="0" Text="Модель" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0"/>

<TextBlock Grid.Row="1" Grid.Column="1" Name="CarModel" Tag="Модель" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" Margin="10 0 0 10" Visibility="Visible"/>

<!-- Третья строка -->

<TextBlock Grid.Row="2" Grid.Column="0" Text="Поколение" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0"/>

<TextBlock Grid.Row="2" Grid.Column="1" x:Name="CarGeneration" Tag="Поколение" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" Margin="10 0 0 10" Visibility="Visible"/>

<!-- Четвертая строка -->

<TextBlock Grid.Row="3" Grid.Column="0" Text="Комплектация" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0"/>

<TextBlock Grid.Row="3" Grid.Column="1" x:Name="CarEquipment" Tag="Комплектация" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" Margin="10 0 0 10" Visibility="Visible"/>

<!-- Пятая строка -->

<TextBlock Grid.Row="4" Grid.Column="0" Text="Год выпуска" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0"/>

<TextBlock Grid.Row="4" Grid.Column="1" x:Name="Year" Tag="Год выпуска" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" Margin="10 0 0 10" />

<!-- Шестая строка -->

<TextBlock Grid.Row="5" Grid.Column="0" Text="Тип кузова" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0"/>

<TextBlock Grid.Row="5" Grid.Column="1" x:Name="Body" FontSize="18" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 10"/>

<!-- Седьмая строка -->

<TextBlock Grid.Row="6" Grid.Column="0" Text="Тип привода" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0"/>

<TextBlock Grid.Row="6" Grid.Column="1" x:Name="Drive" FontSize="18" Margin="10 0 0 10" Foreground="{DynamicResource TextSecundaryColor}"/>

<!-- Восьмая строка -->

<TextBlock Grid.Row="7" Grid.Column="0" Text="Тип трансмиссии" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0"/>

<TextBlock Grid.Row="7" Grid.Column="1" x:Name="GearBox" FontSize="18" Height="35" Margin="10 0 0 10" Foreground="{DynamicResource TextSecundaryColor}"/>

<!-- Девятая строка -->

<TextBlock Grid.Row="8" Grid.Column="0" Text="Число мест" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0"/>

<TextBlock Grid.Row="8" Grid.Column="1" x:Name="Seats" Tag="Число мест" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" Margin="10 0 0 10" />

<!-- Десятая строка -->

<TextBlock Grid.Row="9" Grid.Column="0" Text="Число дверей" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0"/>

<TextBlock Grid.Row="9" Grid.Column="1" x:Name="Doors" Tag="Число дверей" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" Margin="10 0 0 10" />

<!-- Одиннадцатая строка -->

<TextBlock Grid.Row="10" Grid.Column="0" Text="Тип двигателя" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0"/>

<TextBlock Grid.Row="10" Grid.Column="1" x:Name="EngineType" FontSize="18" Height="35" Margin="10 0 5 6" Foreground="{DynamicResource TextSecundaryColor}" />

<!-- Двенадцатая строка -->

<TextBlock Grid.Row="11" Grid.Column="0" x:Name="TextVolume" Text="Объем двигателя, куб.см" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0" Visibility="Visible"/>

<TextBlock Grid.Row="11" Grid.Column="1" x:Name="Volume" Tag="Объем двигателя" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" Margin="10 0 0 10" Visibility="Visible"/>

<!-- Тринадцатая строка -->

<TextBlock Grid.Row="12" Grid.Column="0" Text="Потребляемое топливо" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0"/>

<TextBlock Grid.Row="12" Grid.Column="1" x:Name="Fuel" FontSize="18" Height="35" Margin="10 0 0 10" Foreground="{DynamicResource TextSecundaryColor}"/>

<!-- Четырнадцатая строка -->

<TextBlock Grid.Row="13" Grid.Column="0" x:Name="TextOctaneNumber" Text="Октановое число" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0" Visibility="Visible"/>

<TextBlock Grid.Row="13" Grid.Column="1" x:Name="OctaneNumber" FontSize="18" Height="35" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 10" Visibility="Visible"/>

<!-- Пятнадцатая строка -->

<TextBlock Grid.Row="14" Grid.Column="0" x:Name="TextFuelConsumption" Text="Расход топлива в смешанном цикле, л/100 км" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 6" TextWrapping="WrapWithOverflow"/>

<TextBlock Grid.Row="14" Grid.Column="1" x:Name="FuelConsumption" Tag="Потребление топлива" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" Margin="10 0 0 10"/>

<!-- Шеснадацатая строка -->

<TextBlock Grid.Row="15" Grid.Column="0" Text="Максимальная скорость, км/ч" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0"/>

<TextBlock Grid.Row="15" Grid.Column="1" x:Name="MaxSpeed" Tag="Максимальная скорость" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" Margin="10 0 0 10"/>

<!-- Семнадацатая строка -->

<TextBlock Grid.Row="16" Grid.Column="0" Text="Кондиционер" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 10"/>

<TextBlock Grid.Row="16" Grid.Column="1" x:Name="Conditioner" FontSize="18" Height="35" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 5 6"/>

<!-- Восемнадцатая строка-->

<TextBlock Grid.Row="17" Grid.Column="0" x:Name="TextBlockRangePerCharge" Text="Запас хода на электротяге, км" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0" Visibility="Visible"/>

<TextBlock Grid.Row="17" Grid.Column="1" x:Name="RangePerCharge" Tag="Запас хода на электротяге" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" Margin="10 0 0 10" Visibility="Visible"/>

<!-- Девятнадцатая строка -->

<TextBlock Grid.Row="18" Grid.Column="0" x:Name="TextBlockBatteryCapacity" Text="Ёмкость батареи, кВт\*ч" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0" Visibility="Visible"/>

<TextBlock Grid.Row="18" Grid.Column="1" x:Name="BatteryCapacity" Tag="Объем батареи" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" Margin="10 0 0 10" Visibility="Visible"/>

</Grid>

</StackPanel>

</ScrollViewer>

<StackPanel Grid.Column="1">

<TextBlock Text="Настройки" FontSize ="25" Foreground="{DynamicResource TextSecundaryColor}" Margin="10,10,10,0"/>

<Button x:Name="DeleteButton" Content="Удалить" FontSize="15" Foreground="{DynamicResource SecundaryBackgroundColor}" Margin="10 5 10 5" BorderThickness="0" Click="DeleteButton\_Click">

<Button.Template>

<ControlTemplate TargetType="Button">

<Grid>

<Border x:Name="border" Background="{DynamicResource TextSecundaryColor}" CornerRadius="5">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

</Grid>

<ControlTemplate.Triggers>

<Trigger Property="IsMouseOver" Value="True">

<Setter TargetName="border" Property="Background" Value="#ADD8E6"/>

</Trigger>

</ControlTemplate.Triggers>

</ControlTemplate>

</Button.Template>

</Button>

<Button x:Name="FavoriteButton" FontSize="15" Foreground="{DynamicResource SecundaryBackgroundColor}" Margin="10 5 10 5" BorderThickness="0" Click="FavoriteButton\_Click">

<Button.Template>

<ControlTemplate TargetType="Button">

<Grid>

<Border x:Name="border" Background="{DynamicResource TextSecundaryColor}" CornerRadius="5">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

</Grid>

<ControlTemplate.Triggers>

<Trigger Property="IsMouseOver" Value="True">

<Setter TargetName="border" Property="Background" Value="#ADD8E6"/>

</Trigger>

</ControlTemplate.Triggers>

</ControlTemplate>

</Button.Template>

</Button>

</StackPanel>

</Grid>

</Border>

</Page>

using OneWay.Class;

using OneWay.Controls;

using System;

using System.Windows;

using System.Windows.Controls;

namespace OneWay.Pages

{

public partial class CarInfoPage : Page

{

DataBase db = new DataBase("OneWayDB.db");

Car selectCar = new Car();

public int IdCar;

public int IdUser;

public CarInfoPage(int IdUser,int IdCar)

{

InitializeComponent();

this.IdCar = IdCar;

this.IdUser = IdUser;

selectCar = db.GetCar(IdCar);

СompletionFill(selectCar);

if(db.CheckFavoriteCar(IdUser, IdCar) == 1)

{

this.FavoriteButton.Content = "Удалить из избранных авто";

}

else

{

this.FavoriteButton.Content = "Добавить в избранные авто";

}

}

void СompletionFill(Car selectCar)

{

this.CarBrand.Text = selectCar.Brand;

this.CarModel.Text = selectCar.Model;

this.CarGeneration.Text = selectCar.Generation;

this.CarEquipment.Text = selectCar.Equipment;

this.Year.Text = selectCar.Year;

this.Body.Text = selectCar.Body;

this.Drive.Text = selectCar.Drive;

this.GearBox.Text = selectCar.GearBox;

this.Seats.Text = selectCar.Seats.ToString();

this.Doors.Text = selectCar.Doors.ToString();

this.EngineType.Text = selectCar.EngineType;

this.MaxSpeed.Text = selectCar.MaxSpeed.ToString();

this.Conditioner.Text = selectCar.Conditioner;

if(selectCar.EngineType == "Электрический")

{

this.Volume.Visibility = Visibility.Collapsed;

this.TextVolume.Visibility = Visibility.Collapsed;

this.Fuel.Text = selectCar.Fuel;

this.TextFuelConsumption.Visibility = Visibility.Collapsed;

this.FuelConsumption.Visibility = Visibility.Collapsed;

this.TextOctaneNumber.Visibility = Visibility.Collapsed;

this.OctaneNumber.Visibility = Visibility.Collapsed;

this.BatteryCapacity.Text = selectCar.BatteryCapacity.ToString();

this.TextBlockBatteryCapacity.Visibility = Visibility.Visible;

this.BatteryCapacity.Visibility = Visibility.Visible;

this.RangePerCharge.Text = selectCar.RangePerCharge.ToString();

this.TextBlockRangePerCharge.Visibility = Visibility.Visible;

this.BatteryCapacity.Visibility = Visibility.Visible;

}

else if(selectCar.EngineType == "Гибрид")

{

this.Volume.Text = selectCar.Volume.ToString();

this.Volume.Visibility = Visibility.Visible;

this.TextVolume.Visibility = Visibility.Visible;

this.Fuel.Text = selectCar.Fuel;

this.TextFuelConsumption.Visibility = Visibility.Collapsed;

this.FuelConsumption.Visibility = Visibility.Collapsed;

if(selectCar.FuelOctane != 0)

{

this.OctaneNumber.Text = db.GetFuel(Convert.ToInt32(selectCar.FuelOctane));

this.OctaneNumber.Visibility = Visibility.Visible;

this.TextOctaneNumber.Visibility = Visibility.Visible;

}

else

{

this.OctaneNumber.Visibility = Visibility.Collapsed;

this.TextOctaneNumber.Visibility = Visibility.Collapsed;

}

if(selectCar.BatteryCapacity != null)

{

this.TextBlockBatteryCapacity.Visibility = Visibility.Visible;

this.BatteryCapacity.Visibility = Visibility.Visible;

}

else

{

this.TextBlockBatteryCapacity.Visibility = Visibility.Collapsed;

this.BatteryCapacity.Visibility = Visibility.Collapsed;

}

if(selectCar.RangePerCharge != null)

{

this.TextBlockRangePerCharge.Visibility = Visibility.Visible;

this.RangePerCharge.Visibility = Visibility.Visible;

}

else

{

this.TextBlockRangePerCharge.Visibility = Visibility.Collapsed;

this.RangePerCharge.Visibility = Visibility.Collapsed;

}

}

else

{

this.Volume.Text = selectCar.Volume.ToString();

this.Volume.Visibility = Visibility.Visible;

this.TextVolume.Visibility = Visibility.Visible;

this.FuelConsumption.Text = selectCar.FuelConsumption.ToString();

this.TextFuelConsumption.Visibility = Visibility.Visible;

this.FuelConsumption.Visibility = Visibility.Visible;

this.Fuel.Text = selectCar.Fuel;

if (selectCar.FuelOctane != 0)

{

this.OctaneNumber.Text = db.GetFuel(Convert.ToInt32(selectCar.FuelOctane));

this.OctaneNumber.Visibility = Visibility.Visible;

this.TextOctaneNumber.Visibility = Visibility.Visible;

}

else

{

this.OctaneNumber.Visibility = Visibility.Collapsed;

this.TextOctaneNumber.Visibility = Visibility.Collapsed;

}

this.TextBlockBatteryCapacity.Visibility = Visibility.Collapsed;

this.BatteryCapacity.Visibility = Visibility.Collapsed;

this.TextBlockRangePerCharge.Visibility = Visibility.Collapsed;

this.BatteryCapacity.Visibility = Visibility.Collapsed;

}

}

private void DeleteButton\_Click(object sender, RoutedEventArgs e)

{

var result = CustomMessageBox.Show("Уведомление", $"Желаете удалить данный автомобиль и поездки совершенные на нем?", MessageBoxButton.YesNo, CustomMessageBox.MessageBoxImage.Question);

if (result == MessageBoxResult.Yes)

{

if (db.DeleteCar(IdCar))

{

CustomMessageBox.Show("Уведомление", $"Автомобиль успешно удален.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

AllCarsPage carPage = new AllCarsPage(IdUser);

Window parentWindow = Window.GetWindow(this);

if (parentWindow != null && parentWindow is MainWindow mainWindow && mainWindow.fContainer != null)

{

mainWindow.fContainer.Content = carPage;

}

}

else

{

CustomMessageBox.Show("Уведомление", $"Автомобиль не удален удален.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

}

}

}

private void FavoriteButton\_Click(object sender, RoutedEventArgs e)

{

if (db.CheckFavoriteCar(IdUser,IdCar) == 0) // проверка добавлени ли автмобиль в избранное. если нет то

{

var result = CustomMessageBox.Show("Уведомление", $"Желаете добавить автомобиль в избранные автомобили?", MessageBoxButton.YesNo, CustomMessageBox.MessageBoxImage.Question);

if (result == MessageBoxResult.Yes)

{

if (db.AddFavoriteCar(IdUser, IdCar))

{

this.FavoriteButton.Content = "Удалить из избранных авто";

CustomMessageBox.Show("Уведомление", $"Автомобиль успешно добавлен в избранные автомобили.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

}

else

{

CustomMessageBox.Show("Уведомление", $"Автомобиль не был добавлен в избранные автомобили.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

}

}

}

else //если есть то удалить

{

var result = CustomMessageBox.Show("Уведомление", $"Желаете удалить автомобиль из избранных автомобилей?", MessageBoxButton.YesNo, CustomMessageBox.MessageBoxImage.Question);

if (result == MessageBoxResult.Yes)

{

if (db.DeleteFavoriteCar(IdUser, IdCar))

{

this.FavoriteButton.Content = "Добавить в избранные авто";

CustomMessageBox.Show("Уведомление", $"Автомобиль успешно удален из избранных автомобилей.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

}

else

{

CustomMessageBox.Show("Уведомление", $"Автомобиль не удален из избранных автомобилей.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

}

}

}

}

}

}

<Page x:Class="OneWay.Pages.Dashboard"

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:lvc="clr-namespace:LiveCharts.Wpf;assembly=LiveCharts.Wpf"

xmlns:local="clr-namespace:OneWay.Pages"

xmlns:iconPacks="http://metro.mahapps.com/winfx/xaml/iconpacks"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800"

Title="Dashboard">

<Border CornerRadius="5" Background="{DynamicResource SecundaryBackgroundColor}">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="4\*"/>

<ColumnDefinition Width="auto"/>

</Grid.ColumnDefinitions>

<lvc:CartesianChart Name="liveChart" Margin="10,10,10,10" Grid.Row="0" Zoom="X">

<lvc:CartesianChart.AxisX>

<lvc:Axis Title="Поездки" FontSize="15" Foreground="#bcc0c6" ShowLabels="True">

<lvc:Axis.Labels>

<Binding Path="TripDates"/>

</lvc:Axis.Labels>

<lvc:Axis.Separator>

<lvc:Separator StrokeThickness="0"></lvc:Separator>

</lvc:Axis.Separator>

</lvc:Axis>

</lvc:CartesianChart.AxisX>

<lvc:CartesianChart.AxisY>

<lvc:Axis Title="Сумма" FontSize="15" Foreground="#bcc0c6" ShowLabels="True" MinValue="0">

<lvc:Axis.Separator>

<lvc:Separator StrokeThickness="0"></lvc:Separator>

</lvc:Axis.Separator>

</lvc:Axis>

</lvc:CartesianChart.AxisY>

</lvc:CartesianChart>

<StackPanel Grid.Column="1" x:Name="FilterPanel" Margin="10 10 10 10">

<Grid Grid.Row="0">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<TextBlock Text="Фильтр" Grid.Column="0" FontSize ="25" Foreground="{DynamicResource TextSecundaryColor}" Margin="5 10 10 10"/>

<Button Grid.Column="1" Margin="10" BorderThickness="0" Width="42" Height="34" Click="Button\_Click">

<Button.Template>

<ControlTemplate TargetType="Button">

<Grid>

<Border Background="#006400" CornerRadius="10">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

<iconPacks:PackIconUnicons Kind="Search" Foreground="{DynamicResource TextSecundaryColor}" HorizontalAlignment="Center" VerticalAlignment="Center" />

</Grid>

</ControlTemplate>

</Button.Template>

</Button>

<Button x:Name="ClearButton" Grid.Column="2" Margin="10" BorderThickness="0" Width="42" Height="34" Click="ClearButton\_Click" >

<Button.Template>

<ControlTemplate TargetType="Button">

<Grid>

<Border Background="#FFCA1E1E" CornerRadius="10">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

<iconPacks:PackIconUnicons Kind="Times" Foreground="{DynamicResource TextSecundaryColor}" HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Grid>

</ControlTemplate>

</Button.Template>

</Button>

</Grid>

<TextBlock Text="Дата поездки" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="5 0 0 0"/>

<StackPanel Orientation="Horizontal">

<TextBox x:Name="DateFrom" Tag="c" FontSize="15" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="5 0 0 0" Width="100" />

<TextBlock Text="-" FontSize ="15" Foreground="Gray" HorizontalAlignment="Center"/>

<TextBox x:Name="DateBefore" Tag="по" FontSize="15" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="5 0 0 0" Width="100"/>

</StackPanel>

</StackPanel>

</Grid>

</Border>

</Page>

using LiveCharts;

using LiveCharts.Defaults;

using LiveCharts.Wpf;

using OneWay.Class;

using OneWay.Controls;

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.Globalization;

using System.Linq;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

namespace OneWay.Pages

{

public partial class Dashboard : Page

{

public ObservableCollection<string> TripDates { get; set; }

DataBase db = new DataBase("OneWayDB.db");

List<Class.Trip> userTrip = new List<Class.Trip>();

public static List<TripData> allTrip = new List<TripData> { };

int IdUser;

public Dashboard(int idUser)

{

InitializeComponent();

IdUser = idUser;

liveChart.Series.Clear();

userTrip.Clear();

allTrip.Clear();

userTrip = db.GetAllTripByUserId(IdUser);

if (userTrip.Count > 0)

{

liveChart.Visibility = Visibility.Visible;

userTrip = userTrip.OrderBy(trip => DateTime.Parse(trip.StartTime)).ToList();

for (int i = 0; i < userTrip.Count; i++)

{

allTrip.Add(new TripData(i, userTrip[i].UsedMoney, Convert.ToDateTime(userTrip[i].StartTime)));

}

string name = db.GetFirstNameById(1);

var seriesNew = new LineSeries

{

Title = name,

Values = observablePoints(allTrip),

Visibility = System.Windows.Visibility.Visible

};

TripDates = new ObservableCollection<string>();

DataContext = this;

foreach (var x in userTrip)

{

TripDates.Add(Convert.ToDateTime(x.StartTime).ToString("dd.MM.yyyy"));

}

TripDates.Add(allTrip[allTrip.Count - 1].Date.ToString("dd.MM.yyyy"));

liveChart.Series = new SeriesCollection { seriesNew };

}

else

{

liveChart.Visibility = Visibility.Hidden;

}

}

private ChartValues<ObservablePoint> observablePoints(List<TripData> list)

{

ChartValues<ObservablePoint> newPoint = new ChartValues<ObservablePoint>();

foreach (var x in list)

{

ObservablePoint point = new ObservablePoint(x.X, x.Y);

newPoint.Add(point);

}

return newPoint;

}

private void ClearButton\_Click(object sender, RoutedEventArgs e)

{

DateFrom.Text = string.Empty;

DateBefore.Text = string.Empty;

}

private void Button\_Click(object sender, RoutedEventArgs e)

{

string dateOne = DateFrom.Text;

string dateTwo = DateBefore.Text;

string filterDate = "";

DateTime date1, date2;

if (!string.IsNullOrWhiteSpace(dateOne) || !string.IsNullOrWhiteSpace(dateTwo))

{

if (!DateTime.TryParseExact(dateOne, "dd.MM.yyyy", CultureInfo.InvariantCulture, DateTimeStyles.None, out date1) && !string.IsNullOrWhiteSpace(dateOne))

{

CustomMessageBox.Show("Уведомление", $"Неправильный формат даты 1. Введите дату в формате dd.mm.yyyy.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

return;

}

if (!DateTime.TryParseExact(dateTwo, "dd.MM.yyyy", CultureInfo.InvariantCulture, DateTimeStyles.None, out date2) && !string.IsNullOrWhiteSpace(dateTwo))

{

CustomMessageBox.Show("Уведомление", $"Неправильный формат даты 2. Введите дату в формате dd.mm.yyyy.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

return;

}

if (date1 != DateTime.MinValue && date2 != DateTime.MinValue)

{

if (date1 < date2)

{

date2 = date2.Date.AddHours(23).AddMinutes(59).AddSeconds(59);

filterDate = $"StartTime BETWEEN '{date1.ToString("yyyy-MM-dd HH:mm:ss")}' AND '{date2.ToString("yyyy-MM-dd HH:mm:ss")}'";

}

else

{

CustomMessageBox.Show("Уведомление", $"Дата 1 больше, чем дата 2. Введите корректные данные и повторите попытку.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

return;

}

}

else if (date1 != DateTime.MinValue)

{

filterDate = $"StartTime >= '{date1.ToString("yyyy-MM-dd HH:mm:ss")}'";

}

else if (date2 != DateTime.MinValue)

{

date2 = date2.Date.AddHours(23).AddMinutes(59).AddSeconds(59);

filterDate = $"StartTime <= '{date2.ToString("yyyy-MM-dd HH:mm:ss")}'";

}

userTrip = db.GetAllTripByUserIdAndDateRange(IdUser, "AND " + filterDate);

userTrip = userTrip.OrderBy(trip => DateTime.Parse(trip.StartTime)).ToList();

if (userTrip.Count > 0)

{

liveChart.Series.Clear();

List<TripData> filteredTripData = new List<TripData>();

for (int i = 0; i < userTrip.Count; i++)

{

filteredTripData.Add(new TripData(i, userTrip[i].UsedMoney, Convert.ToDateTime(userTrip[i].StartTime)));

}

var seriesNew = new LineSeries

{

Title = db.GetFirstNameById(1),

Values = observablePoints(filteredTripData),

Visibility = Visibility.Visible

};

liveChart.Series.Add(seriesNew);

TripDates = new ObservableCollection<string>();

DataContext = this;

foreach (var x in userTrip)

{

TripDates.Add(Convert.ToDateTime(x.StartTime).ToString("dd.MM.yyyy"));

}

liveChart.Series = new SeriesCollection { seriesNew };

liveChart.AxisX[0].Labels = TripDates;

}

else

{

CustomMessageBox.Show("Уведомление", $"Поездки за данный период не найдены.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

}

}

else

{

userTrip = db.GetAllTripByUserId(IdUser);

userTrip = userTrip.OrderBy(trip => DateTime.Parse(trip.StartTime)).ToList();

liveChart.Series.Clear();

List<TripData> filteredTripData = new List<TripData>();

for (int i = 0; i < userTrip.Count; i++)

{

filteredTripData.Add(new TripData(i, userTrip[i].UsedMoney, Convert.ToDateTime(userTrip[i].StartTime)));

}

var seriesNew = new LineSeries

{

Title = db.GetFirstNameById(1),

Values = observablePoints(filteredTripData),

Visibility = Visibility.Visible

};

liveChart.Series.Add(seriesNew);

TripDates = new ObservableCollection<string>();

DataContext = this;

foreach (var x in userTrip)

{

TripDates.Add(Convert.ToDateTime(x.StartTime).ToString("dd.MM.yyyy"));

}

liveChart.Series = new SeriesCollection { seriesNew };

liveChart.AxisX[0].Labels = TripDates;

}

}

}

public class TripData

{

public double X { get; set; }

public double Y { get; set; }

public DateTime Date { get; set; }

public TripData(double x, double y, DateTime date)

{

X = x;

Y = y;

Date = date;

}

}

}

<Page x:Class="OneWay.Pages.FuelPriceEditPage"

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:OneWay.Pages"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800"

Title="FuelPriceEditPage">

<Border CornerRadius="5" Background="{DynamicResource SecundaryBackgroundColor}">

<Grid Width="700" HorizontalAlignment="Left">

<TextBlock x:Name="FuelInfo" Text="" Grid.Column="0" FontSize ="25" Foreground="{DynamicResource TextSecundaryColor}" Margin="10,10,10,403" Grid.ColumnSpan="2"/>

<ScrollViewer Grid.Column="0" Margin="0 50 0 0" Grid.ColumnSpan="2">

<StackPanel Orientation="Vertical">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="2\*"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<!-- Первая строка -->

<TextBlock Grid.Row="0" Grid.Column="0" x:Name="FuelPriceText" Text="Цена за литр" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0"/>

<TextBox Grid.Row="0" Grid.Column="1" x:Name="FuelPrice" Height="30" TextAlignment="Left" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0 0 0 1.5" BorderBrush="{DynamicResource TextSecundaryColor}" Margin="1"/>

<!-- вторая -->

<Button Grid.Row="1" Grid.Column="0" x:Name="AddButton" Content="Изменить цену" FontSize="20" Foreground="{DynamicResource SecundaryBackgroundColor}" Margin="10 5 10 5" BorderThickness="0" Click="Button\_Click">

<Button.Template>

<ControlTemplate TargetType="Button">

<Grid>

<Border x:Name="border" Background="{DynamicResource TextSecundaryColor}" CornerRadius="5">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

</Grid>

<ControlTemplate.Triggers>

<Trigger Property="IsMouseOver" Value="True">

<Setter TargetName="border" Property="Background" Value="#ADD8E6"/>

</Trigger>

</ControlTemplate.Triggers>

</ControlTemplate>

</Button.Template>

</Button>

</Grid>

</StackPanel>

</ScrollViewer>

</Grid>

</Border>

</Page>

using OneWay.Class;

using OneWay.Controls;

using System;

using System.Globalization;

using System.Windows;

using System.Windows.Controls;

namespace OneWay.Pages

{

public partial class FuelPriceEditPage : Page

{

DataBase db = new DataBase("OneWayDB.db");

int IdUser;

int IdFuel;

public FuelPriceEditPage(int idUser, int idFuel)

{

InitializeComponent();

IdUser = idUser;

IdFuel = idFuel;

FuelInfo.Text = $"Изменение цены на {db.GetFuelName(IdFuel)}";

if(IdFuel != 7 && IdFuel != 8)

{

FuelPriceText.Text = $"Цена за литр, руб";

}

else

{

FuelPriceText.Text = $"Цена за кВт\*ч, руб";

}

}

private void Button\_Click(object sender, RoutedEventArgs e)

{

if (string.IsNullOrEmpty(FuelPrice.Text))

{

CustomMessageBox.Show("Уведомление", $"Пожалуйста, заполните все обязательные поля.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

return;

}

double fuelPrice = 0;

if (!double.TryParse(FuelPrice.Text, out fuelPrice) || fuelPrice <= 0)

{

CustomMessageBox.Show("Предупреждение", $"Пожалуйста, введите корректное значение для цены топлива в формате (x,xx).", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

return;

}

string formattedFuelConsumption = fuelPrice.ToString("0.00", CultureInfo.InvariantCulture);

string date = DateTime.Now.ToString("dd.MM.yyyy");

if (db.UpdatePrice(IdFuel, IdUser, formattedFuelConsumption, date)) // добавление в цены в базу данных

{

CustomMessageBox.Show("Уведомление", $"Цена на топливо была обновлена.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

}

else

{

CustomMessageBox.Show("Уведомление", $"Цена на топливо не была обновлена.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

}

}

}

}

<Page x:Class="OneWay.Pages.History"

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:iconPacks="http://metro.mahapps.com/winfx/xaml/iconpacks"

xmlns:local="clr-namespace:OneWay.Pages"

xmlns:ctls="clr-namespace:OneWay.Controls"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800"

Title="Dashboard">

<Page.Resources>

<Style x:Key="ScrollBarTrackThumb"

TargetType="{x:Type Thumb}">

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="{x:Type Thumb}">

<Grid x:Name="Grid">

<Rectangle HorizontalAlignment="Stretch"

VerticalAlignment="Stretch"

Width="Auto"

Height="Auto"

Fill="Transparent" />

<Border x:Name="CornerScrollBarRectangle"

CornerRadius="5"

HorizontalAlignment="Stretch"

VerticalAlignment="Stretch"

Width="Auto"

Height="Auto"

Margin="0,1,0,1"

Background="{TemplateBinding Background}" />

</Grid>

<ControlTemplate.Triggers>

<Trigger Property="Tag"

Value="Horizontal">

<Setter TargetName="CornerScrollBarRectangle"

Property="Width"

Value="Auto" />

<Setter TargetName="CornerScrollBarRectangle"

Property="Height"

Value="6" />

</Trigger>

</ControlTemplate.Triggers>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

<Style x:Key="CustomCheckBoxStyle" TargetType="CheckBox">

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="CheckBox">

<Grid Background="Transparent" Margin="5 0 0 5">

<Ellipse x:Name="CheckBoxEllipse" Fill="#B0B7C3" StrokeThickness="1" Stroke="#B0B7C3" Height="15" Width="15" HorizontalAlignment="Left" VerticalAlignment="Center"/>

<Ellipse x:Name="CheckboxCheckMark" Fill="#1d2335" Margin ="2.5 0 0 0" Height="10" Width="10" HorizontalAlignment="Left" VerticalAlignment="Center"/>

<ContentPresenter HorizontalAlignment="Center" Margin="30 0 0 0" VerticalAlignment="Center" Width="210"/>

</Grid>

<ControlTemplate.Triggers>

<Trigger Property="IsChecked" Value="True">

<Setter TargetName="CheckboxCheckMark" Property="Visibility" Value="Visible"/>

</Trigger>

<Trigger Property="IsChecked" Value="False">

<Setter TargetName="CheckboxCheckMark" Property="Visibility" Value="Collapsed"/>

</Trigger>

</ControlTemplate.Triggers>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

</Page.Resources>

<Border CornerRadius="5" Background="{DynamicResource SecundaryBackgroundColor}">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="auto" MaxWidth="245"/>

</Grid.ColumnDefinitions>

<TextBlock Text="История поездок" Grid.Column="0" FontSize ="25" Foreground="{DynamicResource TextSecundaryColor}" Margin="10,10,10,10"/>

<ScrollViewer Grid.Column="0" Margin="0 50 0 0">

<StackPanel x:Name="HistoryPanel" Orientation="Vertical"/>

</ScrollViewer>

<Grid Grid.Column="1">

<Grid.RowDefinitions>

<RowDefinition Height="auto"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<Grid Grid.Row="0">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<TextBlock Text="Фильтры" Grid.Column="0" FontSize ="25" Foreground="{DynamicResource TextSecundaryColor}" Margin="5 10 10 10"/>

<Button Grid.Column="1" Margin="10" BorderThickness="0" Click="AcceptFilter\_Click" Width="42" Height="34">

<Button.Template>

<ControlTemplate TargetType="Button">

<Grid>

<Border Background="#006400" CornerRadius="10">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

<iconPacks:PackIconUnicons Kind="Search" Foreground="{DynamicResource TextSecundaryColor}" HorizontalAlignment="Center" VerticalAlignment="Center" />

</Grid>

</ControlTemplate>

</Button.Template>

</Button>

<Button Grid.Column="2" Margin="10" BorderThickness="0" Width="42" Height="34" Click="ClearFilter\_Click">

<Button.Template>

<ControlTemplate TargetType="Button">

<Grid>

<Border Background="#FFCA1E1E" CornerRadius="10">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

<iconPacks:PackIconUnicons Kind="Times" Foreground="{DynamicResource TextSecundaryColor}" HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Grid>

</ControlTemplate>

</Button.Template>

</Button>

</Grid>

<ScrollViewer Grid.Row="1">

<StackPanel x:Name="FilterPanel">

<TextBlock Text="Марка" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="5 0 0 0"/>

<ComboBox x:Name="comboBoxCar" FontSize="15"/>

<TextBlock Text="Точка отправки" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="5 0 0 0"/>

<TextBox x:Name="PointOne" Tag="Город" FontSize="15" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="5 0 0 0"/>

<TextBlock Text="Точка прибытия" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="5 0 0 0"/>

<TextBox x:Name="PointTwo" Tag="Город" FontSize="15" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="5 0 0 0"/>

<TextBlock Text="Дата поездки" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="5 0 0 0"/>

<StackPanel Orientation="Horizontal">

<TextBox x:Name="DateFrom" Tag="c" FontSize="15" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="5 0 0 0" Width="100"/>

<TextBlock Text="-" FontSize ="15" Foreground="Gray" HorizontalAlignment="Center"/>

<TextBox x:Name="DateTo" Tag="до" FontSize="15" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="5 0 0 0" Width="100"/>

</StackPanel>

<TextBlock Text="Стоимость поездки" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="5 0 0 0"/>

<StackPanel Orientation="Horizontal">

<TextBox x:Name="PriceFrom" Tag="c" FontSize="15" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="5 0 0 0" Width="100" PreviewTextInput="PriceFrom\_PreviewTextInput"/>

<TextBlock Text="-" FontSize ="15" Foreground="Gray" HorizontalAlignment="Center"/>

<TextBox x:Name="PriceTo" Tag="до" FontSize="15" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="5 0 0 0" Width="100" PreviewTextInput="PriceTo\_PreviewTextInput"/>

</StackPanel>

</StackPanel>

</ScrollViewer>

</Grid>

</Grid>

</Border>

</Page>

using OneWay.Class;

using OneWay.Controls;

using System;

using System.Collections.Generic;

using System.Globalization;

using System.Linq;

using System.Text;

using System.Windows;

using System.Windows.Controls;

namespace OneWay.Pages

{

public partial class History : Page

{

public int IdUser;

DataBase db = new DataBase("OneWayDB.db");

List<Trip> allTrip = new List<Trip>();

List<Car> cars = new List<Car>();

public History(int IdUser)

{

InitializeComponent();

comboBoxCar.Items.Clear();

cars = db.GetAllCars(IdUser);

foreach (Car car in cars)

{

string carName = $"{car.Brand} {car.Model} {car.Equipment}";

comboBoxCar.Items.Add(carName);

}

this.IdUser = IdUser;

allTrip = db.GetAllTripByUserId(IdUser);

CreateItem();

}

private void CreateItem()

{

HistoryPanel.Children.Clear();

foreach (var item in allTrip)

{

try

{

var info = db.GetRouteInfoById(item.IdRoute);

string startPoint = info.Item1;

string finishPoint = info.Item2;

double distance = info.Item3;

string travelTime = info.Item4;

string points = info.Item5;

TripItem trip = new TripItem();

trip.TripName = startPoint + " - " + finishPoint;

string formattedPoints = "";

if (!string.IsNullOrEmpty(points))

{

string[] point = points.Split('#');

formattedPoints += string.Join("\n", point);

}

trip.Point = formattedPoints;

TimeSpan date = TimeSpan.FromMinutes(Convert.ToDouble(travelTime));

if (date.Days >= 1)

{

if (date.Hours == 0)

{

trip.Time = $"{date.Days} д {date.Minutes} мин";

}

else

{

trip.Time = $"{date.Days} д {date.Hours} ч {date.Minutes} мин";

}

}

else

{

if (date.Hours != 0)

{

trip.Time = $"{date.Hours} ч {date.Minutes} мин";

}

else

{

trip.Time = $"{date.Minutes} мин";

}

}

trip.Distance = distance + " км";

trip.Id = item.IdTrip.ToString();

Car car = db.GetCar(item.IdCar);

if (car.Brand != null)

{

trip.Car = car.Brand;

trip.CarModel = car.Model;

}

else

{

trip.Car = "неизвестный";

trip.CarModel = "данный автомобиль был удален";

}

trip.ReportButtonClicked += TripItem\_ReportButtonClicked;

trip.InfoButtonClicked += TripItem\_InfoButtonClicked;

trip.DeleteButtonClicked += TripItem\_DeleteButtonClicked;

HistoryPanel.Children.Add(trip);

}

catch

{

}

}

}

private void AcceptFilter\_Click(object sender, RoutedEventArgs e)

{

StringBuilder filterQuery = new StringBuilder("SELECT \* FROM Trips WHERE ");

// автомобиль

if (comboBoxCar.SelectedItem != null)

{

Car selectedCar = cars.FirstOrDefault(item => $"{item.Brand} {item.Model} {item.Equipment}" == comboBoxCar.SelectedItem.ToString());

if (selectedCar != null)

{

filterQuery.Append($"IdCar = {selectedCar.IdCar} AND ");

}

}

// точки

int startId = db.GetCityIdByName(IdUser, PointOne.Text);

int finishId = db.GetCityIdByName(IdUser, PointTwo.Text);

if (startId != -1)

{

filterQuery.Append($"IdRoute IN (SELECT IdRoute FROM Routes WHERE PointOne = {startId}) AND ");

}

else if (!string.IsNullOrWhiteSpace(PointOne.Text))

{

CustomMessageBox.Show("Уведомление", "Точка отправления не найдена. Проверьте название и повторите попытку", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

return;

}

if (finishId != -1)

{

filterQuery.Append($"IdRoute IN (SELECT IdRoute FROM Routes WHERE PointTwo = {finishId}) AND ");

}

else if (!string.IsNullOrWhiteSpace(PointTwo.Text))

{

CustomMessageBox.Show("Уведомление", "Точка прибытия не найдена. Проверьте название и повторите попытку", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

return;

}

// дата

DateTime date1, date2;

if (DateTime.TryParseExact(DateFrom.Text, "dd.MM.yyyy", CultureInfo.InvariantCulture, DateTimeStyles.None, out date1))

{

filterQuery.Append($"StartTime >= '{date1.ToString("yyyy-MM-dd HH:mm:ss")}' AND ");

}

else if (DateFrom.Text != "")

{

CustomMessageBox.Show("Уведомление", $"Введите корректную дату 1 в формате {DateTime.Now.ToString("dd.MM.yyyy")} и повторите попытку. Проверьте название и повторите попытку", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

return;

}

if (DateTime.TryParseExact(DateTo.Text, "dd.MM.yyyy", CultureInfo.InvariantCulture, DateTimeStyles.None, out date2))

{

date2 = date2.Date.AddHours(23).AddMinutes(59).AddSeconds(59);

filterQuery.Append($"StartTime <= '{date2.ToString("yyyy-MM-dd HH:mm:ss")}' AND ");

}

else if (DateTo.Text != "")

{

CustomMessageBox.Show("Уведомление", $"Введите корректную дату 2 в формате {DateTime.Now.ToString("dd.MM.yyyy")} и повторите попытку. Проверьте название и повторите попытку", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

return;

}

// цена

double priceOne, priceTwo;

if (double.TryParse(PriceFrom.Text, out priceOne))

{

filterQuery.Append($"UsedMoney >= {priceOne.ToString("0.00", CultureInfo.InvariantCulture)} AND ");

}

if (double.TryParse(PriceTo.Text, out priceTwo))

{

filterQuery.Append($"UsedMoney <= {priceTwo.ToString("0.00", CultureInfo.InvariantCulture)} AND ");

}

if (filterQuery.ToString().EndsWith(" AND "))

{

filterQuery.Remove(filterQuery.Length - 5, 5);

}

string combinedQuery = filterQuery.ToString();

if(combinedQuery != "SELECT \* FROM Trips WHERE ")

{

allTrip = db.GetHistory(combinedQuery);

CreateItem();

}

else

{

CustomMessageBox.Show("Уведомление", "Отображены все маршруты.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

allTrip = db.GetAllTripByUserId(IdUser);

CreateItem();

}

}

private void ClearFilter\_Click(object sender, RoutedEventArgs e)

{

foreach (var child in FilterPanel.Children)

{

if (child is TextBox textBox && !string.IsNullOrEmpty(textBox.Name))

{

textBox.Text = string.Empty;

}

if (child is StackPanel stackPanel)

{

foreach (var childTwo in stackPanel.Children)

{

if (childTwo is TextBox textBoxTwo && !string.IsNullOrEmpty(textBoxTwo.Name))

{

textBoxTwo.Text = string.Empty;

}

}

}

}

comboBoxCar.SelectedIndex = -1;

}

private void TripItem\_ReportButtonClicked(object sender, EventArgs e)

{

if (sender is TripItem tripItem)

{

string tripName = tripItem.TripName;

string distance = tripItem.Distance;

string time = tripItem.Time;

string car = tripItem.Car;

string id = tripItem.Id;

string point = tripItem.Point;

Report reportPage = new Report(IdUser, Convert.ToInt32(tripItem.Id));

Window parentWindow = Window.GetWindow(this);

if (parentWindow != null && parentWindow is MainWindow mainWindow && mainWindow.fContainer != null)

{

mainWindow.fContainer.Content = reportPage;

}

}

}

private void TripItem\_InfoButtonClicked(object sender, EventArgs e)

{

if (sender is TripItem tripItem)

{

var messageBoxResult = CustomMessageBox.Show("Уведомление", $"Желаете посмотреть информаци о поездке?", MessageBoxButton.YesNo, CustomMessageBox.MessageBoxImage.Information);

if (messageBoxResult == MessageBoxResult.Yes)

{

TripInfoPage reportPage = new TripInfoPage(IdUser, Convert.ToInt32(tripItem.Id));

Window parentWindow = Window.GetWindow(this);

if (parentWindow != null && parentWindow is MainWindow mainWindow && mainWindow.fContainer != null)

{

mainWindow.fContainer.Content = reportPage;

}

}

}

}

private void TripItem\_DeleteButtonClicked(object sender, EventArgs e)

{

var quequestion = CustomMessageBox.Show("Уведомление", $"Желаете удалить данную поездку?", MessageBoxButton.YesNo, CustomMessageBox.MessageBoxImage.Information);

if (quequestion == MessageBoxResult.Yes)

{

if (sender is TripItem tripItem)

{

int id = Convert.ToInt32(tripItem.Id);

if (db.DeleteTrip(id))

{

CustomMessageBox.Show("Уведомление", $"Данная поездка была удалена из истории.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

allTrip = db.GetAllTripByUserId(IdUser);

CreateItem();

}

else

{

CustomMessageBox.Show("Уведомление", $"Данная поездка не была удалена из истории.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

}

}

}

}

private void PriceFrom\_PreviewTextInput(object sender, System.Windows.Input.TextCompositionEventArgs e)

{

if (!char.IsDigit(e.Text, e.Text.Length - 1))

{

e.Handled = true;

return;

}

string currentText = PriceFrom.Text;

int selectionStart = PriceFrom.SelectionStart;

string newText = currentText.Insert(selectionStart, e.Text);

if (int.TryParse(newText, out int price))

{

if (price <= 0)

{

CustomMessageBox.Show("Уведомление", $"Сумма затрат должна быть больше 0", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

e.Handled = true;

}

}

}

private void PriceTo\_PreviewTextInput(object sender, System.Windows.Input.TextCompositionEventArgs e)

{

if (!char.IsDigit(e.Text, e.Text.Length - 1))

{

e.Handled = true;

return;

}

string currentText = PriceTo.Text;

int selectionStart = PriceTo.SelectionStart;

string newText = currentText.Insert(selectionStart, e.Text);

if (int.TryParse(newText, out int price))

{

if (price <= 0)

{

CustomMessageBox.Show("Уведомление", $"Сумма затрат должна быть больше 0", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

e.Handled = true;

}

}

}

}

}

<Page x:Class="OneWay.Pages.Home"

xmlns:ctls="clr-namespace:OneWay.Controls"

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:iconPacks="http://metro.mahapps.com/winfx/xaml/iconpacks"

xmlns:lvc="clr-namespace:LiveCharts.Wpf;assembly=LiveCharts.Wpf"

xmlns:local="clr-namespace:OneWay.Pages"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800"

Title="Home">

<Page.Resources>

<Style x:Key="FuelTypeStyle" TargetType="{x:Type TextBlock}">

<Setter Property="Foreground" Value="{DynamicResource SecundaryIconColor}"/>

<Setter Property="VerticalAlignment" Value="Center"/>

<Setter Property="FontSize" Value="20"/>

<Setter Property="FontWeight" Value="Regular"/>

<Setter Property="Margin" Value="18 0 0 0"/>

<Style.Triggers>

<DataTrigger Binding="{Binding IsMouseOver, RelativeSource={RelativeSource Mode=FindAncestor, AncestorType=ToggleButton}}" Value="True">

<Setter Property="Foreground" Value="{DynamicResource TextFourthColor}"/>

</DataTrigger>

<DataTrigger Binding="{Binding IsChecked, RelativeSource={RelativeSource Mode=FindAncestor, AncestorType=ToggleButton}}" Value="True">

<Setter Property="Foreground" Value="{DynamicResource PrimaryBlueColor}"/>

</DataTrigger>

</Style.Triggers>

</Style>

<Style x:Key="ScrollBarTrackThumb"

TargetType="{x:Type Thumb}">

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="{x:Type Thumb}">

<Grid x:Name="Grid">

<Rectangle HorizontalAlignment="Stretch"

VerticalAlignment="Stretch"

Width="Auto"

Height="Auto"

Fill="Transparent" />

<Border x:Name="CornerScrollBarRectangle"

CornerRadius="5"

HorizontalAlignment="Stretch"

VerticalAlignment="Stretch"

Width="Auto"

Height="Auto"

Margin="0,1,0,1"

Background="{TemplateBinding Background}" />

</Grid>

<ControlTemplate.Triggers>

<Trigger Property="Tag"

Value="Horizontal">

<Setter TargetName="CornerScrollBarRectangle"

Property="Width"

Value="Auto" />

<Setter TargetName="CornerScrollBarRectangle"

Property="Height"

Value="6" />

</Trigger>

</ControlTemplate.Triggers>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

</Page.Resources>

<Border CornerRadius="5" Background="{DynamicResource SecundaryBackgroundColor}">

<!-- Главная таблица-->

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="225"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<Grid Grid.Row="0">

<Grid.RowDefinitions>

<RowDefinition Height="80"/>

<RowDefinition Height="2\*"/>

</Grid.RowDefinitions>

<!-- Первая строка-->

<Grid Grid.Row="0">

<TextBlock Text="Аккаунт" FontSize="30" Foreground="{DynamicResource TextSecundaryColor}" Margin="20 20 20 20 "/>

<TextBlock x:Name ="clock" FontSize="30" Foreground="{DynamicResource TextSecundaryColor}" HorizontalAlignment="Right" Margin="20 20 20 20 "/>

</Grid>

<!-- Вторая строка -->

<Grid Grid.Row="1">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="2\*"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<!-- Поле пользователя -->

<Grid Grid.Row="0">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="140"/>

<ColumnDefinition Width="1\*"/>

</Grid.ColumnDefinitions>

<StackPanel Grid.Column="0">

<iconPacks:PackIconUnicons Kind="User" Foreground="{DynamicResource TextSecundaryColor}" Height="100" Width="100" Margin="20 20 20 20" />

</StackPanel>

<StackPanel Grid.Column="1" Margin="0 20 0 0">

<TextBlock x:Name="timeOfDay" Text="{Binding TimeOfDay}" FontSize="25" Foreground="{DynamicResource TextSecundaryColor}" Margin="0"/>

<TextBlock x:Name="name" Text="{Binding UserName, FallbackValue=unknowUser}" FontSize="25" Foreground="{DynamicResource TextSecundaryColor}" Margin="0"/>

</StackPanel>

</Grid>

<!-- Поле первого столбца информации-->

<Grid Grid.Column="1" Margin="0">

<Grid.RowDefinitions>

<RowDefinition Height="70"/>

<RowDefinition Height="70"/>

</Grid.RowDefinitions>

<!-- Первый блок информации -->

<Grid Grid.Row="0">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="40"/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<StackPanel Grid.Column="0">

<Viewbox Height="20" Width="20" Margin="0 30 0 20 " >

<iconPacks:PackIconUnicons Kind="LocationArrow" Foreground="#49C5BB" Height="20" Width="20"/>

</Viewbox>

</StackPanel>

<StackPanel Grid.Column="1">

<TextBlock x:Name="trip" Text="{Binding Trip,FallbackValue=0}" FontSize="25" FontWeight="Bold" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 20 0 0"/>

<TextBlock x:Name="tripDescription" Text="{Binding TripDescription}" FontSize="15" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 -10 0 0"/>

</StackPanel>

</Grid>

<!-- Второй блок информации -->

<Grid Grid.Row="1">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="40"/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<StackPanel Grid.Column="0">

<Viewbox Height="20" Width="20" Margin="0 15 0 20 " >

<iconPacks:PackIconUnicons Kind="Wallet" Foreground="#e13228"/>

</Viewbox>

</StackPanel>

<StackPanel Grid.Column="1">

<TextBlock x:Name = "money" Text="{Binding Money, FallbackValue=0}" FontSize="25" FontWeight="Bold" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 5 0 0"/>

<TextBlock x:Name = "moneyDescription" Text="{Binding MoneyDescription}" FontSize="15" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 -5 0 0"/>

</StackPanel>

</Grid>

</Grid>

<!-- Поле третьего столбца информации-->

<Grid Grid.Column="2">

<Grid.RowDefinitions>

<RowDefinition Height="70"/>

<RowDefinition Height="70"/>

</Grid.RowDefinitions>

<!-- Первый блок информации -->

<Grid Grid.Row="0">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="40"/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<StackPanel Grid.Column="0">

<Viewbox Height="20" Width="20" Margin="0 30 0 20 " >

<iconPacks:PackIconUnicons Kind="Favorite" Foreground="#FFD700" Height="20" Width="20"/>

</Viewbox>

</StackPanel>

<StackPanel Grid.Column="1">

<TextBlock x:Name="favoriteCar" Text="{Binding FavoriteCar, FallbackValue=0}" FontSize="25" FontWeight="Bold" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 20 0 0"/>

<TextBlock x:Name="favoriteCarDescription" Text="{Binding FavoriteCarDescription}" FontSize="15" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 -10 0 0"/>

</StackPanel>

</Grid>

<!-- Второй блок информации -->

<Grid Grid.Row="1">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="40"/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<StackPanel Grid.Column="0">

<Viewbox Height="20" Width="20" Margin="0 15 0 20 " >

<iconPacks:PackIconRemixIcon Kind="PinDistanceLine" Foreground="#F2FFFF" Height="20" Width="20"/>

</Viewbox>

</StackPanel>

<StackPanel Grid.Column="1">

<TextBlock x:Name = "distanceMonth" Text="{Binding DistanceMonth, FallbackValue=0}" FontSize="25" FontWeight="Bold" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 5 0 0"/>

<TextBlock x:Name = "distanceMonthDescription" Text="{Binding DistanceMonthDescription}" FontSize="15" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 -5 0 0"/>

</StackPanel>

</Grid>

</Grid>

</Grid>

</Grid>

<Grid Grid.Row="1">

<Grid.RowDefinitions>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<!-- Панель моих автмобилей-->

<TextBlock Text="Мои автомобили" Grid.Column="0" Margin="10 0 0 0" FontSize="25" Foreground="{DynamicResource TextSecundaryColor}"></TextBlock>

<ScrollViewer VerticalScrollBarVisibility="Auto" Margin="0 40 0 0" Grid.Column="0">

<StackPanel x:Name="MyCar" Margin="0"/>

</ScrollViewer>

<!-- Панель послдених поездок-->

<TextBlock Text="Последние поездки" Foreground="{DynamicResource TextSecundaryColor}" FontSize="25" Margin="10 0 0 0" Grid.Column="1"/>

<ScrollViewer VerticalScrollBarVisibility="Auto" Margin="0 40 0 0" Grid.Column="1" >

<StackPanel x:Name="MyTrip" Margin="0"/>

</ScrollViewer>

</Grid>

</Grid>

</Border>

</Page>

using LiveCharts;

using OneWay.Class;

using OneWay.Controls;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Windows.Controls;

using System.Windows.Threading;

namespace OneWay.Pages

{

public class HomeViewModel : INotifyPropertyChanged

{

public event PropertyChangedEventHandler PropertyChanged;

private string \_timeOfDay;

public string TimeOfDay

{

get => \_timeOfDay;

set

{

\_timeOfDay = value;

OnPropertyChanged(nameof(TimeOfDay));

}

}

private int \_trip;

public int Trip

{

get => \_trip;

set

{

\_trip = value;

OnPropertyChanged(nameof(Trip));

TripDescription = GetPluralString(Convert.ToInt32(Trip), "поездка за месяц", "поездки за месяц", "поездок за месяц");

}

}

private string \_tripDescription;

public string TripDescription

{

get => \_tripDescription;

set

{

\_tripDescription = value;

OnPropertyChanged(nameof(TripDescription));

}

}

private int \_favoriteCar;

public int FavoriteCar

{

get => \_favoriteCar;

set

{

\_favoriteCar = value;

OnPropertyChanged(nameof(FavoriteCar));

FavoriteCarDescription = GetPluralString(Convert.ToInt32(FavoriteCar), "избранное авто", "избранных авто", "избранных авто");

}

}

private string \_favoriteCarDescription;

public string FavoriteCarDescription

{

get => \_favoriteCarDescription;

set

{

\_favoriteCarDescription = value;

OnPropertyChanged(nameof(FavoriteCarDescription));

}

}

private int \_distanceMonth;

public int DistanceMonth

{

get => \_distanceMonth;

set

{

\_distanceMonth = value;

OnPropertyChanged(nameof(DistanceMonth));

DistanceMonthDescription = GetPluralString(Convert.ToInt32(DistanceMonth), "преодален за месяц", "преодалено за месяц", "преодалено за месяц");

}

}

private string \_distanceMonthDescription;

public string DistanceMonthDescription

{

get => \_distanceMonthDescription;

set

{

\_distanceMonthDescription = value;

OnPropertyChanged(nameof(DistanceMonthDescription));

}

}

private int \_money;

public int Money

{

get => \_money;

set

{

\_money = value;

OnPropertyChanged(nameof(Money));

MoneyDescription = GetPluralString(Convert.ToInt32(Money), "потрачен за месяц", "потрачено за месяц", "потрачено за месяц");

}

}

private string \_moneyDescription;

public string MoneyDescription

{

get => \_moneyDescription;

set

{

\_moneyDescription = value;

OnPropertyChanged(nameof(MoneyDescription));

}

}

public HomeViewModel()

{

UpdateAll();

}

public void UpdateAll()

{

TimeOfDay = GetTimeOfDay(DateTime.Now);

TripDescription = GetPluralString(Convert.ToInt32(Trip), "поездка за месяц", "поездки за месяц", "поездкок за месяц");

FavoriteCarDescription = GetPluralString(Convert.ToInt32(FavoriteCar), "избранное авто", "избранных авто", "избранных авто");

DistanceMonthDescription = GetPluralString(Convert.ToInt32(DistanceMonth), "преодален за месяц", "преодалено за месяц", "преодалено за месяц");

MoneyDescription = GetPluralString(Convert.ToInt32(Money), "потрачен за месяц", "потрачено за месяц", "потрачено за месяц");

}

private string GetTimeOfDay(DateTime date)

{

int hour = date.Hour;

if (hour >= 0 && hour < 6)

{

return "Доброй ночи!";

}

else if (hour >= 6 && hour < 12)

{

return "Доброго утра!";

}

else if (hour >= 12 && hour < 18)

{

return "Доброго дня!";

}

else

{

return "Доброго вечера!";

}

}

private string GetPluralString(int count, string singular, string plural, string pluralMany)

{

if (count < 0)

return "";

int lastDigit = count % 10;

int lastTwoDigits = count % 100;

if (lastTwoDigits >= 11 && lastTwoDigits <= 14)

{

return pluralMany;

}

switch (lastDigit)

{

case 1:

return singular;

case 2:

case 3:

case 4:

return plural;

default:

return pluralMany;

}

}

protected virtual void OnPropertyChanged(string propertyName)

{

PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));

}

}

public partial class Home : Page

{

public SeriesCollection Data { get; set; }

DataBase db = new DataBase("OneWayDB.db");

List<Car> allCars = new List<Car>();

List<Trip> allTrip = new List<Trip>();

public int IdUser;

public string UserName;

public Home(int IdUser)

{

InitializeComponent();

this.IdUser = IdUser;

HomeViewModel ViewModel = new HomeViewModel();

DataContext = ViewModel;

StartTimer();

allCars = db.GetAllCars(IdUser);

CreateItemsCar(allCars);

this.name.Text = db.GetFirstNameById(IdUser);

allTrip = db.GetAllTripByUserId(IdUser);

CreateItemsTrip(allTrip);

var data = db.GetCountAndSpendMoney(IdUser);

trip.Text = data.Item1.ToString();

((HomeViewModel)DataContext).Trip = data.Item1;

money.Text = data.Item2.ToString() + " руб";

((HomeViewModel)DataContext).Money = Convert.ToInt32(data.Item2);

var count = db.GetFavoriteCarCount(IdUser);

favoriteCar.Text = count.ToString();

((HomeViewModel)DataContext).FavoriteCar = count;

var info = db.GetDistanceByMonth(IdUser);

distanceMonth.Text = info.ToString() + " км";

((HomeViewModel)DataContext).DistanceMonth = Convert.ToInt32(info);

}

private void CreateItemsCar(List<Car> cars)

{

foreach (var item in cars)

{

ShortCarItem car = new ShortCarItem();

car.CarName = $"{item.Brand} {item.Model}";

car.Seat = item.Seats.ToString();

if (item.Volume.ToString() == "0")

{

car.Volume = $"{item.EngineType}";

}

else

{

car.Volume = $"{Math.Round(item.Volume.Value / 1000, 2)} / {item.Fuel}";

}

car.Body = item.Body;

car.Transmission = item.GearBox;

MyCar.Children.Add(car);

}

}

private void CreateItemsTrip(List<Class.Trip> trips)

{

foreach (var item in trips)

{

try

{

var info = db.GetRouteInfoById(item.IdRoute);

string startPoint = info.Item1;

string finishPoint = info.Item2;

double distance = info.Item3;

string travelTime = info.Item4;

string points = info.Item5;

ShortTripItem trip = new ShortTripItem();

trip.Trip = startPoint + " - " + finishPoint;

TimeSpan date = TimeSpan.FromMinutes(Convert.ToDouble(travelTime));

if (date.Days >= 1)

{

if (date.Hours == 0)

{

trip.Time = $"{date.Days} д {date.Minutes} мин";

}

else

{

trip.Time = $"{date.Days} д {date.Hours} ч {date.Minutes} мин";

}

}

else

{

if (date.Hours != 0)

{

trip.Time = $"{date.Hours} ч {date.Minutes} мин";

}

else

{

trip.Time = $"{date.Minutes} мин";

}

}

trip.Distanse = distance + " км";

Car car = db.GetCar(item.IdCar);

if (car.Brand != null)

{

trip.Car = car.Brand;

}

else

{

trip.Car = "неизвестный";

}

MyTrip.Children.Add(trip);

}

catch

{

}

}

}

public void StartTimer()

{

DispatcherTimer timer = new DispatcherTimer();

timer.Interval = TimeSpan.FromSeconds(1);

timer.Tick += Timer\_Tick;

timer.Start();

}

private void Timer\_Tick(object sender, EventArgs e)

{

string timeNow = DateTime.Now.ToString("HH:mm:ss");

clock.Text = timeNow;

}

}

}

<Page x:Class="OneWay.Pages.RegistrationPage"

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:iconPacks="http://metro.mahapps.com/winfx/xaml/iconpacks"

xmlns:local="clr-namespace:OneWay.Pages"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800"

Title="RegistrationPage">

<Border CornerRadius="5" Background="{DynamicResource SecundaryBackgroundColor}">

<Border Height="350" Width="350" Background="{DynamicResource SecundaryBackgroundColor}" BorderBrush="{DynamicResource TextSecundaryColor}" BorderThickness="2" CornerRadius="10">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="200"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="auto"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<Button x:Name="AuthBtn" Grid.Column="0" Grid.Row="0" BorderThickness="0" Width="34" Height="34" HorizontalAlignment="Center" VerticalAlignment="Center" Click="AuthBtn\_Click">

<Button.Template>

<ControlTemplate TargetType="{x:Type Button}">

<Grid>

<Border Background="{DynamicResource TextSecundaryColor}" CornerRadius="10">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

<iconPacks:PackIconUnicons x:Name="icon" Kind="AngleLeft" Foreground="{DynamicResource SecundaryBackgroundColor}" HorizontalAlignment="Center" VerticalAlignment="Center" />

</Grid>

</ControlTemplate>

</Button.Template>

</Button>

<TextBlock Grid.Row="0" Grid.Column="1" Text="Регистрация" FontSize ="25" Foreground="{DynamicResource TextSecundaryColor}" Margin="10,10,10,10" TextAlignment="Center"/>

<TextBlock Grid.Row="1" Grid.Column="1" Text="имя" FontSize ="25" TextAlignment="Center" Foreground="{DynamicResource TextSecundaryColor}" Margin="10,0,10,0"/>

<TextBox Grid.Row="2" Grid.Column="1" x:Name="Name" Height="30" TextAlignment="Center" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0 0 0 1.5" BorderBrush="{DynamicResource TextSecundaryColor}"/>

<TextBlock Grid.Row="3" Grid.Column="1" Text="логин" FontSize ="25" TextAlignment="Center" Foreground="{DynamicResource TextSecundaryColor}" Margin="10,00,10,00"/>

<TextBox Grid.Row="4" Grid.Column="1" x:Name="Login" Height="30" TextAlignment="Center" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0 0 0 1.5" BorderBrush="{DynamicResource TextSecundaryColor}"/>

<TextBlock Grid.Row="5" Grid.Column="1" Text="пароль" FontSize ="25" TextAlignment="Center" Foreground="{DynamicResource TextSecundaryColor}" Margin="10,0,10,0"/>

<TextBox Grid.Row="6" Grid.Column="1" x:Name="PasswordTextBox" Height="30" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0 0 0 1.5" BorderBrush="{DynamicResource TextSecundaryColor}" Visibility="Collapsed" TextChanged="PasswordTextBox\_TextChanged"/>

<PasswordBox Grid.Row="6" Grid.Column="1" x:Name="Password" Height="30" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0 0 0 1.5" BorderBrush="{DynamicResource TextSecundaryColor}" Visibility="Visible"/>

<Button Grid.Row="6" Grid.Column="2" BorderThickness="0" Width="34" Height="34" Click="Button\_Click" HorizontalAlignment="Left" Margin="10,0,0,0">

<Button.Template>

<ControlTemplate TargetType="Button">

<Grid>

<Border Background="{DynamicResource TextSecundaryColor}" CornerRadius="10">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

<iconPacks:PackIconFontAwesome x:Name="icon" Kind="EyeSlashSolid" Foreground="{DynamicResource SecundaryBackgroundColor}" HorizontalAlignment="Center" VerticalAlignment="Center" />

</Grid>

</ControlTemplate>

</Button.Template>

</Button>

<Button Grid.Row="7" Grid.Column="1" x:Name="RegButton" Content="Зарегистрироваться" FontSize="20" Height="30" Click="RegButton\_Click" >

<Button.Template>

<ControlTemplate TargetType="Button">

<Grid>

<Border Background="{DynamicResource TextSecundaryColor}" CornerRadius="10">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

</Grid>

</ControlTemplate>

</Button.Template>

</Button>

</Grid>

</Border>

</Border>

</Page>

using MahApps.Metro.IconPacks;

using OneWay.Class;

using OneWay.Controls;

using System;

using System.Windows;

using System.Windows.Controls;

namespace OneWay.Pages

{

public partial class RegistrationPage : Page

{

DataBase db = new DataBase("OneWayDB.db");

public RegistrationPage()

{

InitializeComponent();

}

private void RegButton\_Click(object sender, RoutedEventArgs e)

{

if (string.IsNullOrWhiteSpace(Name.Text) ||

string.IsNullOrWhiteSpace(Login.Text) ||

string.IsNullOrWhiteSpace(Password.Password)) // проверка заполения полей

{

CustomMessageBox.Show("Уведомление", $"Заполните все поля и повторите попытку.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

return;

}

User user = new User

{

FirstName = Name.Text,

Login = Login.Text,

Password = AesEncryption.EncryptString(Password.Password),

RegistrationDate = DateTime.Now.ToString("dd.MM.yyyy")

};

if(db.InsertUser(user))

{

CustomMessageBox.Show("Уведомление", $"Пользователь добавлен.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

NavigationService?.GoBack();

}

else

{

CustomMessageBox.Show("Уведомление", $"Пользователь не добавлен. Возможно данный логин уже занят. Повторите попытку.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

}

}

private void Button\_Click(object sender, RoutedEventArgs e)

{

Button button = sender as Button;

if (button != null)

{

ControlTemplate template = button.Template;

PackIconFontAwesome icon = template.FindName("icon", button) as PackIconFontAwesome;

if (icon != null)

{

if (Password.Visibility == Visibility.Visible)

{

PasswordTextBox.Visibility = Visibility.Visible;

Password.Visibility = Visibility.Collapsed;

PasswordTextBox.Text = Password.Password;

icon.Kind = PackIconFontAwesomeKind.EyeSolid;

}

else

{

PasswordTextBox.Visibility = Visibility.Collapsed;

Password.Visibility = Visibility.Visible;

Password.Password = PasswordTextBox.Text;

icon.Kind = PackIconFontAwesomeKind.EyeSlashSolid;

}

}

}

}

private void AuthBtn\_Click(object sender, RoutedEventArgs e)

{

NavigationService?.GoBack();

}

private void PasswordTextBox\_TextChanged(object sender, TextChangedEventArgs e)

{

Password.Password = PasswordTextBox.Text;

}

}

}<Page x:Class="OneWay.Pages.Report"

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:OneWay.Pages"

xmlns:iconPacks="http://metro.mahapps.com/winfx/xaml/iconpacks"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800"

Title="ReportPage">

<Border CornerRadius="5" Background="{DynamicResource SecundaryBackgroundColor}">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="3\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<TextBlock Text="Отчет" Grid.Column="0" FontSize ="25" Foreground="{DynamicResource TextSecundaryColor}" Margin="10,10,10,10"/>

<ScrollViewer Margin="0 50 0 0">

<StackPanel>

<Grid Grid.Column="0" x:Name="GridField">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="1.5\*"/>

<ColumnDefinition Width="2\*"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

</Grid.RowDefinitions>

<!-- Первая строка -->

<TextBlock Grid.Row="0" Grid.Column="0" x:Name="TripText" Text="Поездка" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0"/>

<ComboBox Grid.Row="0" Grid.Column="1" x:Name="comboBoxTrip" Height="25" FontSize="18" Margin="10 0 0 10" SelectionChanged="comboBoxTrip\_SelectionChanged"/>

<!--Вторая строка-->

<TextBlock Grid.Row="1" Grid.Column="0" x:Name="TextDate" Text="Дата поездок" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0" Visibility="Visible"/>

<WrapPanel Grid.Row="1" Grid.Column="1">

<StackPanel x:Name="StackData" Orientation="Horizontal" VerticalAlignment="Center" Visibility="Visible" Margin="10 0 0 0">

<TextBox x:Name="DateFrom" Tag="c" FontSize="15" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="10 10 0 10" Width="100" />

<TextBlock Text="-" FontSize ="15" Foreground="Gray" HorizontalAlignment="Center" Margin="10 10 0 10"/>

<TextBox x:Name="DateBefore" Tag="по" FontSize="15" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="10 10 0 10" Width="100"/>

</StackPanel>

<Button x:Name="CreateButton" Width="313" Content="Отобразить отчет" FontSize="15" Height="30" Click="CreateButton\_Click" >

<Button.Template>

<ControlTemplate TargetType="Button">

<Grid>

<Border Background="{DynamicResource TextSecundaryColor}" CornerRadius="10">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

</Grid>

</ControlTemplate>

</Button.Template>

</Button>

</WrapPanel>

<!--Четвертая строка-->

<TextBlock Grid.Row="2" Grid.Column="0" x:Name="DistanceText" Text="Расстояние, км" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0"/>

<TextBox Grid.Row="2" Grid.Column="1" x:Name="Distance" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="10 0 0 10" IsReadOnly="True"/>

<!--Пятая строка-->

<TextBlock Grid.Row="3" Grid.Column="0" x:Name="StartPointText" Text="Точка отправления" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0"/>

<TextBox Grid.Row="3" Grid.Column="1" x:Name="StartPoint" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="10 0 0 10" IsReadOnly="True"/>

<TextBlock Grid.Row="4" Grid.Column="0" x:Name="FinishPointText" Text="Точка прибытия" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0"/>

<TextBox Grid.Row="4" Grid.Column="1" Name="FinishPoint" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="10 0 0 10" IsReadOnly="True"/>

<TextBlock Grid.Row="5" Grid.Column="0" Name="TimeText" Text="Время в пути" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0"/>

<TextBox Grid.Row="5" Grid.Column="1" Name="Time" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="10 0 0 10" IsReadOnly="True"/>

<TextBlock Grid.Row="6" Grid.Column="0" x:Name="StartTimeText" Text="Время отправления" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0"/>

<TextBox Grid.Row="6" Grid.Column="1" x:Name="StartTime" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="10 0 0 10" IsReadOnly="True"/>

<TextBlock Grid.Row="7" Grid.Column="0" x:Name="FinishTimeText" Text="Время прибытия" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0"/>

<TextBox Grid.Row="7" Grid.Column="1" x:Name="FinishTime" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="10 0 0 10" IsReadOnly="True"/>

<TextBlock Grid.Row="8" Grid.Column="0" x:Name="TextPoints" Text="Точки остановки" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0" Visibility="Visible"/>

<TextBox Grid.Row="8" Grid.Column="1" x:Name="Points" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="10 0 0 10" Visibility="Visible" IsReadOnly="True" TextWrapping="WrapWithOverflow"/>

<TextBlock Grid.Row="9" Grid.Column="0" x:Name="TextUsedFuel" Text="Использовано топлива, л" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0" TextWrapping="WrapWithOverflow"/>

<TextBox Grid.Row="9" Grid.Column="1" x:Name="UsedFuel" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="10 0 0 10" IsReadOnly="True" />

<TextBlock Grid.Row="10" Grid.Column="0" x:Name="UsedMoneyText" Text="Затраченая сумма, руб" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0"/>

<TextBox Grid.Row="10" Grid.Column="1" x:Name="UsedMoney" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="10 0 0 10" IsReadOnly="True"/>

<TextBlock Grid.Row="11" Grid.Column="0" x:Name="PeopleText" Text="Количество пассажиров" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0"/>

<TextBox Grid.Row="11" Grid.Column="1" x:Name="People" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="10 0 0 10" IsReadOnly="True"/>

<TextBlock Grid.Row="12" Grid.Column="0" x:Name="BrandText" Text="Марка автомобиля" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0"/>

<TextBox Grid.Row="12" Grid.Column="1" x:Name="Brand" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="10 0 0 10" IsReadOnly="True"/>

<TextBlock Grid.Row="13" Grid.Column="0" x:Name="ModelText" Text="Модель автомобиля" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0"/>

<TextBox Grid.Row="13" Grid.Column="1" x:Name="Model" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="10 0 0 10" IsReadOnly="True"/>

<TextBlock Grid.Row="14" Grid.Column="0" x:Name="GenerationText" Text="Поколение автомобиля" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0"/>

<TextBox Grid.Row="14" Grid.Column="1" x:Name="Generation" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="10 0 0 10" IsReadOnly="True"/>

<TextBlock Grid.Row="15" Grid.Column="0" x:Name="EquipmentText" Text="Комплектация автомобиля" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0"/>

<TextBox Grid.Row="15" Grid.Column="1" x:Name="Equipment" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="10 0 0 10" IsReadOnly="True"/>

<TextBlock Grid.Row="16" Grid.Column="0" x:Name="CountText" Text="Количество поездок" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0"/>

<TextBox Grid.Row="16" Grid.Column="1" x:Name="Count" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="10 0 0 10" IsReadOnly="True"/>

<TextBlock Grid.Row="17" Grid.Column="0" x:Name="CountDistanceText" Text="Пройденное расстояние, км" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0"/>

<TextBox Grid.Row="17" Grid.Column="1" x:Name="CountDistance" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="10 0 0 10" IsReadOnly="True"/>

<TextBlock Grid.Row="18" Grid.Column="0" x:Name="AveragePeopleText" Text="Среднее количество пассажиров" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0" TextWrapping="WrapWithOverflow"/>

<TextBox Grid.Row="18" Grid.Column="1" x:Name="AveragePeople" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="10 0 0 10" IsReadOnly="True"/>

<TextBlock Grid.Row="19" Grid.Column="0" x:Name="PopularCarText" Text="Часто используемый автомобиль" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0" TextWrapping="WrapWithOverflow"/>

<TextBox Grid.Row="19" Grid.Column="1" x:Name="PopularCar" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="10 0 0 10" IsReadOnly="True" TextWrapping="WrapWithOverflow"/>

<TextBlock Grid.Row="20" Grid.Column="0" x:Name="PopularRouteText" Text="Часто используемый маршрут" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0" TextWrapping="WrapWithOverflow"/>

<TextBox Grid.Row="20" Grid.Column="1" x:Name="PopularRoute" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="10 0 0 10" IsReadOnly="True"/>

<TextBlock Grid.Row="21" Grid.Column="0" x:Name="CountUseGasolineText" Text="Затрачено бензина, л" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0" TextWrapping="WrapWithOverflow"/>

<TextBox Grid.Row="21" Grid.Column="1" x:Name="CountUseGasoline" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="10 0 0 10" IsReadOnly="True"/>

<TextBlock Grid.Row="22" Grid.Column="0" x:Name="CountUseDieselText" Text="Затрачено дизельного топлива, л" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0" TextWrapping="WrapWithOverflow"/>

<TextBox Grid.Row="22" Grid.Column="1" x:Name="CountUseDiesel" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="10 0 0 10" IsReadOnly="True"/>

<TextBlock Grid.Row="23" Grid.Column="0" x:Name="CountUseElectText" Text="Затрачено электричества, кВт \* ч" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0" TextWrapping="WrapWithOverflow"/>

<TextBox Grid.Row="23" Grid.Column="1" x:Name="CountUseElect" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="10 0 0 10" IsReadOnly="True"/>

<TextBlock Grid.Row="24" Grid.Column="0" x:Name="CountUseGasText" Text="Затрачено газа, л" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0" TextWrapping="WrapWithOverflow"/>

<TextBox Grid.Row="24" Grid.Column="1" x:Name="CountUseGas" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="10 0 0 10" IsReadOnly="True"/>

<TextBlock Grid.Row="25" Grid.Column="0" x:Name="TotalSpentMoneyText" Text="Затрачено на поездки, руб" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0" TextWrapping="WrapWithOverflow"/>

<TextBox Grid.Row="25" Grid.Column="1" x:Name="TotalSpentMoney" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="10 0 0 10" IsReadOnly="True"/>

<StackPanel Grid.Row="26" Orientation="Horizontal">

<Button x:Name="CreateReportButton" Width="160" Content="Создать отчет" FontSize="20" Height="30" Margin="10" Click="CreateReportButton\_Click">

<Button.Template>

<ControlTemplate TargetType="Button">

<Grid>

<Border Background="{DynamicResource TextSecundaryColor}" CornerRadius="10">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

</Grid>

</ControlTemplate>

</Button.Template>

</Button>

</StackPanel>

</Grid>

</StackPanel>

</ScrollViewer>

<StackPanel Grid.Column="1">

<TextBlock Text="Настройки" FontSize ="25" Foreground="{DynamicResource TextSecundaryColor}" Margin="10,10,10,0"/>

<TextBlock Text="Отчет за" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10,10,10,0"/>

<ComboBox Grid.Row="16" Grid.Column="1" x:Name="Fill" FontSize="18" Height="35" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 5 6" SelectionChanged="Fill\_SelectionChanged" >

<ComboBoxItem Content="Поездку"/>

<ComboBoxItem Content="Период"/>

</ComboBox>

</StackPanel>

</Grid>

</Border>

</Page>

using System.Windows.Controls;

using System.Windows;

using OneWay.Class;

using System.Collections.Generic;

using OneWay.Controls;

using System;

using Microsoft.Win32;

using Xceed.Words.NET;

using Xceed.Document.NET;

using System.Globalization;

using System.Text;

using System.Linq;

using System.Data;

using System.Windows.Media;

namespace OneWay.Pages

{

public partial class Report : Page

{

private int IdUser;

private int IdTrip;

DataBase db = new DataBase("OneWayDB.db");

List<Trip> allTrip = new List<Trip>();

List<Trip> filterTrip = new List<Trip>();

Trip selectTrip = new Trip();

List<Trip> tripList = new List<Trip>();

string startData;

string finishData;

public Report(int IdUser, int IdTrip)

{

InitializeComponent();

this.IdUser = IdUser;

this.IdTrip = IdTrip;

Fill.SelectedIndex = 0;

allTrip = db.GetAllTripByUserId(IdUser);

foreach (Trip trip in allTrip)

{

var info = db.GetRouteInfoById(trip.IdRoute);

Car car = db.GetCar(trip.IdCar);

string startPoint = info.Item1;

string finishPoint = info.Item2;

double distance = info.Item3;

string text = $"{startPoint} - {finishPoint} {Convert.ToDateTime(trip.StartTime).ToString("dd.MM.yyyy")}";

text += $"\n{car.Brand} {car.Model}";

text += $"\n{distance} км";

comboBoxTrip.Items.Add(text);

}

comboBoxTrip.Visibility = Visibility.Visible;

for(int i = 0; i < allTrip.Count; i++)

{

if (allTrip[i].IdTrip == IdTrip)

{

comboBoxTrip.SelectedIndex = i;

break;

}

}

}

public Report(int IdUser)

{

InitializeComponent();

this.IdUser = IdUser;

Fill.SelectedIndex = 1;

allTrip = db.GetAllTripByUserId(IdUser);

foreach(Trip trip in allTrip)

{

var info = db.GetRouteInfoById(trip.IdRoute);

Car car = db.GetCar(trip.IdCar);

string startPoint = info.Item1;

string finishPoint = info.Item2;

double distance = info.Item3;

string text = $"{startPoint} - {finishPoint} {Convert.ToDateTime(trip.StartTime).ToString("dd.MM.yyyy")}";

text += $"\n{car.Brand} {car.Model}";

text += $"\n{distance} км";

comboBoxTrip.Items.Add(text);

}

}

private void ClearGridFieldTextBoxes()

{

ClearTextBoxes(GridField);

}

private void ClearTextBoxes(DependencyObject parent)

{

for (int i = 0; i < VisualTreeHelper.GetChildrenCount(parent); i++)

{

var child = VisualTreeHelper.GetChild(parent, i);

if (child is TextBox textBox && !string.IsNullOrEmpty(textBox.Name))

{

textBox.Text = string.Empty;

}

else if (child is Panel panel)

{

ClearTextBoxes(panel);

}

else if (child is ContentControl contentControl && contentControl.Content is DependencyObject content)

{

ClearTextBoxes(content);

}

}

}

private void Fill\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

if(Fill.SelectedIndex == 0)

{

selectTrip = null;

ClearGridFieldTextBoxes();

TripText.Visibility = Visibility.Visible;

comboBoxTrip.Visibility = Visibility.Visible;

StackData.Visibility = Visibility.Collapsed;

TextDate.Visibility = Visibility.Collapsed;

CountText.Visibility = Visibility.Collapsed;

Count.Visibility = Visibility.Collapsed;

CountDistanceText.Visibility = Visibility.Collapsed;

CountDistance.Visibility = Visibility.Collapsed;

AveragePeopleText.Visibility = Visibility.Collapsed;

AveragePeople.Visibility = Visibility.Collapsed;

CreateButton.Visibility = Visibility.Collapsed;

PopularCarText.Visibility = Visibility.Collapsed;

PopularCar.Visibility = Visibility.Collapsed;

PopularRouteText.Visibility = Visibility.Collapsed;

PopularRoute.Visibility = Visibility.Collapsed;

CountUseGasolineText.Visibility = Visibility.Collapsed;

CountUseGasoline.Visibility = Visibility.Collapsed;

CountUseDieselText.Visibility = Visibility.Collapsed;

CountUseDiesel.Visibility = Visibility.Collapsed;

CountUseElectText.Visibility = Visibility.Collapsed;

CountUseElect.Visibility = Visibility.Collapsed;

CountUseGasText.Visibility = Visibility.Collapsed;

CountUseGas.Visibility = Visibility.Collapsed;

TotalSpentMoneyText.Visibility = Visibility.Collapsed;

TotalSpentMoney.Visibility = Visibility.Collapsed;

ShowAllElements();

}

if (Fill.SelectedIndex == 1)

{

filterTrip = new List<Trip>();

ClearGridFieldTextBoxes();

TripText.Visibility = Visibility.Collapsed;

comboBoxTrip.Visibility = Visibility.Collapsed;

StackData.Visibility = Visibility.Visible;

TextDate.Visibility = Visibility.Visible;

CountText.Visibility = Visibility.Visible;

Count.Visibility = Visibility.Visible;

CountDistanceText.Visibility = Visibility.Visible;

CountDistance.Visibility = Visibility.Visible;

AveragePeopleText.Visibility = Visibility.Visible;

AveragePeople.Visibility = Visibility.Visible;

CreateButton.Visibility = Visibility.Visible;

PopularCarText.Visibility = Visibility.Visible;

PopularCar.Visibility = Visibility.Visible;

PopularRouteText.Visibility = Visibility.Visible;

PopularRoute.Visibility = Visibility.Visible;

CountUseGasolineText.Visibility = Visibility.Visible;

CountUseGasoline.Visibility = Visibility.Visible;

CountUseDieselText.Visibility = Visibility.Visible;

CountUseDiesel.Visibility = Visibility.Visible;

CountUseElectText.Visibility = Visibility.Visible;

CountUseElect.Visibility = Visibility.Visible;

CountUseGasText.Visibility = Visibility.Visible;

CountUseGas.Visibility = Visibility.Visible;

TotalSpentMoneyText.Visibility = Visibility.Visible;

TotalSpentMoney.Visibility = Visibility.Visible;

HideAllElements();

}

}

private void comboBoxTrip\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

if(comboBoxTrip.SelectedIndex != -1)

{

int index = comboBoxTrip.SelectedIndex;

Trip trip = db.GetTripById(allTrip[index].IdTrip);

selectTrip = trip;

StartTime.Text = Convert.ToDateTime(trip.StartTime).ToString("HH:mm dd.MM.yyyy");

FinishTime.Text = Convert.ToDateTime(trip.FinishTime).ToString("HH:mm dd.MM.yyyy");

UsedFuel.Text = trip.UsedFuel.ToString();

UsedMoney.Text = trip.UsedMoney.ToString();

People.Text = trip.People.ToString();

var route = db.GetRouteInfoById(trip.IdRoute);

StartPoint.Text = route.Item1;

FinishPoint.Text = route.Item2;

Distance.Text = route.Item3.ToString();

TimeSpan date = TimeSpan.FromMinutes(Convert.ToDouble(route.Item4));

if (date.Days >= 1)

{

if (date.Hours == 0)

{

Time.Text = $"{date.Days} д {date.Minutes} мин";

}

else

{

Time.Text = $"{date.Days} д {date.Hours} ч {date.Minutes} мин";

}

}

else

{

if (date.Hours != 0)

{

Time.Text = $"{date.Hours} ч {date.Minutes} мин";

}

else

{

Time.Text = $"{date.Minutes} мин";

}

}

if (route.Item5 != "")

{

Points.Text = route.Item5.Replace('#', '\n');

}

else

{

Points.Visibility = Visibility.Collapsed;

TextPoints.Visibility = Visibility.Collapsed;

}

Car car = db.GetCar(trip.IdCar);

if (car.Brand != "")

{

Brand.Text = car.Brand;

Model.Text = car.Model;

Generation.Text = car.Generation;

Equipment.Text = car.Equipment;

}

else

{

Brand.Text = "Неизвестно";

Model.Text = "Неизвестно";

Generation.Text = "Неизвестно";

Equipment.Text = "Неизвестно";

}

if (car.Fuel != "Электричество")

{

TextUsedFuel.Text = "Использовано топлива, л";

}

else

{

TextUsedFuel.Text = "Использовано электричества, кВт\*ч";

}

}

}

private void HideAllElements()

{

DistanceText.Visibility = Visibility.Collapsed;

Distance.Visibility = Visibility.Collapsed;

StartPointText.Visibility = Visibility.Collapsed;

StartPoint.Visibility = Visibility.Collapsed;

FinishPointText.Visibility= Visibility.Collapsed;

FinishPoint.Visibility= Visibility.Collapsed;

TimeText.Visibility = Visibility.Collapsed;

Time.Visibility = Visibility.Collapsed;

StartTimeText.Visibility = Visibility.Collapsed;

StartTime.Visibility = Visibility.Collapsed;

FinishTimeText.Visibility = Visibility.Collapsed;

FinishTime.Visibility = Visibility.Collapsed;;

TextPoints.Visibility = Visibility.Collapsed;

Points.Visibility = Visibility.Collapsed;

TextUsedFuel.Visibility = Visibility.Collapsed;

UsedFuel.Visibility = Visibility.Collapsed;

UsedMoneyText.Visibility = Visibility.Collapsed;

UsedMoney.Visibility = Visibility.Collapsed;

PeopleText.Visibility = Visibility.Collapsed;

People.Visibility = Visibility.Collapsed;

BrandText.Visibility = Visibility.Collapsed;

Brand.Visibility = Visibility.Collapsed;

ModelText.Visibility = Visibility.Collapsed;

Model.Visibility = Visibility.Collapsed;

GenerationText.Visibility = Visibility.Collapsed;

Generation.Visibility = Visibility.Collapsed;

EquipmentText.Visibility = Visibility.Collapsed;

Equipment.Visibility = Visibility.Collapsed;

}

private void ShowAllElements()

{

DistanceText.Visibility = Visibility.Visible;

Distance.Visibility = Visibility.Visible;

StartPointText.Visibility = Visibility.Visible;

StartPoint.Visibility = Visibility.Visible;

FinishPointText.Visibility = Visibility.Visible;

FinishPoint.Visibility = Visibility.Visible;

TimeText.Visibility = Visibility.Visible;

Time.Visibility = Visibility.Visible ;

StartTimeText.Visibility = Visibility.Visible;

StartTime.Visibility = Visibility.Visible;

FinishTimeText.Visibility = Visibility.Visible;

FinishTime.Visibility = Visibility.Visible;

StartTimeText.Visibility = Visibility.Visible;

StartTime.Visibility = Visibility.Visible;

TextPoints.Visibility = Visibility.Visible;

Points.Visibility = Visibility.Visible;

TextUsedFuel.Visibility = Visibility.Visible;

UsedFuel.Visibility = Visibility.Visible;

UsedMoneyText.Visibility = Visibility.Visible;

UsedMoney.Visibility = Visibility.Visible;

PeopleText.Visibility = Visibility.Visible;

People.Visibility = Visibility.Visible;

BrandText.Visibility = Visibility.Visible;

Brand.Visibility = Visibility.Visible;

ModelText.Visibility = Visibility.Visible;

Model.Visibility = Visibility.Visible;

GenerationText.Visibility = Visibility.Visible;

Generation.Visibility = Visibility.Visible;

EquipmentText.Visibility = Visibility.Visible;

Equipment.Visibility = Visibility.Visible;

}

private void CreateButton\_Click(object sender, RoutedEventArgs e)

{

StringBuilder filterQuery = new StringBuilder($"SELECT \* FROM Trips WHERE IdUser = {IdUser} ");

startData = null;

finishData = null;

DateTime date1, date2;

if (DateTime.TryParseExact(DateFrom.Text, "dd.MM.yyyy", CultureInfo.InvariantCulture, DateTimeStyles.None, out date1))

{

filterQuery.Append($"AND StartTime >= '{date1.ToString("yyyy-MM-dd HH:mm:ss")}' ");

startData = date1.ToString("dd.MM.yyyy");

}

else if(DateFrom.Text != "")

{

CustomMessageBox.Show("Уведомление", $"Введите корректную дату 1 в формате {DateTime.Now.ToString("dd.MM.yyyy")} и повторите попытку. Проверьте название и повторите попытку", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

return;

}

if (DateTime.TryParseExact(DateBefore.Text, "dd.MM.yyyy", CultureInfo.InvariantCulture, DateTimeStyles.None, out date2))

{

date2 = date2.Date.AddHours(23).AddMinutes(59).AddSeconds(59);

filterQuery.Append($"AND StartTime <= '{date2.ToString("yyyy-MM-dd HH:mm:ss")}' ");

finishData = date2.ToString("dd.MM.yyyy");

}

else if (DateBefore.Text != "")

{

CustomMessageBox.Show("Уведомление", $"Введите корректную дату 2 в формате {DateTime.Now.ToString("dd.MM.yyyy")} и повторите попытку. Проверьте название и повторите попытку", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

return;

}

if (filterQuery.ToString().EndsWith(" AND "))

{

filterQuery.Remove(filterQuery.Length - 5, 5);

}

string combinedQuery = filterQuery.ToString();

if (combinedQuery != "SELECT \* FROM Trips WHERE ")

{

filterTrip = db.GetHistory(combinedQuery);

if(filterTrip.Count > 0)

{

Count.Text = filterTrip.Count.ToString();

double counteDistance = 0;

double money = 0;

int people = 0;

foreach(Trip item in filterTrip)

{

var route = db.GetRouteInfoById(item.IdRoute);

counteDistance += route.Item3;

people += item.People;

money += item.UsedMoney;

}

TotalSpentMoney.Text = money.ToString();

CountDistance.Text = counteDistance.ToString();

AveragePeople.Text = Math.Abs(people / filterTrip.Count).ToString();

double diesel = 0;

double gasoline = 0;

double electricity = 0;

double gas = 0;

foreach (Trip item in filterTrip)

{

Car selectCar = db.GetCar(item.IdCar);

if (selectCar.Fuel == "Бензин")

{

gasoline += item.UsedFuel;

}

else if (selectCar.Fuel == "Дизельное топливо")

{

diesel += item.UsedFuel;

}

else if (selectCar.Fuel == "Электричество")

{

electricity += item.UsedFuel;

}

else if (selectCar.Fuel == "Газ")

{

gas += item.UsedFuel;

}

}

if(gasoline > 0)

{

CountUseGasoline.Text = gasoline.ToString();

CountUseGasolineText.Visibility = Visibility.Visible;

CountUseGasoline.Visibility = Visibility.Visible;

}

else

{

CountUseGasolineText.Visibility = Visibility.Collapsed;

CountUseGasoline.Visibility = Visibility.Collapsed;

}

if (diesel > 0)

{

CountUseDiesel.Text = diesel.ToString();

CountUseDieselText.Visibility = Visibility.Visible;

CountUseDiesel.Visibility = Visibility.Visible;

}

else

{

CountUseDieselText.Visibility = Visibility.Collapsed;

CountUseDiesel.Visibility = Visibility.Collapsed;

}

if (electricity > 0)

{

CountUseElect.Text = electricity.ToString();

CountUseElectText.Visibility = Visibility.Visible;

CountUseElect.Visibility = Visibility.Visible;

}

else

{

CountUseElectText.Visibility = Visibility.Collapsed;

CountUseElect.Visibility = Visibility.Collapsed;

}

if (gas > 0)

{

CountUseGas.Text = gas.ToString();

CountUseGasText.Visibility = Visibility.Visible;

CountUseGas.Visibility = Visibility.Visible;

}

else

{

CountUseGasText.Visibility = Visibility.Collapsed;

CountUseGas.Visibility = Visibility.Collapsed;

}

int popularRoute = filterTrip // расчет популярного маршрута

.GroupBy(trip => trip.IdRoute)

.OrderByDescending(group => group.Count())

.First()

.Key;

int popularCar = filterTrip // расчет популярного автомобиля

.GroupBy(trip => trip.IdCar)

.OrderByDescending(group => group.Count())

.First()

.Key;

int countRoute = 0;

int countCar = 0;

foreach(Trip item in filterTrip)

{

if(item.IdCar == popularCar)

{

countCar++;

}

if(item.IdRoute == popularRoute)

{

countRoute++;

}

}

if(countCar > 1 | (countRoute == 1 && filterTrip.Count == 1))

{

PopularCarText.Visibility = Visibility.Visible;

PopularCar.Visibility = Visibility.Visible;

Car car = db.GetCar(popularCar);

PopularCar.Text = $"{car.Brand} {car.Model} {car.Equipment}";

}

else

{

PopularCarText.Visibility = Visibility.Collapsed;

PopularCar.Visibility = Visibility.Collapsed;

}

if (countRoute > 1 | (countRoute == 1 && filterTrip.Count == 1))

{

PopularRouteText.Visibility = Visibility.Visible;

PopularRoute.Visibility = Visibility.Visible;

var select = db.GetRouteInfoById(popularRoute);

PopularRoute.Text = $"{select.Item1} - {select.Item2}";

}

else

{

PopularRouteText.Visibility = Visibility.Collapsed;

PopularRoute.Visibility = Visibility.Collapsed;

}

}

else

{

CustomMessageBox.Show("Уведомление", "Поездки не найдены.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

}

}

else

{

CustomMessageBox.Show("Уведомление", "Отображен отчет по всем поездкам пользователя.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

filterTrip = db.GetAllTripByUserId(IdUser);

}

}

private void CreateReportButton\_Click(object sender, RoutedEventArgs e)

{

if (Fill.SelectedIndex == 0) // отчет по поездке

{

if (selectTrip != null)

{

Car car = db.GetCar(selectTrip.IdCar);

var result = CustomMessageBox.Show("Уведомление", $"Желаете создать отчет по выбранной поездке?", MessageBoxButton.YesNo, CustomMessageBox.MessageBoxImage.Question);

if (result == MessageBoxResult.Yes)

{

SaveFileDialog saveFileDialog = new SaveFileDialog();

saveFileDialog.Filter = "Документ Word (\*.docx)|\*.docx";

var route = db.GetRouteInfoById(selectTrip.IdRoute);

StartPoint.Text = route.Item1;

FinishPoint.Text = route.Item2;

saveFileDialog.FileName = $"Отчет {route.Item1}-{route.Item2} на {Convert.ToDateTime(selectTrip.StartTime).ToString("dd.MM.yyyy")}.docx";

if (saveFileDialog.ShowDialog() == true)

{

string fileName = saveFileDialog.FileName;

// Создание документа Word

try

{

using (DocX document = DocX.Create(fileName))

{

document.InsertParagraph($"Отчет о поездке" + Environment.NewLine).FontSize(14d).Bold().Alignment = Alignment.center;

Paragraph paragraph = document.InsertParagraph();

paragraph.Append("Пользователь: ").FontSize(12d);

paragraph.Append($"{db.GetFirstNameById(IdUser)};" + Environment.NewLine).FontSize(12d).Bold();

Paragraph paragraphRoute = document.InsertParagraph();

paragraphRoute.Append("Маршрут: ").FontSize(12d);

paragraphRoute.Append($"{route.Item1}-{route.Item2};" + Environment.NewLine).FontSize(12d).Bold();

Paragraph paragraphCar = document.InsertParagraph();

paragraphCar.Append("Автомобиль: ").FontSize(12d);

paragraphCar.Append($"{car.Brand} {car.Model} {car.Equipment}." + Environment.NewLine).FontSize(12d).Bold();

int rowCount = route.Item5 != "" ? 8 : 7;

int currentRow = 1;

Table table = document.AddTable(rowCount, 2);

table.Rows[0].Cells[0].Paragraphs.First().Append("Параметр").FontSize(12d);

table.Rows[0].Cells[1].Paragraphs.First().Append("Значение").FontSize(12d);

table.Rows[currentRow].Cells[0].Paragraphs.First().Append("Расстояние").FontSize(12d);

table.Rows[currentRow++].Cells[1].Paragraphs.First().Append($"{route.Item3} км").FontSize(12d);

table.Rows[currentRow].Cells[0].Paragraphs.First().Append($"Время в пути").FontSize(12d);

table.Rows[currentRow++].Cells[1].Paragraphs.First().Append($"{Time.Text}").FontSize(12d);

table.Rows[currentRow].Cells[0].Paragraphs.First().Append($"Время отправления").FontSize(12d);

table.Rows[currentRow++].Cells[1].Paragraphs.First().Append($"{StartTime.Text}").FontSize(12d);

table.Rows[currentRow].Cells[0].Paragraphs.First().Append($"Время прибытия").FontSize(12d);

table.Rows[currentRow++].Cells[1].Paragraphs.First().Append($"{FinishTime.Text}").FontSize(12d);

if (route.Item5 != "")

{

table.Rows[currentRow].Cells[0].Paragraphs.First().Append($"Точки остановки").FontSize(12d);

table.Rows[currentRow++].Cells[1].Paragraphs.First().Append($"{route.Item5.Replace('#', '\n')}").FontSize(12d);

}

if (car.Fuel != "Электричество")

{

table.Rows[currentRow].Cells[0].Paragraphs.First().Append($"Общее кол-во потраченного топлива").FontSize(12d);

table.Rows[currentRow++].Cells[1].Paragraphs.First().Append($"{selectTrip.UsedFuel} литров").FontSize(12d);

}

else

{

table.Rows[currentRow].Cells[0].Paragraphs.First().Append($"Общее кол-во потраченного электричества").FontSize(12d);

table.Rows[currentRow++].Cells[1].Paragraphs.First().Append($"{selectTrip.UsedFuel} кВт \* ч").FontSize(12d);

}

table.Rows[currentRow].Cells[0].Paragraphs.First().Append($"Затраты на поездку").FontSize(12d);

table.Rows[currentRow++].Cells[1].Paragraphs.First().Append($"{selectTrip.UsedMoney} рублей").FontSize(12d);

document.InsertTable(table);

document.InsertParagraph();

document.InsertParagraph($"Дата формирования отчета {DateTime.Now.ToString("HH:mm dd.MM.yyyy")}").FontSize(12d).Alignment = Alignment.right;

document.Save();

}

CustomMessageBox.Show("Уведомление", $"Данный отчет сохранен.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

}

catch (Exception ex)

{

CustomMessageBox.Show("Ошибка", $"Произошла ошибка при сохранении отчета: {ex.Message}", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

}

}

}

}

else

{

CustomMessageBox.Show("Уведомление", $"Выберите поездку и повторите попытку.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

}

}

if (Fill.SelectedIndex == 1) // отчет за выбранный период времени

{

if (filterTrip.Count > 0) // пользователь не совершил поездку

{

MessageBoxResult result;

if(startData == null & finishData == null)

{

result = CustomMessageBox.Show("Уведомление", $"Желаете создать отчет за все поездки?", MessageBoxButton.YesNo, CustomMessageBox.MessageBoxImage.Question);

}

else

{

result = CustomMessageBox.Show("Уведомление", $"Желаете создать отчет выбранный период времени?", MessageBoxButton.YesNo, CustomMessageBox.MessageBoxImage.Question);

}

if (result == MessageBoxResult.Yes)

{

SaveFileDialog saveFileDialog = new SaveFileDialog();

saveFileDialog.Filter = "Документ Word (\*.docx)|\*.docx";

saveFileDialog.FileName = $"Отчет на {DateTime.Now.ToString("dd.MM.yyyy")}.docx";

if (saveFileDialog.ShowDialog() == true)

{

string fileName = saveFileDialog.FileName;

try

{

using (DocX document = DocX.Create(fileName))

{

if (startData != null & finishData != null)

{

document.InsertParagraph($"Отчет о поездках за {startData} - {finishData}" + Environment.NewLine).FontSize(14d).Bold().Alignment = Alignment.center;

}

else if (startData != null)

{

document.InsertParagraph($"Отчет о поездках c {startData} - {DateTime.Now.ToString("dd.MM.yyyy")}" + Environment.NewLine).FontSize(14d).Bold().Alignment = Alignment.center;

}

else if (finishData != null)

{

document.InsertParagraph($"Отчет о поездках до {finishData}" + Environment.NewLine).FontSize(14d).Bold().Alignment = Alignment.center;

}

else

{

document.InsertParagraph($"Отчет о всех поездках" + Environment.NewLine).FontSize(14d).Bold().Alignment = Alignment.center;

}

Paragraph paragraph = document.InsertParagraph();

paragraph.Append("Пользователь: ").FontSize(12d);

paragraph.Append($"{db.GetFirstNameById(IdUser)}." + Environment.NewLine).FontSize(12d).Bold();

int rowCount = 5; // базовые строки Количество поездок, Общее расстояние, Среднее количество пассажиров

if (PopularCar.Visibility == Visibility.Visible) rowCount++;

if (PopularRoute.Visibility == Visibility.Visible) rowCount++;

if (CountUseGasoline.Visibility == Visibility.Visible) rowCount++;

if (CountUseDiesel.Visibility == Visibility.Visible) rowCount++;

if (CountUseElect.Visibility == Visibility.Visible) rowCount++;

if (CountUseGas.Visibility == Visibility.Visible) rowCount++;

Table table = document.AddTable(rowCount, 2);

table.Rows[0].Cells[0].Paragraphs.First().Append("Параметр").FontSize(12d);

table.Rows[0].Cells[1].Paragraphs.First().Append("Значение").FontSize(12d);

int currentRow = 1;

table.Rows[currentRow].Cells[0].Paragraphs.First().Append("Количество поездок").FontSize(12d);

table.Rows[currentRow++].Cells[1].Paragraphs.First().Append(Count.Text).FontSize(12d);

table.Rows[currentRow].Cells[0].Paragraphs.First().Append("Общее расстояние").FontSize(12d);

table.Rows[currentRow++].Cells[1].Paragraphs.First().Append($"{CountDistance.Text} км").FontSize(12d);

table.Rows[currentRow].Cells[0].Paragraphs.First().Append("Затраты на поездки:").FontSize(12d);

table.Rows[currentRow++].Cells[1].Paragraphs.First().Append($"{TotalSpentMoney.Text} рублей").FontSize(12d);

table.Rows[currentRow].Cells[0].Paragraphs.First().Append("Среднее количество пассажиров").FontSize(12d);

table.Rows[currentRow++].Cells[1].Paragraphs.First().Append(AveragePeople.Text).FontSize(12d);

if (PopularCar.Visibility == Visibility.Visible)

{

table.Rows[currentRow].Cells[0].Paragraphs.First().Append("Часто используемый автомобиль").FontSize(12d);

table.Rows[currentRow++].Cells[1].Paragraphs.First().Append(PopularCar.Text).FontSize(12d);

}

if (PopularRoute.Visibility == Visibility.Visible)

{

table.Rows[currentRow].Cells[0].Paragraphs.First().Append("Часто используемый маршрут").FontSize(12d);

table.Rows[currentRow++].Cells[1].Paragraphs.First().Append(PopularRoute.Text).FontSize(12d);

}

if (CountUseGasoline.Visibility == Visibility.Visible)

{

table.Rows[currentRow].Cells[0].Paragraphs.First().Append("Затрачено бензина").FontSize(12d);

table.Rows[currentRow++].Cells[1].Paragraphs.First().Append($"{CountUseGasoline.Text} л").FontSize(12d);

}

if (CountUseDiesel.Visibility == Visibility.Visible)

{

table.Rows[currentRow].Cells[0].Paragraphs.First().Append("Затрачено дизельного топлива").FontSize(12d);

table.Rows[currentRow++].Cells[1].Paragraphs.First().Append($"{CountUseDiesel.Text} л").FontSize(12d);

}

if (CountUseElect.Visibility == Visibility.Visible)

{

table.Rows[currentRow].Cells[0].Paragraphs.First().Append("Затрачено электричества").FontSize(12d);

table.Rows[currentRow++].Cells[1].Paragraphs.First().Append($"{CountUseElect.Text} кВт \* ч").FontSize(12d);

}

if (CountUseGas.Visibility == Visibility.Visible)

{

table.Rows[currentRow].Cells[0].Paragraphs.First().Append("Затрачено газа").FontSize(12d);

table.Rows[currentRow++].Cells[1].Paragraphs.First().Append($"{CountUseGas.Text} л").FontSize(12d);

}

document.InsertTable(table);

foreach (Trip item in filterTrip)

{

Car car = db.GetCar(item.IdCar);

var route = db.GetRouteInfoById(item.IdRoute);

StartPoint.Text = route.Item1;

FinishPoint.Text = route.Item2;

document.InsertParagraph().InsertPageBreakAfterSelf();

Paragraph paragraphRoute = document.InsertParagraph();

paragraphRoute.Append("Маршрут: ").FontSize(12d);

paragraphRoute.Append($"{route.Item1}-{route.Item2};" + Environment.NewLine).FontSize(12d).Bold();

Paragraph paragraphCar = document.InsertParagraph();

paragraphCar.Append("Автомобиль: ").FontSize(12d);

paragraphCar.Append($"{car.Brand} {car.Model} {car.Equipment}." + Environment.NewLine).FontSize(12d).Bold();

int rowCountItem = route.Item5 != "" ? 8 : 7;

int currentRowItem = 1;

Table tableItem = document.AddTable(rowCountItem, 2);

tableItem.Rows[0].Cells[0].Paragraphs.First().Append("Параметр").FontSize(12d);

tableItem.Rows[0].Cells[1].Paragraphs.First().Append("Значение").FontSize(12d);

tableItem.Rows[currentRowItem].Cells[0].Paragraphs.First().Append("Расстояние").FontSize(12d);

tableItem.Rows[currentRowItem++].Cells[1].Paragraphs.First().Append($"{route.Item3} км").FontSize(12d);

TimeSpan date = TimeSpan.FromMinutes(Convert.ToDouble(route.Item4));

string time = "";

if (date.Days >= 1)

{

if (date.Hours == 0)

{

time = $"{date.Days} д {date.Minutes} мин";

}

else

{

time = $"{date.Days} д {date.Hours} ч {date.Minutes} мин";

}

}

else

{

if (date.Hours != 0)

{

time = $"{date.Hours} ч {date.Minutes} мин";

}

else

{

time = $"{date.Minutes} мин";

}

}

tableItem.Rows[currentRowItem].Cells[0].Paragraphs.First().Append($"Время в пути").FontSize(12d);

tableItem.Rows[currentRowItem++].Cells[1].Paragraphs.First().Append($"{time}").FontSize(12d);

tableItem.Rows[currentRowItem].Cells[0].Paragraphs.First().Append($"Время отправления").FontSize(12d);

tableItem.Rows[currentRowItem++].Cells[1].Paragraphs.First().Append($"{Convert.ToDateTime(item.StartTime).ToString("HH:mm dd.MM.yyyy")}").FontSize(12d);

tableItem.Rows[currentRowItem].Cells[0].Paragraphs.First().Append($"Время прибытия").FontSize(12d);

tableItem.Rows[currentRowItem++].Cells[1].Paragraphs.First().Append($"{Convert.ToDateTime(item.FinishTime).ToString("HH:mm dd.MM.yyyy")}").FontSize(12d);

if (route.Item5 != "")

{

tableItem.Rows[currentRowItem].Cells[0].Paragraphs.First().Append($"Точки остановки").FontSize(12d);

tableItem.Rows[currentRowItem++].Cells[1].Paragraphs.First().Append($"{route.Item5.Replace('#', '\n')}").FontSize(12d);

}

if (car.Fuel != "Электричество")

{

tableItem.Rows[currentRowItem].Cells[0].Paragraphs.First().Append($"Общее кол-во потраченного топлива").FontSize(12d);

tableItem.Rows[currentRowItem++].Cells[1].Paragraphs.First().Append($"{item.UsedFuel} литров").FontSize(12d);

}

else

{

tableItem.Rows[currentRowItem].Cells[0].Paragraphs.First().Append($"Общее кол-во потраченного электричества").FontSize(12d);

tableItem.Rows[currentRowItem++].Cells[1].Paragraphs.First().Append($"{item.UsedFuel} кВт \* ч").FontSize(12d);

}

tableItem.Rows[currentRowItem].Cells[0].Paragraphs.First().Append($"Затраты на поездку").FontSize(12d);

tableItem.Rows[currentRowItem++].Cells[1].Paragraphs.First().Append($"{item.UsedMoney} рублей").FontSize(12d);

document.InsertTable(tableItem);

}

document.InsertParagraph($"Дата формирования отчета {DateTime.Now.ToString("HH:mm dd.MM.yyyy")}").FontSize(12d).Alignment = Alignment.right;

document.Save();//сохранение word документа

}

CustomMessageBox.Show("Уведомление", $"Данный отчет сохранен.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

}

catch (Exception ex)

{

CustomMessageBox.Show("Ошибка", $"Произошла ошибка при сохранении отчета: {ex.Message}", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

}

}

}

}

else

{

CustomMessageBox.Show("Уведомление", $"Поездки не найдены, повторите попытку.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

}

}

}

}

}

<Page x:Class="OneWay.Pages.RouteInfoPage"

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:OneWay.Pages"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800"

Title="AddCarPage">

<Border CornerRadius="5" Background="{DynamicResource SecundaryBackgroundColor}">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="3\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<TextBlock Text="Информация о маршруте" Grid.Column="0" FontSize ="25" Foreground="{DynamicResource TextSecundaryColor}" Margin="10,10,10,403"/>

<ScrollViewer Grid.Column="0" Margin="0 50 0 0">

<StackPanel x:Name="HistoryPanel" Orientation="Vertical">

<Grid x:Name="FilterGrid">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="1\*"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<!-- Первая строка -->

<TextBlock Grid.Row="0" Grid.Column="0" Text="Точка отправления" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0"/>

<TextBlock Grid.Row="0" Grid.Column="1" x:Name="StartPoint" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" Margin="10 0 0 10" Visibility="Visible"/>

<!-- Вторая строка -->

<TextBlock Grid.Row="1" Grid.Column="0" x:Name="AdditionalPointText" Text="Дополнительные точки" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0"/>

<TextBlock Grid.Row="1" Grid.Column="1" x:Name="AdditionalPoint" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" Margin="10 0 0 10" Visibility="Visible" TextWrapping="WrapWithOverflow"/>

<!--<TextBlock Grid.Row="1" Grid.Column="1" x:Name="AdditionalPoint" Text="привет &#x0a;прите" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" Margin="10 0 0 10" Visibility="Visible" TextWrapping="WrapWithOverflow"/>-->

<!-- Третья строка -->

<TextBlock Grid.Row="2" Grid.Column="0" Text="Точка прибытия" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0"/>

<TextBlock Grid.Row="2" Grid.Column="1" Name="FinishPoint" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" Margin="10 0 0 10" Visibility="Visible"/>

<!-- Четвертая строка -->

<TextBlock Grid.Row="3" Grid.Column="0" Text="Расстояние" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0"/>

<TextBlock Grid.Row="3" Grid.Column="1" x:Name="Distance" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" Margin="10 0 0 10" Visibility="Visible"/>

<!-- Пятая строка -->

<TextBlock Grid.Row="4" Grid.Column="0" Text="Время в пути" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0"/>

<TextBlock Grid.Row="4" Grid.Column="1" x:Name="Time" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" Margin="10 0 0 10" Visibility="Visible"/>

</Grid>

</StackPanel>

</ScrollViewer>

<StackPanel Grid.Column="1">

<TextBlock Text="Настройки" FontSize ="25" Foreground="{DynamicResource TextSecundaryColor}" Margin="10,10,10,0"/>

<Button x:Name="DeleteButton" Content="Удалить" FontSize="15" Foreground="{DynamicResource SecundaryBackgroundColor}" Margin="10 5 10 5" BorderThickness="0" Click="DeleteButton\_Click">

<Button.Template>

<ControlTemplate TargetType="Button">

<Grid>

<Border x:Name="border" Background="{DynamicResource TextSecundaryColor}" CornerRadius="5">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

</Grid>

<ControlTemplate.Triggers>

<Trigger Property="IsMouseOver" Value="True">

<Setter TargetName="border" Property="Background" Value="#ADD8E6"/>

</Trigger>

</ControlTemplate.Triggers>

</ControlTemplate>

</Button.Template>

</Button>

<Button x:Name="FavoriteButton" Content="" FontSize="15" Foreground="{DynamicResource SecundaryBackgroundColor}" Margin="10 5 10 5" BorderThickness="0" Click="FavoriteButton\_Click">

<Button.Template>

<ControlTemplate TargetType="Button">

<Grid>

<Border x:Name="border" Background="{DynamicResource TextSecundaryColor}" CornerRadius="5">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

</Grid>

<ControlTemplate.Triggers>

<Trigger Property="IsMouseOver" Value="True">

<Setter TargetName="border" Property="Background" Value="#ADD8E6"/>

</Trigger>

</ControlTemplate.Triggers>

</ControlTemplate>

</Button.Template>

</Button>

</StackPanel>

</Grid>

</Border>

</Page>

using OneWay.Class;

using OneWay.Controls;

using System;

using System.Windows;

using System.Windows.Controls;

namespace OneWay.Pages

{

public partial class RouteInfoPage : Page

{

int IdUser;

int IdRoute;

DataBase db = new DataBase("OneWayDB.db");

public (string, string, double, string, string) route;

public RouteInfoPage(int iduser, int idRoute)

{

InitializeComponent();

this.IdUser = iduser;

this.IdRoute = idRoute;

if(db.CheckFavoriteRoute(IdUser, IdRoute) == 1)

{

this.FavoriteButton.Content = "Удалить из избранных";

}

else

{

this.FavoriteButton.Content = "Добавить в избранные";

}

route = db.GetRouteInfoById(iduser);

StartPoint.Text = route.Item1;

FinishPoint.Text = route.Item2;

Distance.Text = route.Item3.ToString() + " км";

TimeSpan date = TimeSpan.FromMinutes(Convert.ToDouble(route.Item4));

if (date.Days >= 1)

{

if (date.Hours == 0)

{

Time.Text = $"{date.Days} д {date.Minutes} мин";

}

else

{

Time.Text = $"{date.Days} д {date.Hours} ч {date.Minutes} мин";

}

}

else

{

if (date.Hours != 0)

{

Time.Text = $"{date.Hours} ч {date.Minutes} мин";

}

else

{

Time.Text = $"{date.Minutes} мин";

}

}

string formattedPoints = "";

if (!string.IsNullOrEmpty(route.Item5))

{

AdditionalPoint.Visibility = Visibility.Visible;

AdditionalPointText.Visibility = Visibility.Visible;

string[] point = route.Item5.Split('#');

formattedPoints += string.Join("\n", point);

AdditionalPoint.Text = formattedPoints;

}

else

{

AdditionalPoint.Visibility = Visibility.Collapsed;

AdditionalPointText.Visibility = Visibility.Collapsed;

}

}

private void DeleteButton\_Click(object sender, RoutedEventArgs e)

{

var result = CustomMessageBox.Show("Уведомление", $"Желаете удалить данный маршрут и удалить все совершенные поездки по данному маршруту.", MessageBoxButton.YesNo, CustomMessageBox.MessageBoxImage.Information);

if (result == MessageBoxResult.Yes)

{

if (db.DeleteRoute(IdRoute))

{

CustomMessageBox.Show("Уведомление", $"Маршрут был успешно удален.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

AllCarsPage carPage = new AllCarsPage(IdUser);

Window parentWindow = Window.GetWindow(this);

if (parentWindow != null && parentWindow is MainWindow mainWindow && mainWindow.fContainer != null)

{

mainWindow.fContainer.Content = carPage;

}

}

else

{

CustomMessageBox.Show("Уведомление", $"Маршрут не удален.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

}

}

}

private void FavoriteButton\_Click(object sender, RoutedEventArgs e)

{

if (db.CheckFavoriteRoute(IdUser,IdRoute) == 0) // проверка наличия маршрута в избранном

{

var result = CustomMessageBox.Show("Уведомление", $"Желаете добавить маршрут в избранные маршруты?", MessageBoxButton.YesNo, CustomMessageBox.MessageBoxImage.Question);

if (result == MessageBoxResult.Yes)

{

if (db.AddFavoriteRoute(IdUser, IdRoute))

{

this.FavoriteButton.Content = "Удалить из избранных";

CustomMessageBox.Show("Уведомление", $"Маршрут успешно добавлен в избранные маршруты.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

}

else

{

CustomMessageBox.Show("Уведомление", $"Маршрут не был добавлен в избранные маршруты.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

}

}

}

else

{

var result = CustomMessageBox.Show("Уведомление", $"Желаете удалить маршрут из избранных машрутов?", MessageBoxButton.YesNo, CustomMessageBox.MessageBoxImage.Question);

if (result == MessageBoxResult.Yes)

{

if (db.DeleteFavoriteRoute(IdUser, IdRoute))

{

this.FavoriteButton.Content = "Добавить в избранные";

CustomMessageBox.Show("Уведомление", $"Машрут успешно удален из избранных маршрутов.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

}

else

{

CustomMessageBox.Show("Уведомление", $"Маршрут не удален из избранных маршрутов.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Error);

}

}

}

}

}

}

<Page x:Class="OneWay.Pages.TripInfoPage"

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:OneWay.Pages"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800"

Title="TripInfo">

<Border CornerRadius="5" Background="{DynamicResource SecundaryBackgroundColor}">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="3\*"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<TextBlock Text="Информация о поездке" Grid.Column="0" FontSize ="25" Foreground="{DynamicResource TextSecundaryColor}" Margin="10,10,10,403"/>

<ScrollViewer Grid.Column="0" Margin="0 50 0 0">

<StackPanel x:Name="HistoryPanel" Orientation="Vertical">

<Grid x:Name="FilterGrid">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="1\*"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<TextBlock Grid.Row="0" Grid.Column="0" Text="Расстояние, км" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0"/>

<TextBox Grid.Row="0" Grid.Column="1" x:Name="Distance" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="10 0 0 10" IsReadOnly="True"/>

<TextBlock Grid.Row="1" Grid.Column="0" Text="Точка отправления" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0"/>

<TextBox Grid.Row="1" Grid.Column="1" x:Name="StartPoint" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="10 0 0 10" IsReadOnly="True"/>

<TextBlock Grid.Row="2" Grid.Column="0" Text="Точка прибытия" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0"/>

<TextBox Grid.Row="2" Grid.Column="1" Name="FinishPoint" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="10 0 0 10" IsReadOnly="True"/>

<TextBlock Grid.Row="3" Grid.Column="0" Text="Время в пути" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0"/>

<TextBox Grid.Row="3" Grid.Column="1" Name="Time" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="10 0 0 10" IsReadOnly="True"/>

<TextBlock Grid.Row="4" Grid.Column="0" Text="Время отправления" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0"/>

<TextBox Grid.Row="4" Grid.Column="1" x:Name="StartTime" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="10 0 0 10" IsReadOnly="True"/>

<TextBlock Grid.Row="5" Grid.Column="0" Text="Время прибытия" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0"/>

<TextBox Grid.Row="5" Grid.Column="1" x:Name="FinishTime" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="10 0 0 10" IsReadOnly="True"/>

<TextBlock Grid.Row="6" Grid.Column="0" x:Name="TextPoints" Text="Точки остановки" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0" Visibility="Visible"/>

<TextBox Grid.Row="6" Grid.Column="1" x:Name="Points" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="10 0 0 10" Visibility="Visible" IsReadOnly="True" TextWrapping="WrapWithOverflow"/>

<TextBlock Grid.Row="7" Grid.Column="0" x:Name="TextUsedFuel" Text="Использовано топлива, л" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0"/>

<TextBox Grid.Row="7" Grid.Column="1" x:Name="UsedFuel" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="10 0 0 10" IsReadOnly="True" />

<TextBlock Grid.Row="8" Grid.Column="0" Text="Затраченная сумма, руб" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0" TextWrapping="WrapWithOverflow"/>

<TextBox Grid.Row="8" Grid.Column="1" x:Name="UsedMoney" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="10 0 0 10" IsReadOnly="True"/>

<TextBlock Grid.Row="9" Grid.Column="0" Text="Количество пассажиров" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0"/>

<TextBox Grid.Row="9" Grid.Column="1" x:Name="People" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="10 0 0 10" IsReadOnly="True"/>

<TextBlock Grid.Row="10" Grid.Column="0" Text="Марка автомобиля" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0"/>

<TextBox Grid.Row="10" Grid.Column="1" x:Name="Brand" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="10 0 0 10" IsReadOnly="True"/>

<TextBlock Grid.Row="11" Grid.Column="0" Text="Модель автомобиля" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0"/>

<TextBox Grid.Row="11" Grid.Column="1" x:Name="Model" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="10 0 0 10" IsReadOnly="True"/>

<TextBlock Grid.Row="12" Grid.Column="0" Text="Поколение автомобиля" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0"/>

<TextBox Grid.Row="12" Grid.Column="1" x:Name="Generation" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="10 0 0 10" IsReadOnly="True"/>

<TextBlock Grid.Row="13" Grid.Column="0" Text="Комплектация автомобиля" FontSize ="20" Foreground="{DynamicResource TextSecundaryColor}" Margin="10 0 0 0"/>

<TextBox Grid.Row="13" Grid.Column="1" x:Name="Equipment" FontSize="20" Foreground="{DynamicResource TextSecundaryColor}" Background="Transparent" BorderThickness="0" Margin="10 0 0 10" IsReadOnly="True"/>using OneWay.Class;

using OneWay.Controls;

using System;

using System.Windows;

using System.Windows.Controls;

namespace OneWay.Pages

{

public partial class TripInfoPage : Page

{

public int IdUser;

public int IdTrip;

DataBase db = new DataBase("OneWayDB.db");

public TripInfoPage(int idUser, int idTrip)

{

InitializeComponent();

IdUser = idUser;

IdTrip = idTrip;

Trip trip = db.GetTripById(idTrip);

StartTime.Text = Convert.ToDateTime(trip.StartTime).ToString("HH:mm dd.MM.yyyy");

FinishTime.Text = Convert.ToDateTime(trip.FinishTime).ToString("HH:mm dd.MM.yyyy");

UsedFuel.Text = trip.UsedFuel.ToString();

UsedMoney.Text = trip.UsedMoney.ToString();

People.Text = trip.People.ToString();

var route = db.GetRouteInfoById(trip.IdRoute);

StartPoint.Text = route.Item1;

FinishPoint.Text = route.Item2;

Distance.Text = route.Item3.ToString();

TimeSpan date = TimeSpan.FromMinutes(Convert.ToDouble(route.Item4));

if (date.Days >= 1)

{

if (date.Hours == 0)

{

Time.Text = $"{date.Days} д {date.Minutes} мин";

}

else

{

Time.Text = $"{date.Days} д {date.Hours} ч {date.Minutes} мин";

}

}

else

{

if (date.Hours != 0)

{

Time.Text = $"{date.Hours} ч {date.Minutes} мин";

}

else

{

Time.Text = $"{date.Minutes} мин";

}

}

if (route.Item5 != "")

{

Points.Text = route.Item5.Replace('#', '\n');

}

else

{

Points.Visibility = Visibility.Collapsed;

TextPoints.Visibility = Visibility.Collapsed;

}

Car car = db.GetCar(trip.IdCar);

if (car.Brand != "")

{

Brand.Text = car.Brand;

Model.Text = car.Model;

Generation.Text = car.Generation;

Equipment.Text = car.Equipment;

}

else

{

Brand.Text = "Неизвестно";

Model.Text = "Неизвестно";

Generation.Text = "Неизвестно";

Equipment.Text = "Неизвестно";

}

if (car.Fuel != "Электричество")

{

TextUsedFuel.Text = "Использовано топлива, л";

}

else

{

TextUsedFuel.Text = "Использовано электричества, кВт\*ч";

}

}

private void DeleteButton\_Click(object sender, RoutedEventArgs e)

{

var messageBoxResult = CustomMessageBox.Show("Уведомление", $"Желаете удалить данную поездку?", MessageBoxButton.YesNo, CustomMessageBox.MessageBoxImage.Information);

if (messageBoxResult == MessageBoxResult.Yes)

{

if(db.DeleteTrip(IdTrip))

{

CustomMessageBox.Show("Уведомление", $"Поездка была удалена.", MessageBoxButton.OK, CustomMessageBox.MessageBoxImage.Information);

History historyPage = new History(IdUser);

Window parentWindow = Window.GetWindow(this);

if (parentWindow != null && parentWindow is MainWindow mainWindow && mainWindow.fContainer != null)

{

mainWindow.fContainer.Content = historyPage;

}

}

}

}

}

}

</Grid>

</StackPanel>

</ScrollViewer>

<StackPanel Grid.Column="1">

<TextBlock Text="Настройки" FontSize ="25" Foreground="{DynamicResource TextSecundaryColor}" Margin="10,10,10,0"/>

<Button x:Name="DeleteButton" Content="Удалить" FontSize="15" Foreground="{DynamicResource SecundaryBackgroundColor}" Margin="10 5 10 5" BorderThickness="0" Click="DeleteButton\_Click">

<Button.Template>

<ControlTemplate TargetType="Button">

<Grid>

<Border x:Name="border" Background="{DynamicResource TextSecundaryColor}" CornerRadius="5">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

</Grid>

<ControlTemplate.Triggers>

<Trigger Property="IsMouseOver" Value="True">

<Setter TargetName="border" Property="Background" Value="#ADD8E6"/>

</Trigger>

</ControlTemplate.Triggers>

</ControlTemplate>

</Button.Template>

</Button>

</StackPanel>

</Grid>

</Border>

</Page>

<ResourceDictionary xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:local="clr-namespace:OneWay.Styles">

<Style x:Key="PopupButtonStyle" TargetType="{x:Type Button}">

<Setter Property="Background" Value="Transparent"/>

<Setter Property="BorderBrush" Value="Transparent"/>

<Setter Property="Foreground" Value="Transparent"/>

<Setter Property="BorderThickness" Value="0"/>

<Setter Property="HorizontalContentAlignment" Value="Center"/>

<Setter Property="VerticalContentAlignment" Value="Center"/>

<Setter Property="Padding" Value="0"/>

<Setter Property="Height" Value="50"/>

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="{x:Type Button}">

<Border x:Name="border"

BorderBrush="{TemplateBinding BorderBrush}"

BorderThickness="{TemplateBinding BorderThickness}"

Background="{TemplateBinding Background}"

SnapsToDevicePixels="true">

<ContentPresenter x:Name="contentPresenter"

Focusable="False"

RecognizesAccessKey="True"

SnapsToDevicePixels="{TemplateBinding SnapsToDevicePixels}"

VerticalAlignment="{TemplateBinding VerticalContentAlignment}"

HorizontalAlignment="{TemplateBinding HorizontalContentAlignment}"

Margin="{TemplateBinding Padding}"/>

</Border>

<ControlTemplate.Triggers>

<Trigger Property="IsMouseOver" Value="True">

<Setter Property="Background" Value="Transparent"/>

<Setter Property="BorderBrush" Value="Transparent"/>

</Trigger>

<Trigger Property="IsPressed" Value="True">

<Setter Property="Background" Value="Transparent"/>

<Setter Property="BorderBrush" Value="Transparent"/>

</Trigger>

</ControlTemplate.Triggers>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

<Style x:Key="iToggleButtonStyle" TargetType="{x:Type ToggleButton}">

<Setter Property="Width" Value="30"/>

<Setter Property="Height" Value="30"/>

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="{x:Type ToggleButton}">

<Border x:Name="border" BorderThickness="{TemplateBinding BorderThickness}" Background="{TemplateBinding Background}" SnapsToDevicePixels="True">

<Viewbox Height="20" Width="20" Margin="0 6 0 0" Stretch="UniformToFill">

<Canvas Height="512" Width="512">

<Path Data="m32 0h448c17.671875 0 32 14.328125 32 32s-14.328125 32-32 32h-448c-17.671875 0-32-14.328125-32-32s14.328125-32 32-32zm0 0" Fill="{DynamicResource SecundaryIconColor}" Margin="0 0 0 0"/>

<Path Data="m32 128h448c17.671875 0 32 14.328125 32 32s-14.328125 32-32 32h-448c-17.671875 0-32-14.328125-32-32s14.328125-32 32-32zm0 0" Fill="{DynamicResource SecundaryIconColor}" Margin="0 20 0 0"/>

<Path Data="m32 256h448c17.671875 0 32 14.328125 32 32s-14.328125 32-32 32h-448c-17.671875 0-32-14.328125-32-32s14.328125-32 32-32zm0 0" Fill="{DynamicResource SecundaryIconColor}" Margin="0 40 0 0"/>

</Canvas>

</Viewbox>

</Border>

<ControlTemplate.Triggers>

<Trigger Property="IsMouseOver" Value="true">

<Setter Property="Background"

TargetName="border"

Value="Transparent"/>

<Setter Property="BorderBrush"

TargetName="border"

Value="#FFFFFF"/>

</Trigger>

<Trigger Property="IsChecked" Value="true">

<Setter Property="Background"

TargetName="border"

Value="Transparent"/>

<Setter Property="BorderBrush"

TargetName="border"

Value="#FFFFFF"/>

</Trigger>

</ControlTemplate.Triggers>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

<Style x:Key="IconButtonsStyle" TargetType="{x:Type Button}">

<Setter Property="Background" Value="Transparent"/>

<Setter Property="Cursor" Value="Hand"/>

<Setter Property="BorderThickness" Value="0"/>

<Setter Property="Height" Value="25"/>

<Setter Property="Width" Value="25"/>

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="{x:Type Button}">

<Border Name="bd" CornerRadius="5" Background="{TemplateBinding Background}" BorderThickness="{TemplateBinding BorderThickness}">

<Path Name="ico" Data="{TemplateBinding Content}" Fill="{DynamicResource PrimaryTextColor}" Height="10" Width="10" Stretch="Uniform"/>

</Border>

<ControlTemplate.Triggers>

<Trigger Property="IsMouseOver" Value="True">

<Setter Property="Background" Value="{DynamicResource SecundaryBackgroundColor}"/>

</Trigger>

<MultiTrigger>

<MultiTrigger.Conditions>

<Condition Property="IsMouseOver" Value="True"/>

<Condition Property="Tag" Value="IsCloseButton"/>

</MultiTrigger.Conditions>

<Setter Property="Background" Value="{DynamicResource btnClose}"/>

<Setter TargetName="ico" Property="Fill" Value="{DynamicResource PrimaryTexColor}"/>

</MultiTrigger>

<Trigger Property="IsPressed" Value="True">

<Setter TargetName="ico" Property="Fill" Value="{DynamicResource PrimaryTexColor}"/>

</Trigger>

</ControlTemplate.Triggers>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

<Style x:Key="ScrollBarTrackThumb"

TargetType="{x:Type Thumb}">

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="{x:Type Thumb}">

<Grid x:Name="Grid">

<Rectangle HorizontalAlignment="Stretch"

VerticalAlignment="Stretch"

Width="Auto"

Height="Auto"

Fill="Transparent" />

<Border x:Name="CornerScrollBarRectangle"

CornerRadius="5"

HorizontalAlignment="Stretch"

VerticalAlignment="Stretch"

Width="Auto"

Height="Auto"

Margin="0,1,0,1"

Background="{TemplateBinding Background}" />

</Grid>

<ControlTemplate.Triggers>

<Trigger Property="Tag"

Value="Horizontal">

<Setter TargetName="CornerScrollBarRectangle"

Property="Width"

Value="Auto" />

<Setter TargetName="CornerScrollBarRectangle"

Property="Height"

Value="6" />

</Trigger>

</ControlTemplate.Triggers>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

<Style TargetType="{x:Type ScrollBar}">

<Setter Property="Stylus.IsFlicksEnabled"

Value="false" />

<Setter Property="Foreground"

Value="#ADABAB" />

<Setter Property="Background"

Value="Transparent" />

<Setter Property="Width"

Value="7" />

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="{x:Type ScrollBar}">

<Grid x:Name="GridRoot"

Width="7"

Background="{TemplateBinding Background}">

<Grid.RowDefinitions>

<RowDefinition Height="0.00001\*" />

</Grid.RowDefinitions>

<Track x:Name="PART\_Track"

Grid.Row="0"

IsDirectionReversed="true"

Focusable="false">

<Track.Thumb>

<Thumb x:Name="Thumb"

Background="{TemplateBinding Foreground}"

Style="{DynamicResource ScrollBarTrackThumb}" />

</Track.Thumb>

<Track.IncreaseRepeatButton>

<RepeatButton x:Name="PageUp"

Command="ScrollBar.PageDownCommand"

Opacity="0"

Focusable="false" />

</Track.IncreaseRepeatButton>

<Track.DecreaseRepeatButton>

<RepeatButton x:Name="PageDown"

Command="ScrollBar.PageUpCommand"

Opacity="0"

Focusable="false" />

</Track.DecreaseRepeatButton>

</Track>

</Grid>

<ControlTemplate.Triggers>

<Trigger SourceName="Thumb"

Property="IsMouseOver"

Value="true">

<Setter Value="{DynamicResource ButtonSelectBrush}"

TargetName="Thumb"

Property="Background" />

</Trigger>

<Trigger SourceName="Thumb"

Property="IsDragging"

Value="true">

<Setter Value="{DynamicResource DarkBrush}"

TargetName="Thumb"

Property="Background" />

</Trigger>

<Trigger Property="IsEnabled"

Value="false">

<Setter TargetName="Thumb"

Property="Visibility"

Value="Collapsed" />

</Trigger>

<Trigger Property="Orientation"

Value="Horizontal">

<Setter TargetName="GridRoot"

Property="LayoutTransform">

<Setter.Value>

<RotateTransform Angle="-90" />

</Setter.Value>

</Setter>

<Setter TargetName="PART\_Track"

Property="LayoutTransform">

<Setter.Value>

<RotateTransform Angle="-90" />

</Setter.Value>

</Setter>

<Setter Property="Width"

Value="Auto" />

<Setter Property="Height"

Value="8" />

<Setter TargetName="Thumb"

Property="Tag"

Value="Horizontal" />

<Setter TargetName="PageDown"

Property="Command"

Value="ScrollBar.PageLeftCommand" />

<Setter TargetName="PageUp"

Property="Command"

Value="ScrollBar.PageRightCommand" />

</Trigger>

</ControlTemplate.Triggers>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

<Style TargetType="TextBox" BasedOn="{StaticResource {x:Type TextBox}}">

<Setter Property="CaretBrush" Value="{DynamicResource TextSecundaryColor}"/>

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="{x:Type TextBox}">

<Grid>

<Border Background="{TemplateBinding Background}"

BorderBrush="{TemplateBinding BorderBrush}"

BorderThickness="{TemplateBinding BorderThickness}"/>

<TextBlock x:Name="Placeholder"

Text="{TemplateBinding Tag}"

Foreground="Gray"

Visibility="Collapsed"/>

<ScrollViewer x:Name="PART\_ContentHost"

Margin="{TemplateBinding Padding}"/>

</Grid>

<ControlTemplate.Triggers>

<MultiTrigger>

<MultiTrigger.Conditions>

<Condition Property="Text" Value=""/>

</MultiTrigger.Conditions>

<Setter TargetName="Placeholder" Property="Visibility" Value="Visible"/>

</MultiTrigger>

</ControlTemplate.Triggers>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

<Style TargetType="PasswordBox" BasedOn="{StaticResource {x:Type PasswordBox}}">

<Setter Property="CaretBrush" Value="{DynamicResource TextSecundaryColor}"/>

<Setter Property="Foreground" Value="{DynamicResource TextPrimaryColor}"/>

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="{x:Type PasswordBox}">

<Grid>

<Border Background="{TemplateBinding Background}"

BorderBrush="{TemplateBinding BorderBrush}"

BorderThickness="{TemplateBinding BorderThickness}"/>

<TextBlock x:Name="Placeholder"

Text="{TemplateBinding Tag}"

Foreground="Gray"

Visibility="Collapsed"/>

<ScrollViewer x:Name="PART\_ContentHost"

Margin="{TemplateBinding Padding}"/>

</Grid>

<ControlTemplate.Triggers>

<DataTrigger Binding="{Binding Password.Length, RelativeSource={RelativeSource TemplatedParent}}" Value="0">

<Setter TargetName="Placeholder" Property="Visibility" Value="Visible"/>

</DataTrigger>

</ControlTemplate.Triggers>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

<ControlTemplate x:Key="ComboBoxToggleButton" TargetType="{x:Type ToggleButton}">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition />

<ColumnDefinition Width="20" />

</Grid.ColumnDefinitions>

<!-- На весь текст-->

<Border

x:Name="Border"

Grid.ColumnSpan="2"

CornerRadius="0"

/>

<!-- На текст-->

<Border

Grid.Column="0"

CornerRadius="0"

Margin="1"

Background="Transparent"

/>

<Path x:Name="Arrow"

Grid.Column="1"

Fill="{DynamicResource TextSecundaryColor}"

HorizontalAlignment="Center"

VerticalAlignment="Center"

Data="M17,9.17a1,1,0,0,0-1.41,0L12,12.71,8.46,9.17a1,1,0,0,0-1.41,0,1,1,0,0,0,0,1.42l4.24,4.24a1,1,0,0,0,1.42,0L17,10.59A1,1,0,0,0,17,9.17Z"

/>

</Grid>

</ControlTemplate>

<ControlTemplate x:Key="ComboBoxTextBox" TargetType="{x:Type TextBox}">

<Border x:Name="PART\_ContentHost" Focusable="False" Background="{TemplateBinding Background}" />

</ControlTemplate>

<Style x:Key="{x:Type ComboBox}" TargetType="{x:Type ComboBox}"> <!-- Текст в комбобокс-->

<Setter Property="SnapsToDevicePixels" Value="true"/>

<Setter Property="OverridesDefaultStyle" Value="true"/>

<Setter Property="ScrollViewer.HorizontalScrollBarVisibility" Value="Auto"/>

<Setter Property="ScrollViewer.VerticalScrollBarVisibility" Value="Auto"/>

<Setter Property="ScrollViewer.CanContentScroll" Value="true"/>

<Setter Property="Foreground" Value="{DynamicResource TextSecundaryColor}"/>

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="{x:Type ComboBox}">

<Grid>

<ToggleButton

Name="ToggleButton"

Template="{StaticResource ComboBoxToggleButton}"

Grid.Column="2"

Focusable="false"

IsChecked="{Binding Path=IsDropDownOpen,Mode=TwoWay,RelativeSource={RelativeSource TemplatedParent}}"

ClickMode="Press">

</ToggleButton>

<ContentPresenter Name="ContentSite" IsHitTestVisible="False" Content="{TemplateBinding SelectionBoxItem}"

ContentTemplate="{TemplateBinding SelectionBoxItemTemplate}"

ContentTemplateSelector="{TemplateBinding ItemTemplateSelector}"

Margin="3,3,23,3"

VerticalAlignment="Center"

HorizontalAlignment="Left" />

<TextBox x:Name="PART\_EditableTextBox"

Style="{x:Null}"

Template="{StaticResource ComboBoxTextBox}"

HorizontalAlignment="Left"

VerticalAlignment="Center"

Margin="3,3,23,3"

Focusable="True"

Background="Transparent"

Foreground="Gray"

Visibility="Hidden"

IsReadOnly="{TemplateBinding IsReadOnly}"/>

<Popup

Name="Popup"

Placement="Bottom"

IsOpen="{TemplateBinding IsDropDownOpen}"

AllowsTransparency="True"

Focusable="False"

PopupAnimation="Slide">

<Grid Name="DropDown"

SnapsToDevicePixels="True"

MinWidth="{TemplateBinding ActualWidth}"

MaxHeight="{TemplateBinding MaxDropDownHeight}">

<Border

x:Name="DropDownBorder"

Background="{DynamicResource PrimaryBackgroundColor}"/>

<ScrollViewer Margin="4,6,4,6" SnapsToDevicePixels="True">

<StackPanel IsItemsHost="True" KeyboardNavigation.DirectionalNavigation="Contained" />

</ScrollViewer>

</Grid>

</Popup>

</Grid>

<ControlTemplate.Triggers>

<Trigger Property="HasItems" Value="false">

<Setter TargetName="DropDownBorder" Property="MinHeight" Value="95"/>

</Trigger>

<Trigger Property="IsEnabled" Value="false">

<Setter Property="Foreground" Value="#888888"/>

</Trigger>

<Trigger Property="IsGrouping" Value="true">

<Setter Property="ScrollViewer.CanContentScroll" Value="false"/>

</Trigger>

<Trigger SourceName="Popup" Property="Popup.AllowsTransparency" Value="true">

<Setter TargetName="DropDownBorder" Property="CornerRadius" Value="0"/>

<Setter TargetName="DropDownBorder" Property="Margin" Value="0,2,0,0"/>

</Trigger>

<Trigger Property="IsEditable" Value="true">

<Setter Property="IsTabStop" Value="false"/>

<Setter TargetName="PART\_EditableTextBox" Property="Visibility" Value="Visible"/>

<Setter TargetName="ContentSite" Property="Visibility" Value="Hidden"/>

</Trigger>

</ControlTemplate.Triggers>

</ControlTemplate>

</Setter.Value>

</Setter>

<Style.Triggers>

</Style.Triggers>

</Style>

<Style x:Key="{x:Type ComboBoxItem}" TargetType="{x:Type ComboBoxItem}">

<Setter Property="SnapsToDevicePixels" Value="true"/>

<Setter Property="Foreground" Value="{DynamicResource TextSecundaryColor}"/>

<Setter Property="OverridesDefaultStyle" Value="true"/>

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="{x:Type ComboBoxItem}">

<Border Name="Border"

Padding="2"

SnapsToDevicePixels="true">

<ContentPresenter />

</Border>

<ControlTemplate.Triggers>

<Trigger Property="IsHighlighted" Value="true">

<Setter TargetName="Border" Property="Background" Value="#FF4F4F4F"/>

</Trigger>

<Trigger Property="IsEnabled" Value="false">

<Setter Property="Foreground" Value="#888888"/>

</Trigger>

</ControlTemplate.Triggers>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

</ResourceDictionary>

<ResourceDictionary xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">

<SolidColorBrush x:Key="PrimaryBackgroundColor" Color="#131725"/>

<SolidColorBrush x:Key="SecundaryBackgroundColor" Color="#1d2335"/>

<SolidColorBrush x:Key="PrimaryBlueColor" Color="#3772FF"/>

<SolidColorBrush x:Key="PrimaryTextColor" Color="#FFFFFF"/>

<SolidColorBrush x:Key="TextSecundaryColor" Color="#B0B7C3"/>

<SolidColorBrush x:Key="TextTertiaryColor" Color="#8A94A6"/>

<SolidColorBrush x:Key="TextFourthColor" Color="#4B5D78"/>

<SolidColorBrush x:Key="PrimaryIconColor" Color="#B0B7C3"/>

<SolidColorBrush x:Key="SecundaryIconColor" Color="#8A94A6"/>

<SolidColorBrush x:Key="TertiarIconyColor" Color="#4B5D78"/>

<SolidColorBrush x:Key="FourthIconColor" Color="#323B4B"/>

<SolidColorBrush x:Key="btnClose" Color="#F72626"/>

</ResourceDictionary>

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

<configuration>

<startup>

<supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.8" />

</startup>

</configuration>

using System.Windows;

namespace OneWay

{

public partial class App : Application

{

}

}

<Application x:Class="OneWay.App"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:local="clr-namespace:OneWay"

StartupUri="MainWindow.xaml">

<Application.Resources>

<ResourceDictionary>

<ResourceDictionary.MergedDictionaries>

<ResourceDictionary Source="Themes/DarkTheme.xaml"/>

<ResourceDictionary Source="Assets/Icons.xaml"/>

<ResourceDictionary Source="Styles/MainStyle.xaml"/>

</ResourceDictionary.MergedDictionaries>

</ResourceDictionary>

</Application.Resources>

</Application>

<Window x:Class="OneWay.MainWindow"

xmlns:ctls="clr-namespace:OneWay.Controls"

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:OneWay"

mc:Ignorable="d" Name="home"

Title="MainWindow" Height="650" Width="1000"

WindowStartupLocation="CenterScreen" WindowStyle="None"

Background="Transparent" AllowsTransparency="True"

ResizeMode="NoResize" FontFamily="Poppins">

<Border Background="{DynamicResource PrimaryBackgroundColor}"

CornerRadius="8">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="Auto"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<!-- Tg\_Btn-->

<Border x:Name="ItemPanel" Grid.Column="0" Visibility="Visible"

Background="{DynamicResource SecundaryBackgroundColor}"

CornerRadius="8 0 0 8">

<Grid x:Name="GridNav" Width="60">

<Grid.RowDefinitions>

<RowDefinition Height="60"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<Grid Grid.Row="0">

<Border

Margin="14 -5 0 0"

Width="32" Height="24"

HorizontalAlignment="Left">

<Border.Background>

<ImageBrush ImageSource="Resources/logo.png" Stretch="Uniform"/>

</Border.Background>

</Border>

<Label Content="OneWay"

Foreground="{DynamicResource PrimaryTextColor}"

FontWeight="Bold"

FontSize="16"

VerticalAlignment="Center"

HorizontalAlignment="Left"

Margin="60 0 0 0"/>

</Grid>

<StackPanel Grid.Row="1">

<Button x:Name="btnHome" Style="{DynamicResource PopupButtonStyle}" MouseEnter="btnHome\_MouseEnter" MouseLeave="btnHome\_MouseLeave" Click="btnHome\_Click">

<ctls:MenuItem GroupName="MenuItem" IndicatorBrush="{DynamicResource PrimaryBlueColor}" Icon="{DynamicResource home}" IconWidth="18" Text="Главная" VerticalAlignment="Center"/>

</Button>

<Button x:Name="btnDashboard" Style="{DynamicResource PopupButtonStyle}" MouseEnter="btnDashboard\_MouseEnter" MouseLeave="btnDashboard\_MouseLeave" Click="btnDashboard\_Click">

<ctls:MenuItem GroupName="MenuItem" IndicatorBrush="{DynamicResource PrimaryBlueColor}" Icon="{DynamicResource dashboard}" IconWidth="18" Text="Статистика" VerticalAlignment="Center"/>

</Button>

<Button x:Name="btnTrip" Style="{DynamicResource PopupButtonStyle}" MouseEnter="btnTrip\_MouseEnter" MouseLeave="btnTrip\_MouseLeave" Click="btnTrip\_Click" >

<ctls:MenuItem GroupName="MenuItem" IndicatorBrush="{DynamicResource PrimaryBlueColor}" Icon="{DynamicResource trip}" IconWidth="18" Text="Поездка" VerticalAlignment="Center"/>

</Button>

<Button x:Name="btnRoute" Style="{DynamicResource PopupButtonStyle}" MouseEnter="btnRoute\_MouseEnter" MouseLeave="btnTrip\_MouseLeave" Click="btnRoute\_Click" >

<ctls:MenuItem GroupName="MenuItem" IndicatorBrush="{DynamicResource PrimaryBlueColor}" Icon="{DynamicResource route}" IconWidth="18" Text="Маршрут" VerticalAlignment="Center"/>

</Button>

<Button x:Name="btnHistory" Style="{DynamicResource PopupButtonStyle}" MouseEnter="btnHistory\_MouseEnter" MouseLeave="btnHistory\_MouseLeave" Click="btnHistory\_Click">

<ctls:MenuItem GroupName="MenuItem" IndicatorBrush="{DynamicResource PrimaryBlueColor}" Icon="{DynamicResource history}" IconWidth="20" Text="История" VerticalAlignment="Center"/>

</Button>

<Button x:Name="btnFuel" Style="{DynamicResource PopupButtonStyle}" MouseEnter="btnFuel\_MouseEnter" MouseLeave="btnFuel\_MouseLeave" Click="btnFuel\_Click">

<ctls:MenuItem GroupName="MenuItem" IndicatorBrush="{DynamicResource PrimaryBlueColor}" Icon="{DynamicResource fuel}" IconWidth="16" Text="Топливо" VerticalAlignment="Center"/>

</Button>

<Button x:Name="btnCar" Style="{DynamicResource PopupButtonStyle}" MouseEnter="btnCar\_MouseEnter" MouseLeave="btnCar\_MouseLeave" Click="btnCars\_Click">

<ctls:MenuItem GroupName="MenuItem" IndicatorBrush="{DynamicResource PrimaryBlueColor}" Icon="{DynamicResource car}" IconWidth="16" Text="Транспорт" VerticalAlignment="Center"/>

</Button>

<Button x:Name="btnReport" Style="{DynamicResource PopupButtonStyle}" MouseEnter="btnReport\_MouseEnter" MouseLeave="btnReport\_MouseLeave" Click="btnReport\_Click">

<ctls:MenuItem GroupName="MenuItem" IndicatorBrush="{DynamicResource PrimaryBlueColor}" Icon="{DynamicResource report}" IconWidth="16" Text="Отчет" VerticalAlignment="Center"/>

</Button>

<Button x:Name="btnHelp" Style="{DynamicResource PopupButtonStyle}" MouseEnter="btnHelp\_MouseEnter" MouseLeave="btnHelp\_MouseLeave" Click="btnHelp\_Click">

<ctls:MenuItem GroupName="MenuItem" IndicatorBrush="{DynamicResource PrimaryBlueColor}" Icon="{DynamicResource help}" IconWidth="18" Text="Справка" VerticalAlignment="Center"/>

</Button>

</StackPanel>

<Popup x:Name="Popup" AllowsTransparency="True" HorizontalOffset="0" VerticalOffset="10">

<ctls:UserPopup x:Name="Header" Foreground="{DynamicResource PrimaryTextColor}"/>

</Popup>

</Grid>

</Border>

<Grid Grid.Column="1">

<Grid.RowDefinitions>

<RowDefinition Height="60"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<ToggleButton x:Name="Tg\_Btn" Grid.Column="1" VerticalAlignment="Center" Style="{DynamicResource iToggleButtonStyle}" HorizontalAlignment="Left" BorderThickness="0" Background="Transparent" Margin="10,15,0,15" Height="30" Width="30" Visibility="Visible">

<ToggleButton.Triggers>

<EventTrigger RoutedEvent="ToggleButton.Unchecked">

<BeginStoryboard>

<Storyboard x:Name="HideStackPanel">

<DoubleAnimation Storyboard.TargetName="GridNav" Storyboard.TargetProperty="Width" BeginTime="0:0:0" From="210" To="60" Duration="0:0:0.5">

</DoubleAnimation>

</Storyboard>

</BeginStoryboard>

</EventTrigger>

<EventTrigger RoutedEvent="ToggleButton.Checked">

<BeginStoryboard>

<Storyboard x:Name="ShowStackPanel">

<DoubleAnimation Storyboard.TargetName="GridNav" Storyboard.TargetProperty="Width" BeginTime="0:0:0" From="60" To="210" Duration="0:0:0.5">

</DoubleAnimation>

</Storyboard>

</BeginStoryboard>

</EventTrigger>

</ToggleButton.Triggers>

</ToggleButton>

<StackPanel Grid.Row="0" HorizontalAlignment="Right" VerticalAlignment="Top" Margin="0,5,10,0" Height="30" Orientation="Horizontal" FlowDirection="RightToLeft" Grid.RowSpan="2">

<Button x:Name="btnClose" Style="{StaticResource IconButtonsStyle}" Content="{DynamicResource close}" Click="btnClose\_Click" Tag="IsCloseButton" Cursor="Hand"/>

<Button x:Name="btnRestore" Margin="5,0" Click="btnRestore\_Click">

<Button.Style>

<Style TargetType="{x:Type Button}" BasedOn="{StaticResource IconButtonsStyle}">

<Setter Property="Content" Value="{DynamicResource maximize}"/>

<Style.Triggers>

<DataTrigger Binding="{Binding ElementName=home, Path=WindowState}" Value="Maximized">

<Setter Property="Content" Value="{DynamicResource restore}"/>

</DataTrigger>

</Style.Triggers>

</Style>

</Button.Style>

</Button>

<Button x:Name="btnMinimize" Style="{DynamicResource IconButtonsStyle}" Content="{DynamicResource minimize}" Click="btnMinimize\_Click"/>

</StackPanel>

<Frame Grid.Row="1"

Name="fContainer"

Margin="20"

NavigationUIVisibility="Hidden">

</Frame>

</Grid>

</Grid>

</Border>

</Window>

using HtmlAgilityPack;

using OneWay.Class;

using OneWay.Pages;

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.Net.NetworkInformation;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls.Primitives;

using System.Windows.Input;

namespace OneWay

{

public partial class MainWindow : Window

{

public int IdUser;

DataBase db = new DataBase("OneWayDB.db");

public MainWindow()

{

InitializeComponent();

Tg\_Btn.Visibility = Visibility.Collapsed;

ItemPanel.Visibility = Visibility.Collapsed;

AuthenticationPage authenticationPage = new AuthenticationPage();

authenticationPage.AuthenticationCompleted += (sender, IdUser) =>

{

this.IdUser = IdUser;

if (IdUser != 0)

{

Tg\_Btn.Visibility = Visibility.Visible;

ItemPanel.Visibility = Visibility.Visible;

Home HomePage = new Home(IdUser);

fContainer.Navigate(HomePage);

}

};

fContainer.Navigate(authenticationPage);

UpdateFuelPrice();

}

public async void UpdateFuelPrice()

{

DateTime currentDate = DateTime.Now.Date;

DateTime lastUpdateDate = DateTime.ParseExact(db.GetLastDate(), "dd.MM.yyyy", null);

if (lastUpdateDate != currentDate)

{

bool interner = await CheckInternetConnectionAsync();

if (interner)

{

List<string> prices = GetFuelPrices();

int[] idFuel = new int[] { 9, 4, 5, 6, 1, 2, 3 };

string date = DateTime.Now.ToString("dd.MM.yyyy");

for (int i = 0; i < prices.Count - 1; i++)

{

db.UpdatePrice(idFuel[i], -1, prices[i], date);

}

}

}

}

public async Task<bool> CheckInternetConnectionAsync()

{

try

{

Ping ping = new Ping();

PingReply reply = await ping.SendPingAsync("www.google.com");

return reply.Status == IPStatus.Success;

}

catch (PingException)

{

return false;

}

}

public List<string> GetFuelPrices()

{

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

string web = "https://azs.belorusneft.by/sitebeloil/ru/center/azs/center/fuelandService/price/";

try

{

HtmlWeb webGet = new HtmlWeb();

HtmlDocument document = webGet.Load(web);

HtmlNode fuelNode = document.DocumentNode.SelectSingleNode("//div[@class='b-left\_\_informer']");

if (fuelNode != null)

{

HtmlNodeCollection tdNodes = fuelNode.SelectNodes(".//td");

if (tdNodes != null)

{

foreach (HtmlNode tdNode in tdNodes)

{

prices.Add(tdNode.InnerText);

}

}

}

prices.Add(DateTime.Now.ToString("f"));

}

catch (Exception ex)

{

MessageBox.Show("Произошла ошибка: " + ex.Message);

}

return prices;

}

private void BG\_PreviewMouseLeftButtonDown(object sender, MouseButtonEventArgs e)

{

Tg\_Btn.IsChecked = false;

}

private void btnHome\_MouseEnter(object sender, MouseEventArgs e)

{

if (Tg\_Btn.IsChecked == false)

{

Popup.PlacementTarget = btnHome;

Popup.Placement = PlacementMode.Right;

Popup.IsOpen = true;

Header.PopupText.Text = "Главная";

}

}

private void btnHome\_MouseLeave(object sender, MouseEventArgs e)

{

Popup.Visibility = Visibility.Collapsed;

Popup.IsOpen = false;

}

private void btnDashboard\_MouseEnter(object sender, MouseEventArgs e)

{

if (Tg\_Btn.IsChecked == false)

{

Popup.PlacementTarget = btnDashboard;

Popup.Placement = PlacementMode.Right;

Popup.IsOpen = true;

Header.PopupText.Text = "Статистика";

}

}

private void btnDashboard\_MouseLeave(object sender, MouseEventArgs e)

{

Popup.Visibility = Visibility.Collapsed;

Popup.IsOpen = false;

}

private void btnTrip\_MouseEnter(object sender, MouseEventArgs e)

{

if (Tg\_Btn.IsChecked == false)

{

Popup.PlacementTarget = btnTrip;

Popup.Placement = PlacementMode.Right;

Popup.IsOpen = true;

Header.PopupText.Text = "Поездка";

}

}

private void btnTrip\_MouseLeave(object sender, MouseEventArgs e)

{

Popup.Visibility = Visibility.Collapsed;

Popup.IsOpen = false;

}

private void btnRoute\_MouseEnter(object sender, MouseEventArgs e)

{

if (Tg\_Btn.IsChecked == false)

{

Popup.PlacementTarget = btnRoute;

Popup.Placement = PlacementMode.Right;

Popup.IsOpen = true;

Header.PopupText.Text = "Маршрут";

}

}

private void btnRoute\_MouseLeave(object sender, MouseEventArgs e)

{

Popup.Visibility = Visibility.Collapsed;

Popup.IsOpen = false;

}

private void btnHistory\_MouseEnter(object sender, MouseEventArgs e)

{

if (Tg\_Btn.IsChecked == false)

{

Popup.PlacementTarget = btnHistory;

Popup.Placement = PlacementMode.Right;

Popup.IsOpen = true;

Header.PopupText.Text = "История";

}

}

private void btnHistory\_MouseLeave(object sender, MouseEventArgs e)

{

Popup.Visibility = Visibility.Collapsed;

Popup.IsOpen = false;

}

private void btnFuel\_MouseEnter(object sender, MouseEventArgs e)

{

if (Tg\_Btn.IsChecked == false)

{

Popup.PlacementTarget = btnFuel;

Popup.Placement = PlacementMode.Right;

Popup.IsOpen = true;

Header.PopupText.Text = "Топливо";

}

}

private void btnFuel\_MouseLeave(object sender, MouseEventArgs e)

{

Popup.Visibility = Visibility.Collapsed;

Popup.IsOpen = false;

}

private void btnCar\_MouseEnter(object sender, MouseEventArgs e)

{

if (Tg\_Btn.IsChecked == false)

{

Popup.PlacementTarget = btnCar;

Popup.Placement = PlacementMode.Right;

Popup.IsOpen = true;

Header.PopupText.Text = "Транспорт";

}

}

private void btnCar\_MouseLeave(object sender, MouseEventArgs e)

{

Popup.Visibility = Visibility.Collapsed;

Popup.IsOpen = false;

}

private void btnReport\_MouseEnter(object sender, MouseEventArgs e)

{

if (Tg\_Btn.IsChecked == false)

{

Popup.PlacementTarget = btnReport;

Popup.Placement = PlacementMode.Right;

Popup.IsOpen = true;

Header.PopupText.Text = "Отчет";

}

}

private void btnReport\_MouseLeave(object sender, MouseEventArgs e)

{

Popup.Visibility = Visibility.Collapsed;

Popup.IsOpen = false;

}

private void btnHelp\_MouseEnter(object sender, MouseEventArgs e)

{

if (Tg\_Btn.IsChecked == false)

{

Popup.PlacementTarget = btnHelp;

Popup.Placement = PlacementMode.Right;

Popup.IsOpen = true;

Header.PopupText.Text = "Справка";

}

}

private void btnHelp\_MouseLeave(object sender, MouseEventArgs e)

{

Popup.Visibility = Visibility.Collapsed;

Popup.IsOpen = false;

}

private void btnClose\_Click(object sender, RoutedEventArgs e)

{

Close();

}

private void btnRestore\_Click(object sender, RoutedEventArgs e)

{

if (WindowState == WindowState.Normal)

WindowState = WindowState.Maximized;

else

WindowState = WindowState.Normal;

}

private void btnMinimize\_Click(object sender, RoutedEventArgs e)

{

WindowState = WindowState.Minimized;

}

private void btnHome\_Click(object sender, RoutedEventArgs e)

{

Home HomePage = new Home(IdUser);

fContainer.Navigate(HomePage);

}

private void btnDashboard\_Click(object sender, RoutedEventArgs e)

{

Dashboard DashboardPage = new Dashboard(IdUser);

fContainer.Navigate(DashboardPage);

}

private void btnTrip\_Click(object sender, RoutedEventArgs e)

{

AddTripPage TripPage = new AddTripPage(IdUser);

fContainer.Navigate(TripPage);

}

private void btnRoute\_Click(object sender, RoutedEventArgs e)

{

AllRoutes RoutePage = new AllRoutes(IdUser);

fContainer.Navigate(RoutePage);

}

private void btnHistory\_Click(object sender, RoutedEventArgs e)

{

History HistoryPage = new History(IdUser);

fContainer.Navigate(HistoryPage);

}

private void btnCars\_Click(object sender, RoutedEventArgs e)

{

AllCarsPage CarsPage = new AllCarsPage(IdUser);

fContainer.Navigate(CarsPage);

}

private void btnFuel\_Click(object sender, RoutedEventArgs e)

{

AllFuel FuelPage = new AllFuel(IdUser);

fContainer.Navigate(FuelPage);

}

private void btnReport\_Click(object sender, RoutedEventArgs e)

{

Report ReportPage = new Report(IdUser);

fContainer.Navigate(ReportPage);

}

private void btnHelp\_Click(object sender, RoutedEventArgs e)

{

Process.Start("Help.chm");

}

}

}

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

<packages>

<package id="DocX" version="3.0.0" targetFramework="net48" />

<package id="HtmlAgilityPack" version="1.11.60" targetFramework="net48" />

<package id="LiveCharts" version="0.9.7" targetFramework="net48" />

<package id="LiveCharts.Wpf" version="0.9.7" targetFramework="net48" />

<package id="MahApps.Metro.IconPacks" version="4.11.0" targetFramework="net48" />

<package id="Newtonsoft.Json" version="13.0.3" targetFramework="net48" />

<package id="OxyPlot.Core" version="2.1.2" targetFramework="net48" />

<package id="OxyPlot.Wpf" version="2.1.2" targetFramework="net48" />

<package id="OxyPlot.Wpf.Shared" version="2.1.2" targetFramework="net48" />

<package id="SQLitePCLRaw.core" version="2.1.6" targetFramework="net48" />

<package id="Stub.System.Data.SQLite.Core.NetFramework" version="1.0.118.0" targetFramework="net48" />

<package id="System.Buffers" version="4.4.0" targetFramework="net48" />

<package id="System.Data.SQLite.Core" version="1.0.118.0" targetFramework="net48" />

<package id="System.Memory" version="4.5.3" targetFramework="net48" />

<package id="System.Numerics.Vectors" version="4.4.0" targetFramework="net48" />

<package id="System.Runtime.CompilerServices.Unsafe" version="4.5.2" targetFramework="net48" />

</packages>