|  |
| --- |
| **МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ** |
| **федеральное государственное бюджетное образовательное учреждение высшего образования «Российский экономический университет имени Г.В. Плеханова»**  **Московский приборостроительный техникум** |

ОТЧЕТ

по учебной практике

УП.01.01 Прикладное программирование

Профессионального модуля ПМ.01 Разработка модулей программного обеспечения для компьютерных систем

Специальность 09.02.07 Информационные системы и программирование

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Соколов Н.Н

*подпись фамилия, имя, отчество*

Группа П50-6-20

Руководитель по практической подготовке от техникума

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Буканов И.Д.

*подпись фамилия, имя, отчество*

«\_\_\_\_» \_\_\_\_\_\_\_\_\_ 2022 года

СОДЕРЖАНИЕ

[1. ЦЕЛИ ПРАКТИКИ 3](#_Toc97750048)

[2. ПРОЕКТИРОВАНИЕ 4](#_Toc97750049)

[2.1. Описание предметной области 4](#_Toc97750050)

[2.2. Инфологическая модель данных 4](#_Toc97750051)

[2.3. Даталогическая модель данных 4](#_Toc97750052)

[2.4. Словарь данных 4](#_Toc97750053)

[3. РЕАЛИЗАЦИЯ 5](#_Toc97750054)

[3.1. Описание модулей приложения 5](#_Toc97750055)

[3.2. Программный код модулей 5](#_Toc97750056)

[4. РЕЗУЛЬТАТ РАБОТЫ 6](#_Toc97750057)

[4.1. Описание функций приложения базы данных 6](#_Toc97750058)

[4.2. Описание дополнительных функций 6](#_Toc97750059)

[5. ВЫВОД 7](#_Toc97750060)

[6. СПИСОК ИСПОЛЬЗУЕМЫХ МАТЕРИАЛОВ 8](#_Toc97750061)

1. ЦЕЛИ ПРАКТИКИ

Целью учебной практики УП 01.01 «Прикладное программирование», в рамках профессионального модуля ПМ 01 «Разработка модулей программного обеспечения для компьютерных систем», является закрепление практических навыков и умений:

* Формировать алгоритмы разработки программных модулей в соответствии с требованиями заказчика;
* Разрабатывать программные модули в соответствии с представленными требованиями заказчика;
* Выполнять отладку модулей с использованием специализированных программных средств;
* Выполнять тестирование программных модулей;
* Осуществлять рефакторинг и оптимизацию программного кода

1. ПРОЕКТИРОВАНИЕ
   1. Описание предметной области
      1. Название предметной области.

«Службы такси»

* + 1. Название подсистемы.

Учёт и классификации таксопарка, оформление заказа.

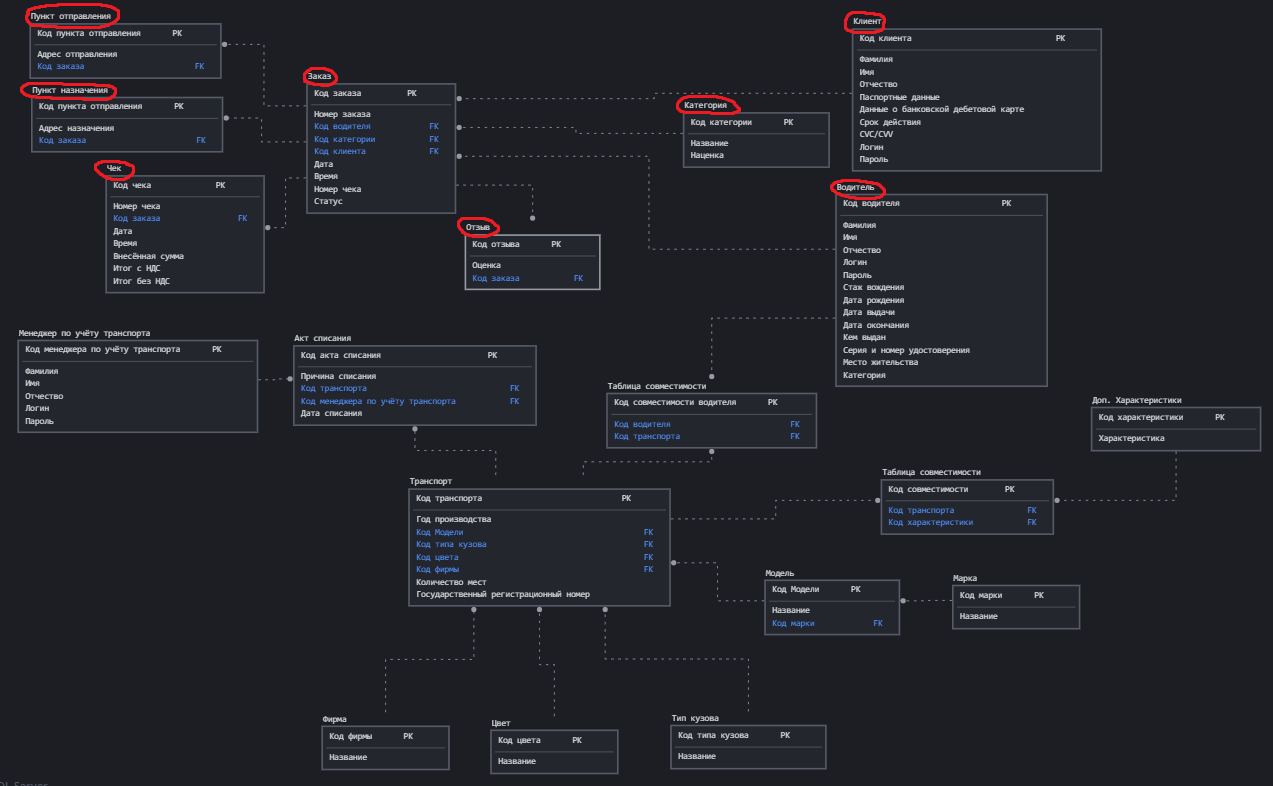
* + 1. Описание предметной области

Фирма занимается перевозкой одно или нескольких пассажиров из точки А в точку Б. При этом перед началом поездки, клиент вносит персональные данные: паспортные данные, данные о банковской дебетовой карты, для осуществления списания денежных средств. Клиент может выбрать удобную категорию автомобиля, которые заранее определены в службе доставки в качестве таксопарка. Клиент может оценить поездку по пятибалльной шкале и оставить отзыв о поезде. Каждая категория автомобилей имеет свою собственную цену, при этом в зависимости от количества пассажиров (количество не должно превышать, количество посадочных мест в транспорте) наценка на категорию составляет 20% от категории. Для каждого клиента и водителя доступно формирование и просмотр личной истории поездок.

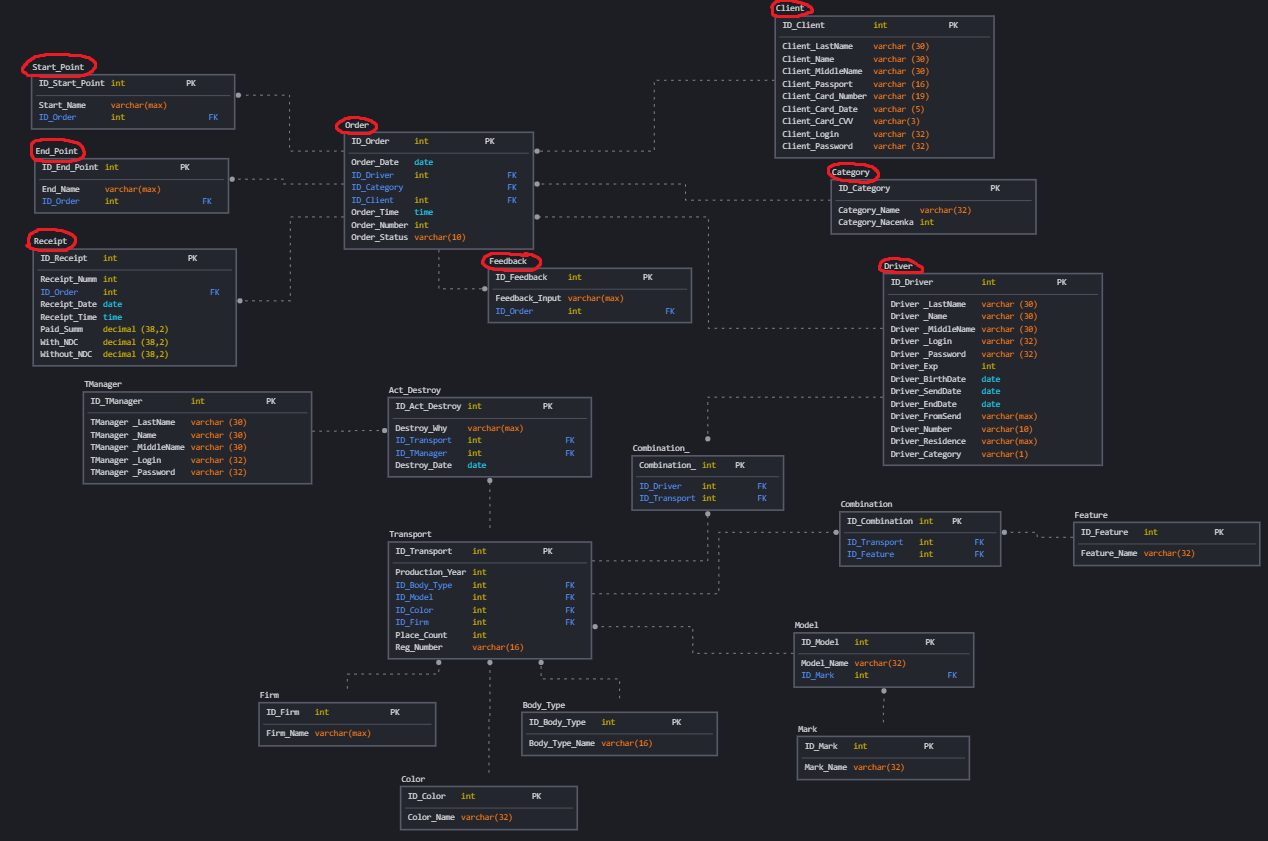
* + 1. Описание подсистемы

Для оформления заказа, клиент должен выбрать пункт отправления и пункт назначения, при условии, что пункты не совпадают (дополнительно клиент может добавить промежуточные точки остановки, которые увеличивают стоимость на 30% от стоимости категории). После этого любой водитель, который не занят сейчас на заказе, принимает заказ и автоматически изменяет статус на «В пути», после того как пассажиры произвели посадку, водитель меняет статус на «В дороге». После окончания заказа, водитель завершает заказ со статусом «Закрыт», у клиента экспортируется чек с детализацией поездки. Водитель и клиент могут оставить отзыв и оценку друг о друге.

* 1. Инфологическая модель данных



* 1. Даталогическая модель данных



* 1. Словарь данных

|  |  |  |  |
| --- | --- | --- | --- |
| Ключ | Наименование | Тип данных | Примечание |
| Start\_Point | | | |
| PK | ID\_Start\_Point | INT | Первичный ключ таблицы «Пункт отправления» |
|  | Start\_Name | VARCHAR(MAX) |  |
| FK | ID\_Order | INT |  |
| End\_Point | | | |
| PK | ID\_End\_Point | INT | Первичный ключ таблицы «Пункт назначения» |
|  | End\_Name | VARCHAR(MAX) |  |
| FK | ID\_Order | INT |  |
| Receipt | | | |
| PK | ID\_Receipt | INT | Первичный ключ таблицы «Чек» |
|  | Receipt\_Numm | INT |  |
|  | Receipt\_Date | Date | Текущая дата |
|  | Receipt\_Time | Time | Текущее время |
|  | Paid\_Summ | DECIMAL (38,2) | Не отрицательное |
|  | With\_NDC | DECIMAL (38,2) | Не отрицательное |
|  | Without\_NDC | DECIMAL (38,2) | Не отрицательное |
| FK | ID\_Order | INT |  |
| Order | | | |
| PK | ID\_Order | INT | Первичный ключ таблицы «Заказ» |
|  | Order\_Date | Date | Текущая дата |
|  | Order\_Time | Time | Текущее время |
|  | Order\_Number | INT |  |
|  | Order\_Status | VARCHAR(10) | «В пути», «Закрыт» |
| FK | ID\_Category | INT |  |
| FK | ID\_Client | INT |  |
| FK | ID\_Driver | INT |  |
| Client | | | |
| PK | ID\_Client | INT | Первичный ключ таблицы «Клиент» |
|  | Client\_LastName | VARCHAR (30) |  |
|  | Client\_Name | VARCHAR (30) |  |
|  | Client\_MiddleName | VARCHAR (30) |  |
|  | Client\_Passport | VARCHAR (16) | Уникальное поле |
|  | Client\_Card\_Number | VARCHAR (19) | Уникальное поле, 0000 0000 0000 0000 |
|  | Client\_Card\_Date | VARCHAR (5) | MM/YY |
|  | Client\_Card\_CVV | VARCHAR(3) | 000 |
|  | Client\_Login | VARCHAR (32) | Не менее 8 символов,  уникальное поле |
|  | Client\_Password | VARCHAR (32) | Спец символы, минимум одна  заглавная одна прописная  латинская буквы |
| Category | | | |
| PK | ID\_ Category | INT | Первичный ключ таблицы «Категория» |
|  | Category\_Name | VARCHAR(32) | Уникальное поле |
|  | Category\_Nacenka | INT |  |
| Feedback | | | |
| PK | ID\_Feedback | INT | Первичный ключ таблицы «Отзыв» |
|  | Feedback\_Input | VARCHAR(MAX) |  |
| FK | ID\_Order | INT |  |
| Driver | | | |
| PK | ID\_Driver | INT | Первичный ключ таблицы «Водитель» |
|  | Driver \_LastName | VARCHAR (30) |  |
|  | Driver \_Name | VARCHAR (30) |  |
|  | Driver \_MiddleName | VARCHAR (30) |  |
|  | Driver \_Login | VARCHAR (32) | Не менее 8 символов,  уникальное поле |
|  | Driver \_Password | VARCHAR (32) | Спец символы, минимум одна  заглавная одна прописная  латинская буквы |
|  | Driver\_Exp | INT |  |
|  | Driver\_BirthDate | Date | До текущей даты |
|  | Driver\_SendDate | Date | Текущая дата |
|  | Driver\_EndDate | Date | После текущей даты |
|  | Driver\_FromSend | VARCHAR(MAX) |  |
|  | Driver\_Number | VARCHAR(10) | уникальное поле |
|  | Driver\_Residence | VARCHAR(MAX) |  |
|  | Driver\_Category | VARCHAR(1) |  |

1. РЕАЛИЗАЦИЯ
   1. Описание модулей приложения

Таблица 3.1 – Описание модулей

| № ПП | Название файла | Размер в КБ | Краткое описание |
| --- | --- | --- | --- |
| 1 | DataSetClass\_Taxiapp.sln | 2 | Файл решения проекта |
| 2 | AdminWindow.xaml.cs | 18 | Реализация окна администратора |
| 3 | MainWindow.xaml.cs | 18 | Реализация основного окна с подключением к базе данных и авторизацией пользователей |
| 4 | LicencesWindow.xaml.cs | 2 | Реализация окна проверки лицензии приложения. |
| 5 | FPSMonitorWindow.xaml.cs | 4 | Реализация окна с мониторингом загруженности ЦПУ, ОЗУ, HDD/SSD дисков и проверка скорости сети. |
| 6 | ClientWindow.xaml.cs | 13 | Реализация окна клиента, создание заказов, истории заказов, экспорт чека. |
| 7 | DriverWindow.xaml.cs | 4 | Реализация окна водителя, принятие заказов, история заказов. |
| 8 | SendFeedbackWindow.xaml.cs | 3 | Реализация окна отправки отзыва для клиента и водителя. |
| 9 | SendMailWindow.xaml.cs | 3 | Реализация окна отправки кода подтверждения авторизации. |
| 10 | messageWindow.xaml.cs | 1 | Реализация окна, отображающего текст выхода из программы. |
| 11 | App.xaml.cs | ~1 | Подключение дополнительных модулей к программе. |

* 1. Программный код модулей

using Microsoft.Win32;

using ModernWpf;

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.Diagnostics;

using System.IO;

using System.Linq;

using System.Net;

using System.Net.Mail;

using System.Security.Cryptography;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using System.Windows.Threading;

namespace DataSetClass\_Taxiapp

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

public static SQL session;

public string BD, SERVER;

Boolean quest = true;

ThemeManager tm = ThemeManager.Current;

RegistryKey registry = Registry.CurrentUser;

public static DispatcherTimer waitTimer = new DispatcherTimer();

public MainWindow()

{

InitializeComponent();

//Guid guid = Guid.NewGuid();

//regSet("key", guid.ToString());

string key = regGet("key");

if (checkLicence(key)) new LicencesWindow().ShowDialog();

checkAppInstall();

timerStart();

}

public static bool checkLicence(string key)

{

string[] licences = (File.Exists("licences")) ? File.ReadAllLines("licences") : new string[] { };

foreach (var l in licences)

{

if (key == Decrypt(l, "key123")) return false;

}

return true;

}

public void timerStart()

{

waitTimer.Interval = TimeSpan.FromSeconds(120);

waitTimer.Tick += TimerMessage;

waitTimer.Start();

}

private void Window\_MouseMove(object sender, MouseEventArgs e)

{

waitTimer.Stop();

waitTimer.Start();

}

private void TimerMessage(object sender, EventArgs e)

{

ModernWpf.MessageBox.Show("Вы слишком долго ничего не делали.");

}

public void checkAppInstall()

{

try

{

if (Type.GetTypeFromProgID("Word.Application") == null) { ModernWpf.MessageBox.Show("У вас не установлен Microsoft Word, из-за чего некоторый функционал программы может быть недоступен и/или программа может работать некорректно."); }

if (Type.GetTypeFromProgID("Excel.Application") == null) { ModernWpf.MessageBox.Show("У вас не установлен Microsoft Excel, из-за чего некоторый функционал программы может быть недоступен и/или программа может работать некорректно."); }

if (!(Directory.Exists(@"C:\Program Files\Microsoft SQL Server"))) { ModernWpf.MessageBox.Show("У вас не установлен Microsoft SQL Server, установите его для продолжения работы с программой."); Application.Current.Shutdown(); };

}

catch (Exception e)

{

exMessage(e);

}

}

private void closingRequest(object sender, System.ComponentModel.CancelEventArgs e)

{

if (!quest) return;

messageWindow msgwindow = new messageWindow();

msgwindow.ShowDialog();

if (msgwindow.ex == true)

{

// RestoreBounds - Возвращает размер и расположение окна перед тем как оно было свернуто или развернуто.

//Properties.Settings.Default.WindowPosition = this.RestoreBounds;

// Сохранение настроек.

//Properties.Settings.Default.Save();

regSet("Left", Left.ToString());

regSet("Top", Top.ToString());

e.Cancel = false;

Application.Current.Shutdown();

}

else e.Cancel = true;

}

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

{

string temp = regGet("Left");

if(temp != null && temp !="") Left = Convert.ToDouble(temp);

temp = regGet("Top");

if(temp != null && temp != "") Top = Convert.ToDouble(temp);

//Left = Properties.Settings.Default.WindowPosition.Left;

//Top = Properties.Settings.Default.WindowPosition.Top;

SERVER = regGet("SERVER");

BD = regGet("BD");

if(regGet("Theme") == "Light") LightTheme();

else DarkTheme();

Login\_tb.Text = regGet("Login");

Password\_tb.Password = regGet("Password");

if (SERVER != null && BD != null)

{

Thread connectionThread = new Thread(connection);

connectionThread.Start();

}

else

{

status\_tb.Text = "Ошибка подключения";

status\_pr.IsActive = false;

connection\_grid.Visibility = Visibility.Visible;

}

}

private string regGet(string textload, string encryptKey = "key123")

{

RegistryKey key = registry.CreateSubKey("TaxiDBConfig");

textload = key.GetValue(textload)?.ToString();

return Decrypt(textload, encryptKey);

}

private void regSet(string keyname, string textsave, string encryptKey = "key123")

{

RegistryKey key = registry.CreateSubKey("TaxiDBConfig");

textsave = Encrypt(textsave, encryptKey);

key.SetValue(keyname, textsave);

}

public static string Encrypt(string ishText, string pass, string sol = "doberman", string cryptographicAlgorithm = "SHA1", int passIter = 2, string initVec = "a8doSuDitOz1hZe#", int keySize = 256)

{

if (string.IsNullOrEmpty(ishText)) return "";

byte[] initVecB = Encoding.ASCII.GetBytes(initVec);

byte[] solB = Encoding.ASCII.GetBytes(sol);

byte[] ishTextB = Encoding.UTF8.GetBytes(ishText);

PasswordDeriveBytes derivPass = new PasswordDeriveBytes(pass, solB, cryptographicAlgorithm, passIter);

byte[] keyBytes = derivPass.GetBytes(keySize / 8);

RijndaelManaged symmK = new RijndaelManaged();

symmK.Mode = CipherMode.CBC;

byte[] cipherTextBytes = null;

using (ICryptoTransform encryptor = symmK.CreateEncryptor(keyBytes, initVecB))

{

using (MemoryStream memStream = new MemoryStream())

{

using (CryptoStream cryptoStream = new CryptoStream(memStream, encryptor, CryptoStreamMode.Write))

{

cryptoStream.Write(ishTextB, 0, ishTextB.Length);

cryptoStream.FlushFinalBlock();

cipherTextBytes = memStream.ToArray();

memStream.Close();

cryptoStream.Close();

}

}

}

symmK.Clear();

return Convert.ToBase64String(cipherTextBytes);

}

public static string Decrypt(string ciphText, string pass, string sol = "doberman", string cryptographicAlgorithm = "SHA1", int passIter = 2, string initVec = "a8doSuDitOz1hZe#", int keySize = 256)

{

if (string.IsNullOrEmpty(ciphText)) return "";

byte[] initVecB = Encoding.ASCII.GetBytes(initVec);

byte[] solB = Encoding.ASCII.GetBytes(sol);

byte[] cipherTextBytes = Convert.FromBase64String(ciphText);

PasswordDeriveBytes derivPass = new PasswordDeriveBytes(pass, solB, cryptographicAlgorithm, passIter);

byte[] keyBytes = derivPass.GetBytes(keySize / 8);

RijndaelManaged symmK = new RijndaelManaged();

symmK.Mode = CipherMode.CBC;

byte[] plainTextBytes = new byte[cipherTextBytes.Length];

int byteCount = 0;

using (ICryptoTransform decryptor = symmK.CreateDecryptor(keyBytes, initVecB))

{

using (MemoryStream mSt = new MemoryStream(cipherTextBytes))

{

using (CryptoStream cryptoStream = new CryptoStream(mSt, decryptor, CryptoStreamMode.Read))

{

byteCount = cryptoStream.Read(plainTextBytes, 0, plainTextBytes.Length);

mSt.Close();

cryptoStream.Close();

}

}

}

symmK.Clear();

return Encoding.UTF8.GetString(plainTextBytes, 0, byteCount);

}

private void Find\_BD\_Click(object sender, RoutedEventArgs e)

{

Find\_bt.IsEnabled = false;

status\_tb.Text = "Поиск БД";

status\_pr.IsActive = true;

string temp = Server\_tb.Text;

Thread findTr = new Thread(() =>

{

SQL master = new SQL(temp, "master");

DataTable bds = master.selectTable("select name from sys.databases");

bool bd = false;

for (int i = 4; i < bds.Rows.Count; i++) if (bds.Rows[i][0].ToString() == "Taxi\_DB") bd = true;

Application.Current.Dispatcher.Invoke(new Action(() =>

{

try

{

BD\_cb.Items.Clear();

BD\_cb.IsEnabled = true;

connection\_bt.IsEnabled = true;

for (int i = 4; i < bds.Rows.Count; i++)

{

BD\_cb.Items.Add(bds.Rows[i][0].ToString());

}

BD\_cb.SelectedIndex = 0;

if (BD\_cb.Items.Count > 0)

{

status\_tb.Text = "Выберите БД";

if (!bd) { ModernWpf.MessageBox.Show("У вас не установлен БД для работы с программой, установите Taxi\_BD.sql из папки с программой"); Application.Current.Shutdown(); };

}

else status\_tb.Text = "Введён неправильный сервер";

}

catch (Exception ex)

{

exMessage(ex);

}

Find\_bt.IsEnabled = true;

status\_pr.IsActive = false;

}));

});

findTr.Start();

}

private void connection\_bt\_Click(object sender, RoutedEventArgs e)

{

SERVER = Server\_tb.Text;

BD = BD\_cb.SelectedItem.ToString();

status\_tb.Text = "Подключение к базе данных...";

status\_pr.IsActive = true;

if (SelectedAuth\_cb.SelectedIndex == 0) {

Thread connectionThread = new Thread(connection);

connectionThread.Start();

}

else{

session = new SQL(SERVER, BD, SelectedAuth\_Log.Text, SelectedAuth\_Pass.Password);

try

{

session.sql.Open();

session.sql.Close();

new AdminWindow().Show();

quest = false;

this.Close();

}

catch (Exception ex) {

exMessage(ex);

}

}

}

private void change\_bt\_Click(object sender, RoutedEventArgs e)

{

Login\_bt.IsEnabled = false;

Login\_tb.IsEnabled = false;

status\_tb.Text = "Измените базу данных.";

Password\_tb.IsEnabled = false;

connection\_grid.Visibility = Visibility.Visible;

change\_bt.Visibility = Visibility.Hidden;

}

private void Login\_bt\_Click(object sender, RoutedEventArgs e)

{

try

{

DataTable Client = session.selectTable($@"select \* from dbo.Client WHERE Client\_Login = '{Login\_tb.Text}' and Client\_Password = '{Password\_tb.Password}'");

DataTable Driver = session.selectTable($@"select \* from dbo.Driver WHERE Driver\_Login = '{Login\_tb.Text}' and Driver\_Password = '{Password\_tb.Password}'");

if (Client.Rows.Count > 0)

{

SendMailWindow sendMailWindow = new SendMailWindow(Client.Rows[0][2].ToString());

sendMailWindow.ShowDialog();

if (sendMailWindow.DialogResult == false) return;

else new ClientWindow(Client).Show();

}

else if (Driver.Rows.Count > 0)

{

SendMailWindow sendMailWindow = new SendMailWindow(Driver.Rows[0][2].ToString());

sendMailWindow.ShowDialog();

if (sendMailWindow.DialogResult == false) return;

else new DriverWindow(Driver).Show();

}

else { ModernWpf.MessageBox.Show("Неверный Логин или/и Пароль"); return; }

if (remember\_checkb.IsChecked == true)

{

regSet("Login", Login\_tb.Text);

regSet("Password", Password\_tb.Password);

}

else {

regSet("Login", "");

regSet("Password", "");

}

quest = false;

this.Close();

}

catch (Exception ex)

{

exMessage(ex);

}

}

private void SelectedAuth\_cb\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

if (SelectedAuth\_Log == null || SelectedAuth\_Pass == null) return;

if(SelectedAuth\_cb.SelectedIndex == 0)

{

SelectedAuth\_Log.Visibility = Visibility.Hidden;

SelectedAuth\_Pass.Visibility = Visibility.Hidden;

}

else

{

SelectedAuth\_Log.Visibility = Visibility.Visible;

SelectedAuth\_Pass.Visibility = Visibility.Visible;

}

}

public void connection()

{

session = new SQL(SERVER,BD);

try

{

session.sql.Open();

DataTable schema = session.sql.GetSchema("Tables");

Thread.Sleep(3000);

Application.Current.Dispatcher.Invoke(new Action(() =>

{

if (schema.Rows.Count != 19) ModernWpf.MessageBox.Show($"Количество таблиц в БД не совпадает ({schema.Rows.Count}/19), возможно БД установленна не корректно.");

status\_pr.IsActive = false;

Login\_bt.IsEnabled = true;

Login\_tb.IsEnabled = true;

Password\_tb.IsEnabled = true;

connection\_grid.Visibility = Visibility.Hidden;

change\_bt.Visibility = Visibility.Visible;

regSet("SERVER", SERVER);

regSet("BD", BD);

status\_tb.Text = $"Подключено к {BD}, авторизуйтесь.";

}));

}

catch (Exception)

{

Application.Current.Dispatcher.Invoke(new Action(() =>

{

status\_tb.Text = "Ошибка подключения";

status\_pr.IsActive = false;

connection\_grid.Visibility = Visibility.Visible;

}));

}

finally

{

session.sql.Close();

}

}

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

{

if (tm.ActualApplicationTheme == ApplicationTheme.Light) DarkTheme();

else LightTheme();

}

public void LightTheme()

{

tm.ApplicationTheme = ApplicationTheme.Light;

Background = (SolidColorBrush)new BrushConverter().ConvertFromString("#FFFFFFFF");

regSet("Theme", "Light");

}

public void DarkTheme()

{

tm.ApplicationTheme = ApplicationTheme.Dark;

Background = (SolidColorBrush)new BrushConverter().ConvertFromString("#1F2023");

regSet("Theme", "Dark");

}

public static void exMessage(Exception e)

{

Application.Current.Dispatcher.Invoke(new Action(() =>

{

ModernWpf.MessageBox.Show(e.Message);

}));

LogExcep(e);

}

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

{

bool isWindowOpen = false;

foreach (Window w in Application.Current.Windows)

{

if (w is FPSMonitorWindow)

{

isWindowOpen = true;

w.Activate();

}

}

if (!isWindowOpen)

{

new FPSMonitorWindow().Show(); ;

}

}

public static void LogExcep(Exception e)

{

if (!File.Exists("ErrorLogs.txt")){ File.Create("ErrorLogs.txt").Close(); File.WriteAllText("ErrorLogs.txt", "Log file is initialized"); }

string temp = File.ReadAllText("ErrorLogs.txt");

temp += $"\n{DateTime.Now} {e.Message}";

File.WriteAllText("ErrorLogs.txt", temp);

}

}

} using Microsoft.Win32;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

using System.Windows.Threading;

namespace DataSetClass\_Taxiapp

{

/// <summary>

/// Логика взаимодействия для LicencesWindow.xaml

/// </summary>

public partial class LicencesWindow : Window

{

public static DispatcherTimer trialTimer = new DispatcherTimer();

bool exitvopros = true;

public LicencesWindow()

{

InitializeComponent();

}

private void Window\_Closing(object sender, System.ComponentModel.CancelEventArgs e)

{

if(exitvopros) Application.Current.Shutdown();

}

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

{

exitvopros = false;

trialTimer.Interval = TimeSpan.FromSeconds(5);

trialTimer.Tick += TimerEnd;

trialTimer.Start();

this.Close();

}

private void TimerEnd(object sender, EventArgs e)

{

ModernWpf.MessageBox.Show("Время закочилось :3");

Application.Current.Shutdown();

}

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

{

try

{

string textkey = MainWindow.Encrypt(licence\_tb.Text, "key123");

if (MainWindow.checkLicence(textkey))

{

RegistryKey key = Registry.CurrentUser.CreateSubKey("TaxiDBConfig");

key.SetValue("key", textkey);

ModernWpf.MessageBox.Show("Успешно активированно.");

exitvopros = false;

this.Close();

}

else

{

ModernWpf.MessageBox.Show("Ключ неверный.");

}

}

catch (Exception ex)

{

MainWindow.exMessage(ex);

}

}

}

} using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace DataSetClass\_Taxiapp

{

/// <summary>

/// Логика взаимодействия для AdminWindow.xaml

/// </summary>

public partial class AdminWindow : Window

{

Boolean quest = false;

SQL session = MainWindow.session;

int selectedClientID, selectedDriverID, selectedOrderID, selectedFeedbackID, selectedStartPointID, selectedEndPointID;

public AdminWindow()

{

InitializeComponent();

Background = App.Current.MainWindow.Background;

}

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

{

try

{

string key = File.ReadAllText("licences");

string newkeys = "";

for (int i = 1; i < Convert.ToInt32(Count\_tb.Text); i++)

{

Guid guid = Guid.NewGuid();

key += MainWindow.Encrypt(guid.ToString(), "key123")+"\n";

newkeys += guid.ToString() + "\n";

}

File.WriteAllText("licences", key);

File.WriteAllText("newkeys", newkeys);

}catch (Exception ex)

{

MainWindow.exMessage(ex);

}

}

private void closingRequest(object sender, System.ComponentModel.CancelEventArgs e)

{

if (quest) return;

messageWindow msgwindow = new messageWindow();

msgwindow.ShowDialog();

if (msgwindow.ex == true)

{

e.Cancel = false;

Application.Current.Shutdown();

}

else e.Cancel = true;

}

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

{

new MainWindow().Show();

quest = true;

this.Close();

}

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

{

try

{

ClientDataGridFill();

DriverDataGridFill();

OrderDataGridFill();

CommentDataGridFill();

StartPointDataGridFill();

EndPointDataGridFill();

}

catch (Exception ex)

{

MainWindow.exMessage(ex);

}

}

public void ClientDataGridFill()

{

Client\_dg.Columns.Clear();

Client\_dg.ItemsSource = session.selectTable($@"select [ID\_Client] as 'ID',[Client\_LastName] as 'Фамилия',[Client\_Name] as 'Имя',[Client\_MiddleName] as 'Отчество', [Client\_Login] as 'Логин', [Client\_Password] as 'Пароль', [Client\_Passport] as 'Паспорт', [Client\_Card\_Number] as 'Номер карты',[Client\_Card\_Date] as 'Срок карты',[Client\_Card\_CVV] as 'CVV' from [dbo].[Client] where [Client\_Name] like '%{SearchClient\_tb.Text}%' order by [Client\_LastName] {Sort\_cb.Text}").DefaultView;

//Client\_dg.Columns[0].Visibility = Visibility.Hidden;

}

private void Client\_dg\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

if (!(Client\_dg.SelectedItem is DataRowView)) return;

DataRowView selected = Client\_dg.SelectedItem as DataRowView;

selectedClientID = (int)selected[0];

ClientLastName\_tb.Text = selected[1].ToString();

ClientName\_tb.Text = selected[2].ToString();

ClientMiddleName\_tb.Text = selected[3].ToString();

ClientLogin\_tb.Text = selected[4].ToString();

ClientPass\_tb.Text = selected[5].ToString();

ClientPassport\_tb.Text = selected[6].ToString();

CardNumber\_tb.Text = selected[7].ToString();

CardDate\_tb.Text = selected[8].ToString();

CardCVV\_tb.Text= selected[9].ToString();

}

private void Client\_Save\_bt\_Click(object sender, RoutedEventArgs e)

{

session.action($"update [dbo].[Client] set [Client\_LastName]='{ClientLastName\_tb.Text}', [Client\_Name]='{ClientName\_tb.Text}',[Client\_MiddleName]='{ClientMiddleName\_tb.Text}',[Client\_Login]='{ClientLogin\_tb.Text}',[Client\_Password]='{ClientPass\_tb.Text}',[Client\_Passport]='{ClientPassport\_tb.Text}',[Client\_Card\_Number]='{CardNumber\_tb.Text}',[Client\_Card\_Date]='{CardDate\_tb.Text}',[Client\_Card\_CVV]='{CardCVV\_tb.Text}' where[ID\_Client] = '{selectedClientID}'");

ClientDataGridFill();

}

private void Client\_Add\_bt\_Click(object sender, RoutedEventArgs e)

{

session.action($"insert into [dbo].[Client] ([Client\_LastName],[Client\_Name],[Client\_MiddleName],[Client\_Passport],[Client\_Card\_Number],[Client\_Card\_Date],[Client\_Card\_CVV],[Client\_Login],[Client\_Password]) values('{ClientLastName\_tb.Text}', '{ClientName\_tb.Text}', '{ClientMiddleName\_tb.Text}', '{ClientPassport\_tb.Text}', '{CardNumber\_tb.Text}', '{CardDate\_tb.Text}', '{CardCVV\_tb.Text}', '{ClientLogin\_tb.Text}', '{ClientPass\_tb.Text}')");

ClientDataGridFill();

}

private void Client\_Del\_bt\_Click(object sender, RoutedEventArgs e)

{

session.action($"delete from [dbo].[Client] where [ID\_Client]='{selectedClientID}'");

ClientDataGridFill();

}

public void DriverDataGridFill()

{

Driver\_dg.Columns.Clear();

Driver\_dg.ItemsSource = session.selectTable($@"select [ID\_Driver] as 'ID', [Driver\_LastName] as 'Фамилия',[Driver\_Name] as 'Имя', [Driver\_MiddleName] as 'Отчество', [Driver\_Login] as 'Логин', [Driver\_Password] as 'Пароль', [Driver\_Exp] as 'Стаж (лет)', [Driver\_BirthDate] as 'Дата рождения',[Driver\_SendDate] as 'Дата выдачи', [Driver\_EndDate] as 'Дата окончания', [Driver\_FromSend] as 'Кем выдан',[Driver\_Number] as 'Серия и номер',[Driver\_Residence] as 'Место жительства', [Driver\_Category] as 'Категория' from [dbo].[Driver] where [Driver\_Name] like '%{SearchDriver\_tb.Text}%'").DefaultView;

//Driver\_dg.Columns[0].Visibility = Visibility.Hidden;

}

private void Driver\_dg\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

if (!(Driver\_dg.SelectedItem is DataRowView)) return;

DataRowView selected = Driver\_dg.SelectedItem as DataRowView;

selectedDriverID = (int)selected[0];

DriverLastName\_tb.Text = selected[1].ToString();

DriverName\_tb.Text = selected[2].ToString();

DriverMiddleName\_tb.Text = selected[3].ToString();

DriverLogin\_tb.Text = selected[4].ToString();

DriverPass\_tb.Text = selected[5].ToString();

DriverExp\_tb.Text = selected[6].ToString();

DriverBirthDate\_tb.Text = selected[7].ToString();

DriverStartDate\_tb.Text = selected[8].ToString();

DriverEndDate\_tb.Text = selected[9].ToString();

Driverkv\_tb.Text = selected[10].ToString();

DriverNumber\_tb.Text= selected[11].ToString();

DriverMestoShitelstva\_tb.Text = selected[12].ToString();

DriverCategory\_tb.Text = selected[13].ToString();

}

private void Driver\_Save\_bt\_Click(object sender, RoutedEventArgs e)

{

session.action($"update [dbo].[Driver] set [Driver\_LastName] = '{DriverLastName\_tb.Text}',[Driver\_Name] = '{DriverName\_tb.Text}',[Driver\_MiddleName] = '{DriverMiddleName\_tb.Text}',[Driver\_Login] = '{DriverLogin\_tb.Text}',[Driver\_Password] = '{DriverPass\_tb.Text}',[Driver\_Exp] = '{DriverExp\_tb.Text}',[Driver\_BirthDate] = '{DriverBirthDate\_tb.Text}',[Driver\_SendDate] = '{DriverStartDate\_tb.Text}',[Driver\_EndDate] = '{DriverEndDate\_tb.Text}',[Driver\_FromSend] = '{Driverkv\_tb.Text}',[Driver\_Number] = '{DriverNumber\_tb.Text}',[Driver\_Residence] = '{DriverMestoShitelstva\_tb.Text}',[Driver\_Category] = '{DriverCategory\_tb.Text}'");

DriverDataGridFill();

}

private void Driver\_Add\_bt\_Click(object sender, RoutedEventArgs e)

{

session.action($"insert into [dbo].[Driver] ([Driver\_LastName],[Driver\_Name],[Driver\_MiddleName],[Driver\_Login],[Driver\_Password],[Driver\_Exp],[Driver\_BirthDate],[Driver\_SendDate],[Driver\_EndDate],[Driver\_FromSend],[Driver\_Number],[Driver\_Residence],[Driver\_Category]) values('{DriverLastName\_tb.Text}', '{DriverName\_tb.Text}', '{DriverMiddleName\_tb.Text}', '{DriverLogin\_tb.Text}', '{DriverPass\_tb.Text}', '{DriverExp\_tb.Text}', '{DriverBirthDate\_tb.Text}', '{DriverStartDate\_tb.Text}', '{DriverEndDate\_tb.Text}', '{Driverkv\_tb.Text}', '{DriverNumber\_tb.Text}', '{DriverMestoShitelstva\_tb.Text}', '{DriverCategory\_tb.Text}')");

DriverDataGridFill();

}

private void Driver\_Del\_bt\_Click(object sender, RoutedEventArgs e)

{

session.action($"delete from [dbo].[Driver] where [ID\_Driver]='{selectedDriverID}'");

DriverDataGridFill();

}

private void dataGridBugFix(object sender, DataGridAutoGeneratingColumnEventArgs e)

{

PropertyDescriptor propertyDescriptor = (PropertyDescriptor)e.PropertyDescriptor;

e.Column.Header = propertyDescriptor.DisplayName;

if (propertyDescriptor.DisplayName == "ID") e.Column.Visibility = Visibility.Hidden;

if (e.PropertyType == typeof(DateTime)) (e.Column as DataGridTextColumn).Binding.StringFormat = "dd/MM/yyyy";

}

public void OrderDataGridFill()

{

Order\_dg.Columns.Clear();

Order\_dg.ItemsSource = session.selectTable($@"select [ID\_Order] as 'ID' ,[Order\_Date] as 'Дата заказа',[Order\_Time] as 'Время заказа',[Order\_Number] as 'Номер заказа',[Order\_Status] as 'Статус заказа', [Category\_ID] as 'Категория', [Client\_ID] as 'Клиент', [Driver\_ID] as 'Водитель' from [dbo].[Order] inner join [dbo].[Category] on [Category\_ID] = [ID\_Category] inner join [dbo].[Client] on [Client\_ID] = [ID\_Client] inner join [dbo].[Driver] on [Driver\_ID] = [ID\_Driver] where [Order\_Status] like '%{SearchOrder\_tb.Text}%'").DefaultView;

}

private void Order\_dg\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

if (!(Order\_dg.SelectedItem is DataRowView)) return;

DataRowView selected = Order\_dg.SelectedItem as DataRowView;

selectedOrderID = (int)selected[0];

OrderDate\_tb.Text = selected[1].ToString();

OrderTime\_tb.Text = selected[2].ToString();

OrderNumber\_tb.Text = selected[3].ToString();

OrderStatus\_tb.Text = selected[4].ToString();

Category\_tb.Text = selected[5].ToString();

Client\_tb.Text = selected[6].ToString();

Driver\_tb.Text = selected[7].ToString();

}

private void Order\_Save\_bt\_Click(object sender, RoutedEventArgs e)

{

session.action($"update [dbo].[Order] set [Order\_Date] = '{OrderDate\_tb.Text}', [Order\_Time] = '{OrderTime\_tb.Text}', [Order\_Number] = '{OrderNumber\_tb.Text}', [Order\_Status] = '{OrderStatus\_tb.Text}', [Category\_ID] = '{Category\_tb.Text}', [Client\_ID] = ''{Client\_tb.Text}, [Driver\_ID] = '{Driver\_tb.Text}' where [ID\_Order] = '{selectedOrderID}'");

OrderDataGridFill();

}

private void Order\_Add\_bt\_Click(object sender, RoutedEventArgs e)

{

session.action($"insert into [dbo].[Order] ([Order\_Number],[Order\_Status],[Category\_ID],[Client\_ID],[Driver\_ID]) values('{OrderNumber\_tb.Text}', '{OrderStatus\_tb.Text}', '{Category\_tb.Text}', '{Client\_tb.Text}', '{Driver\_tb.Text}')");

OrderDataGridFill();

}

private void Order\_Del\_bt\_Click(object sender, RoutedEventArgs e)

{

session.action($"delete from [dbo].[Order] where[ID\_Order] = {selectedOrderID}");

OrderDataGridFill();

}

public void CommentDataGridFill()

{

Comments\_dg.Columns.Clear();

Comments\_dg.ItemsSource = session.selectTable($@"select [ID\_Feedback] as 'ID', [Feedback\_Text] as 'Комментарий', [Feedback\_Mark] as 'Оценка', [Order\_ID] as [Заказ ID] from [dbo].[Feedback] where [Feedback\_Text] like '%{SearchComments\_tb.Text}%'").DefaultView;

}

private void Comments\_dg\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

if (!(Comments\_dg.SelectedItem is DataRowView)) return;

DataRowView selected = Comments\_dg.SelectedItem as DataRowView;

selectedFeedbackID = (int)selected[0];

CommentInputText\_tb.Text = selected[1].ToString();

CommentMark\_tb.Text = selected[2].ToString();

CommentOrderID\_tb.Text = selected[3].ToString();

}

private void StartPoint\_dg\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

if (!(StartPoint\_dg.SelectedItem is DataRowView)) return;

DataRowView selected = StartPoint\_dg.SelectedItem as DataRowView;

selectedStartPointID = (int)selected[0];

StartPoint\_tb.Text = selected[1].ToString();

StartPointOrderID\_tb.Text = selected[2].ToString();

}

public void StartPointDataGridFill()

{

StartPoint\_dg.Columns.Clear();

StartPoint\_dg.ItemsSource = session.selectTable($@"select [ID\_Start\_Point] as 'ID', [Start\_Point\_Name] as 'Пункт Назначения', [Order\_ID] as 'Заказ ID' from [dbo].[Start\_Point] where [Start\_Point\_Name] like '%{SearchStartAdr\_tb.Text}%'").DefaultView;

}

private void StartPoint\_Save\_bt\_Click(object sender, RoutedEventArgs e)

{

session.action($"update [dbo].[Start\_Point] set [Start\_Point\_Name] = '{StartPoint\_tb.Text}', [Order\_ID] = '{StartPointOrderID\_tb.Text}' where [ID\_Start\_Point] = {selectedStartPointID}");

StartPointDataGridFill();

}

private void StartPoint\_Add\_bt\_Click(object sender, RoutedEventArgs e)

{

session.action($"insert into [dbo].[Start\_Point] ([Start\_Point\_Name],[Order\_ID]) values('{StartPoint\_tb.Text}', {StartPointOrderID\_tb.Text})");

StartPointDataGridFill();

}

public void EndPointDataGridFill()

{

EndPoint\_dg.Columns.Clear();

EndPoint\_dg.ItemsSource = session.selectTable($@"select [ID\_End\_Point] as 'ID', [End\_Point\_Name] as 'Пункт Назначения', [Order\_ID] as 'Заказ ID' from [dbo].[End\_Point] where [End\_Point\_Name] like '%{SearchEndAdr\_tb.Text}%'").DefaultView;

}

private void EndPoint\_dg\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

if (!(EndPoint\_dg.SelectedItem is DataRowView)) return;

DataRowView selected = EndPoint\_dg.SelectedItem as DataRowView;

selectedEndPointID = (int)selected[0];

EndPoint\_tb.Text = selected[1].ToString();

EndPointOrderID\_tb.Text = selected[2].ToString();

}

private void EndPoint\_Save\_bt\_Click(object sender, RoutedEventArgs e)

{

session.action($"update [dbo].[End\_Point] set [End\_Point\_Name] = '{EndPoint\_tb.Text}', [Order\_ID] = '{EndPointOrderID\_tb.Text}' where [ID\_End\_Point] = {selectedEndPointID}");

EndPointDataGridFill();

}

private void EndPoint\_Add\_bt\_Click(object sender, RoutedEventArgs e)

{

session.action($"insert into [dbo].[End\_Point] ([End\_Point\_Name],[Order\_ID]) values('{EndPoint\_tb.Text}', {EndPointOrderID\_tb.Text})");

EndPointDataGridFill();

}

private void SearchStartAdr\_tb\_TextChanged(object sender, TextChangedEventArgs e)

{

StartPointDataGridFill();

}

private void SearchEndAdr\_tb\_TextChanged(object sender, TextChangedEventArgs e)

{

EndPointDataGridFill();

}

private void SearchClient\_tb\_TextChanged(object sender, TextChangedEventArgs e)

{

ClientDataGridFill();

}

private void SearchDriver\_tb\_TextChanged(object sender, TextChangedEventArgs e)

{

DriverDataGridFill();

}

private void SearchOrder\_tb\_TextChanged(object sender, TextChangedEventArgs e)

{

OrderDataGridFill();

}

private void SearchComments\_tb\_TextChanged(object sender, TextChangedEventArgs e)

{

CommentDataGridFill();

}

private void Sort\_cb\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

ClientDataGridFill();

}

private void EndPoint\_Del\_bt\_Click(object sender, RoutedEventArgs e)

{

session.action($"delete from [dbo].[End\_Point] where[ID\_End\_Point] = {selectedEndPointID}");

EndPointDataGridFill();

}

private void StartPoint\_Del\_bt\_Click(object sender, RoutedEventArgs e)

{

session.action($"delete from [dbo].[Start\_Point] where[ID\_Start\_Point] = {selectedStartPointID}");

StartPointDataGridFill();

}

private void Comment\_Save\_bt\_Click(object sender, RoutedEventArgs e)

{

session.action($"update [dbo].[Feedback] set [Feedback\_Text] = '{CommentInputText\_tb.Text}', [Feedback\_Mark] = {CommentMark\_tb.Text} where [ID\_Feedback] = '{selectedFeedbackID}'");

CommentDataGridFill();

}

private void Comment\_Add\_bt\_Click(object sender, RoutedEventArgs e)

{

session.action($"insert into [dbo].[Feedback] ([Feedback\_Text],[Feedback\_Mark],[Order\_ID]) values('{CommentInputText\_tb.Text}', {CommentMark\_tb.Text}, {CommentOrderID\_tb.Text})");

CommentDataGridFill();

}

private void Comment\_Del\_bt\_Click(object sender, RoutedEventArgs e)

{

session.action($"delete from [dbo].[Feedback] where [ID\_Feedback] = {selectedFeedbackID}");

CommentDataGridFill();

}

}

} using System;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace DataSetClass\_Taxiapp

{

/// <summary>

/// Логика взаимодействия для SendFeedbackWindow.xaml

/// </summary>

public partial class SendFeedbackWindow : Window

{

SQL session = MainWindow.session;

Boolean quest = false;

int selectedhistoryOrderID;

public SendFeedbackWindow(int selectedhistoryOrderID)

{

InitializeComponent();

this.selectedhistoryOrderID = selectedhistoryOrderID;

Background = App.Current.MainWindow.Background;

}

private void closingRequest(object sender, System.ComponentModel.CancelEventArgs e)

{

if (quest) return;

messageWindow msgwindow = new messageWindow();

msgwindow.ShowDialog();

if (msgwindow.ex == true)

{

e.Cancel = false;

Close();

}

else e.Cancel = true;

}

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

{

try

{

Thread thread = new Thread(() =>

{

DataTable Feedbacks = session.selectTable($"select [ID\_Feedback] as 'ID', [Feedback\_Text] as 'Текст отзыва', [Feedback\_Mark] as 'Оценка отзыва' from [dbo].[Feedback] where[Order\_ID] = '{selectedhistoryOrderID}'");

Application.Current.Dispatcher.Invoke(new Action(() =>

{

if (Feedbacks.Rows.Count == 0)

{

session.action($"insert into [dbo].[Feedback] ([Feedback\_Text],[Feedback\_Mark],[Order\_ID]) values('{Comment\_tb.Text}', {Mark\_cb.Text}, '{selectedhistoryOrderID}')");

ModernWpf.MessageBox.Show("Спасибо за отзыв!");

}

else

{

session.action($"update [dbo].[Feedback] set [Feedback\_Text] = '{Comment\_tb.Text}', [Feedback\_Mark] = '{Mark\_cb.Text}' where [Order\_ID] = '{selectedhistoryOrderID}'");

ModernWpf.MessageBox.Show("Отзыв обновлён!");

}

quest = true;

this.Close();

}));

});

thread.Start();

}

catch (Exception ex)

{

MainWindow.exMessage(ex);

}

}

}

}

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.IO;

using System.Linq;

using System.Net.NetworkInformation;

using System.Net.Sockets;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

using System.Windows.Threading;

using Microsoft.VisualBasic.Devices;

using LiveCharts;

using LiveCharts.Wpf;

namespace DataSetClass\_Taxiapp

{

/// <summary>

/// Логика взаимодействия для FPSMonitorWindow.xaml

/// </summary>

public partial class FPSMonitorWindow : Window

{

DispatcherTimer dispatcher = new DispatcherTimer();

PerformanceCounter CPU = new PerformanceCounter("Processor", "% Processor Time", "\_Total");

PerformanceCounter RAM = new PerformanceCounter("Memory", "Available MBytes");

ComputerInfo computerInfo = new ComputerInfo();

public ChartValues<double> CPULine { get; set; }

public ChartValues<double> RAMLine { get; set; }

public double RAMMax { get; set; }

public ChartValues<double> SpeedLine { get; set; }

public ChartValues<double> ROMLine { get; set; }

public FPSMonitorWindow()

{

InitializeComponent();

CPULine = new ChartValues<double> { 0,0,0,0,0,0 };

RAMLine = new ChartValues<double> { 0,0,0,0,0,0 };

SpeedLine = new ChartValues<double> { 0,0,0,0,0,0 };

ROMLine = new ChartValues<double> { 0,0,0,0,0, 0 };

RAMMax = computerInfo.TotalPhysicalMemory / 1024 / 1024;

Background = App.Current.MainWindow.Background;

staticInfo();

dispatcher.Interval = TimeSpan.FromSeconds(1);

dispatcher.Tick += updateDispatcher;

dispatcher.Start();

DataContext = this;

}

void updateDispatcher(object sender, EventArgs e)

{

double cpuload = CPU.NextValue();

double ramsave = RAM.NextValue();

CPUSAGE\_tb.Text = $"Загружен на {cpuload}%";

RAM\_tb.Text = $"Свободно {ramsave}/{RAMMax} MB";

CPULine.RemoveAt(0);

CPULine.Add(cpuload);

RAMLine.RemoveAt(0);

RAMLine.Add(ramsave);

CheckInternetSpeed();

ROM\_tb.Text = "";

foreach (var drive in DriveInfo.GetDrives())

{

ROM\_tb.Text += $"Диск {drive.Name}\n";

ROM\_tb.Text += $"{drive.AvailableFreeSpace / 1024 / 1024}/{drive.TotalSize / 1024 / 1024} MB\n";

if(drive.Name == @"C:\")

{

ROMLine.RemoveAt(0);

ROMLine.Add(drive.AvailableFreeSpace / 1024 / 1024);

}

}

}

public void CheckInternetSpeed()

{

Thread thread = new Thread(new ThreadStart(() =>

{

try

{

System.Net.WebClient wc = new System.Net.WebClient();

DateTime dt1 = DateTime.Now;

byte[] data = wc.DownloadData("http://google.com");

DateTime dt2 = DateTime.Now;

Application.Current.Dispatcher.Invoke(new Action(() =>

{

double curspeed = Math.Round((data.Length / 1024) / (dt2 - dt1).TotalSeconds, 2);

Network\_tb.Text = $"{curspeed} kbps\n";

SpeedLine.RemoveAt(0);

SpeedLine.Add(curspeed);

}));

}

catch (Exception) { }

}

));

thread.Start();

}

void staticInfo()

{

CPUCOUNT\_tb.Text = $"Кол-во ядер {Environment.ProcessorCount}";

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace DataSetClass\_Taxiapp

{

/// <summary>

/// Логика взаимодействия для ClientWindow.xaml

/// </summary>

public partial class ClientWindow : Window

{

DataTable Client;

SQL session = MainWindow.session;

Boolean quest = false;

public double price = 0;

public int countAddPoint = 0;

string ClientID;

int selectedhistoryOrderID;

public ClientWindow(DataTable Client)

{

InitializeComponent();

Background = App.Current.MainWindow.Background;

this.Client = Client;

}

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

{

updateHistory();

updateFiles();

}

public void updateHistory()

{

HistoryClient\_dg.Columns.Clear();

ClientID = Client.Rows[0][0].ToString();

HistoryClient\_dg.ItemsSource = session.selectTable($@"select [ID\_Order] as 'ID', [Order\_Number] as '№ Заказа',[Order\_Date] as 'Дата заказа',[Order\_Time] as 'Время заказа', [Order\_Status] as 'Статус заказа' from [dbo].[Order] where [Client\_ID] = '{ClientID}' AND [Order\_Number] like '%{SearchHistory\_tb.Text}%'").DefaultView;

}

private void dataGridBugFix(object sender, DataGridAutoGeneratingColumnEventArgs e)

{

PropertyDescriptor propertyDescriptor = (PropertyDescriptor)e.PropertyDescriptor;

e.Column.Header = propertyDescriptor.DisplayName;

if (propertyDescriptor.DisplayName == "ID") e.Column.Visibility = Visibility.Hidden;

if (e.PropertyType == typeof(DateTime)) (e.Column as DataGridTextColumn).Binding.StringFormat = "dd/MM/yyyy";

}

private void closingRequest(object sender, System.ComponentModel.CancelEventArgs e)

{

if (quest) return;

messageWindow msgwindow = new messageWindow();

msgwindow.ShowDialog();

if (msgwindow.ex == true)

{

e.Cancel = false;

Application.Current.Shutdown();

}

else e.Cancel = true;

}

public void CalculatePrice()

{

Thread thread = new Thread(() =>

{

Application.Current.Dispatcher.Invoke(new Action(() =>

{

if (PassagerCount\_cb == null || PassagerCount\_cb.Text == "") return;

int categoryPrice = 0;

if (Category\_cb.SelectedIndex == 0) categoryPrice = 170;

if (Category\_cb.SelectedIndex == 1) categoryPrice = 370;

if (Category\_cb.SelectedIndex == 2) categoryPrice = 570;

double addprice = countAddPoint \* (0.3 \* categoryPrice);

int temp = Convert.ToInt32(PassagerCount\_cb.Text);

price = (categoryPrice + (temp \* (categoryPrice \* 0.2))) + addprice;

price\_tb.Text = price.ToString() + "₽";

}));

});

thread.Start();

}

private void Category\_cb\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

CalculatePrice();

}

private void PassagerCount\_cb\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

CalculatePrice();

}

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

{

countAddPoint++;

CalculatePrice();

}

private void Button\_Click\_1(object sender, RoutedEventArgs e)

{

try

{

newOrder\_bt.IsEnabled = false;

int categoryID = (Category\_cb.Text == "Эконом") ? 1 : (Category\_cb.Text == "Комфорт") ? 2 : 3;

string StartPointAdress = StartPoint\_tb.Text, EndPointAdress = EndPoint\_tb.Text;

Thread thread = new Thread(() =>

{

Thread.CurrentThread.CurrentCulture = new System.Globalization.CultureInfo("en-US");

int lastID = Convert.ToInt32(session.selectTable("SELECT MAX(Order\_Number) FROM[dbo].[Order]").Rows[0][0]) + 1;

session.action($"insert into [dbo].[Order] ([Order\_Number],[Order\_Status],[Category\_ID],[Client\_ID],[Driver\_ID]) values('{lastID}', 'В пути', '{categoryID}', '{ClientID}', '1')");

if (StartPointAdress == "") { ModernWpf.MessageBox.Show("Поле ввода адреса отправления не должно быть пустое"); newOrder\_bt.IsEnabled = true; return; }

if (EndPointAdress == "") { ModernWpf.MessageBox.Show("Поле ввода адреса назначения не должно быть пустое"); newOrder\_bt.IsEnabled = true; return; }

if (StartPointAdress == EndPointAdress) { ModernWpf.MessageBox.Show("Изначальный и конечный пункты не должны совпадать"); newOrder\_bt.IsEnabled = true; return; }

session.action($"insert into [dbo].[Start\_Point] ([Start\_Point\_Name],[Order\_ID]) values ('{StartPointAdress}','{lastID}')");

session.action($"insert into [dbo].[End\_Point] ([End\_Point\_Name],[Order\_ID]) values ('{EndPointAdress}','{lastID}')");

int lastReceiptID = Convert.ToInt32(session.selectTable("SELECT MAX(Receipt\_Numm) FROM[dbo].[Receipt]").Rows[0][0]) + 1;

session.action($"insert into [dbo].[Receipt] ([Receipt\_Numm],[Paid\_Summ],[With\_NDC],[Without\_NDC],[Order\_ID]) values('{lastReceiptID}', '{price}', '{price}', '{price - (price \* 0.2)}', '{lastID}')");

Application.Current.Dispatcher.Invoke(new Action(() =>

{

updateHistory();

ModernWpf.MessageBox.Show($"Заказ №{lastID} успешно создан, ожидайте водителя");

newOrder\_bt.IsEnabled = true;

}));

});

thread.Start();

}

catch (Exception ex)

{

MainWindow.exMessage(ex);

}

}

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

{

new MainWindow().Show();

quest = true;

this.Close();

}

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

{

try

{

DataTable Receipt = session.selectTable($"select [Receipt\_Numm] as 'Номер чека',[Receipt\_Date] as 'Дата чека',[Receipt\_Time] as 'Время чека',[Paid\_Summ] as 'Внесённая сумма',[With\_NDC] as 'С НДС',[Without\_NDC] as 'БЕЗ НДС', [Order\_ID] as 'Order\_ID' from [dbo].[Receipt] inner join [dbo].[Order] on [Order\_ID] = [ID\_Order] where [Order\_ID] = '{selectedhistoryOrderID}'");

if(Receipt.Rows.Count == 0) { ModernWpf.MessageBox.Show("Чек ещё не создан"); return; }

Microsoft.Office.Interop.Word.Application app = new Microsoft.Office.Interop.Word.Application();

app.ShowAnimation = false;

var doc = app.Documents.Add(Visible: false);

var r = doc.Range();

var t = doc.Tables.Add(r,16, 2);

t.Borders.Enable = 0;

//OOO "СИТИ-МОБИЛ" | https://city-mobil.ru

//КАССОВЫЙ ЧЕК/ПРИХОД | 21.02.22 15:25

// | Чек N:24

//СНО: ОСН | ИНН: 7728697453

//ИТОГ | = 108

//ВСЕГО ПОЛУЧЕНО: | 108

//БЕЗНАЛИЧНЫМИ: | 108

//ИТОГО без НДС: | 108

//САЙТ | ФНС: www.nalog.gov.ru

//ЭЛ.АДР.ОТПРАВИТЕЛЯ | support@city-mobil.ru

//РН ККТ: 0002472508022739 | N ФН: 9960440301680011

//N ФД: 164354 | ФП: 2042698758

//N АВТ.: 3820017912070

//for (int i = 1; i <= 13; i++)

//{

// t.Columns[2].Pa =

//}

//t.Rows[2].Range.ParagraphFormat.Alignment = Microsoft.Office.Interop.Word.WdParagraphAlignment.wdAlignParagraphRight;

for(int i = 1; i < 16; i++)

{

t.Rows[i].Cells[2].Range.ParagraphFormat.Alignment = Microsoft.Office.Interop.Word.WdParagraphAlignment.wdAlignParagraphRight;

}

//t.Range.ParagraphFormat.Alignment = Microsoft.Office.Interop.Word.WdParagraphAlignment.wdAlignParagraphRight;

t.Rows[1].Cells[1].Range.Text = "OOO \"СИТИ - МОБИЛ\""; t.Rows[1].Cells[2].Range.Text = "https://city-mobil.ru";

t.Rows[2].Cells[1].Range.Text = "Дата"; t.Rows[2].Cells[2].Range.Text = $"{Receipt.Rows[0][1]}";//<----

t.Rows[3].Cells[1].Range.Text = "Время"; t.Rows[3].Cells[2].Range.Text = $"{Receipt.Rows[0][2]}";//<----

t.Rows[4].Cells[1].Range.Text = "КАССОВЫЙ ЧЕК/ПРИХОД"; t.Rows[4].Cells[2].Range.Text = $"Чек №{Receipt.Rows[0][0]}";//<----

t.Rows[5].Cells[1].Range.Text = "СНО: ОСН"; t.Rows[5].Cells[2].Range.Text = "ИНН: 7728697453";

t.Rows[6].Cells[1].Range.Text = "ИТОГ"; t.Rows[6].Cells[2].Range.Text = $"{Receipt.Rows[0][4]}";//<----

t.Rows[7].Cells[1].Range.Text = "ВСЕГО ПОЛУЧЕНО:"; t.Rows[7].Cells[2].Range.Text = $"{Receipt.Rows[0][3]}";//<----

t.Rows[8].Cells[1].Range.Text = "БЕЗНАЛИЧНЫМИ:"; t.Rows[8].Cells[2].Range.Text = $"{Receipt.Rows[0][3]}";//<----

t.Rows[9].Cells[1].Range.Text = "ИТОГО без НДС:"; t.Rows[9].Cells[2].Range.Text = $"{Receipt.Rows[0][5]}"; //<----

t.Rows[10].Cells[1].Range.Text = "САЙТ"; t.Rows[10].Cells[2].Range.Text = "ФНС: www.nalog.gov.ru";

t.Rows[11].Cells[1].Range.Text = "ЭЛ.АДР.ОТПРАВИТЕЛЯ"; t.Rows[11].Cells[2].Range.Text = "support@city-mobil.ru";

t.Rows[12].Cells[1].Range.Text = "РН ККТ: 0002472508022739"; t.Rows[12].Cells[2].Range.Text = "N ФН: 9960440301680011";

t.Rows[13].Cells[1].Range.Text = "N ФД: 164354"; t.Rows[13].Cells[2].Range.Text = "ФП: 2042698758";

t.Rows[14].Cells[1].Range.Text = "N АВТ.: 3820017912070"; t.Rows[14].Cells[2].Range.Text = " ";

t.Rows[15].Cells[1].Range.Text = "Дата создания документа"; t.Rows[15].Cells[2].Range.Text = DateTime.Now.ToString("dd/MM/yyyy HH:mm");

t.Rows[16].Cells[1].Range.Text = "Документ создан"; t.Rows[16].Cells[2].Range.Text = $"{Client.Rows[0][2].ToString()+ " " + Client.Rows[0][3].ToString()}";

doc.Save();

doc.Close();

app.Quit();

if (!File.Exists("CreatedFiles.txt")) { File.Create("CreatedFiles.txt").Close();}

string temp = File.ReadAllText("CreatedFiles.txt");

temp += $"\n{DateTime.Now.ToString("dd/MM/yyyy HH:mm")}";

File.WriteAllText("CreatedFiles.txt", temp);

updateFiles();

}

catch(Exception ex)

{

MainWindow.exMessage(ex);

}

}

private void HistoryClient\_dg\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

if (!(HistoryClient\_dg.SelectedItem is DataRowView)) return;

DataRowView selected = HistoryClient\_dg.SelectedItem as DataRowView;

selectedhistoryOrderID = (int)selected[0];

}

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

{

new SendFeedbackWindow(selectedhistoryOrderID).ShowDialog();

}

private void SearchHistory\_tb\_TextChanged(object sender, TextChangedEventArgs e)

{

updateHistory();

}

void updateFiles()

{

FilesClient\_dg.Columns.Clear();

DataTable dt = new DataTable();

if (!File.Exists("CreatedFiles.txt")) return;

string temp = File.ReadAllText("CreatedFiles.txt");

string[] temp2 = temp.Split('\n');

dt.Columns.Add("Номер документа");

dt.Columns.Add("Дата создания документа");

for (int i = 1; i < temp2.Length-1; i++)

dt.Rows.Add(i,temp2[i]);

FilesClient\_dg.ItemsSource = dt.DefaultView;

}

}

}

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Data;

using System.Linq;

using System.Threading.Tasks;

using System.Windows;

namespace DataSetClass\_Taxiapp

{

/// <summary>

/// Логика взаимодействия для App.xaml

/// </summary>

public partial class App : Application

{

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace DataSetClass\_Taxiapp

{

/// <summary>

/// Логика взаимодействия для messageWindow.xaml

/// </summary>

public partial class messageWindow : Window

{

public Boolean ex;

public messageWindow(string messagetext = "Вы действительно хотите выйти?")

{

InitializeComponent();

msg\_tb.Text = messagetext;

var temp = App.Current.MainWindow.Background;

if(temp != null) Background = temp;

else

{

foreach (Window w in App.Current.Windows)

{

temp = w.Background;

if (temp != null) Background = temp;

}

}

}

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

{

ex = true;

this.Close();

}

private void Button\_Click\_1(object sender, RoutedEventArgs e)

{

ex = false;

this.Close();

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace DataSetClass\_Taxiapp

{

/// <summary>

/// Логика взаимодействия для DriverWindow.xaml

/// </summary>

public partial class DriverWindow : Window

{

Boolean quest = false;

DataTable Driver;

SQL session = MainWindow.session;

string DriverID;

int selectedDriverOrderID, newOrderID;

public DriverWindow(DataTable Driver)

{

InitializeComponent();

Background = App.Current.MainWindow.Background;

this.Driver = Driver;

}

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

{

try

{

updateHistory();

newOrderUpdate();

}

catch (Exception ex)

{

MainWindow.exMessage(ex);

}

}

public void updateHistory()

{

HistoryDriver\_dg.Columns.Clear();

DriverID = Driver.Rows[0][0].ToString();

HistoryDriver\_dg.ItemsSource = session.selectTable($@"select [ID\_Order] as 'ID', [Order\_Number] as '№ Заказа',[Order\_Date] as 'Дата заказа',[Order\_Time] as 'Время заказа', [Order\_Status] as 'Статус заказа' from [dbo].[Order] where[Driver\_ID] = '{DriverID}' AND [Order\_Number] like '%{SearchHistory\_tb.Text}%'").DefaultView;

}

private void closingRequest(object sender, System.ComponentModel.CancelEventArgs e)

{

if (quest) return;

messageWindow msgwindow = new messageWindow();

msgwindow.ShowDialog();

if (msgwindow.ex == true)

{

e.Cancel = false;

Application.Current.Shutdown();

}

else e.Cancel = true;

}

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

{

new MainWindow().Show();

quest = true;

this.Close();

}

private void dataGridBugFix(object sender, DataGridAutoGeneratingColumnEventArgs e)

{

PropertyDescriptor propertyDescriptor = (PropertyDescriptor)e.PropertyDescriptor;

e.Column.Header = propertyDescriptor.DisplayName;

if (propertyDescriptor.DisplayName == "ID") e.Column.Visibility = Visibility.Hidden;

if (e.PropertyType == typeof(DateTime)) (e.Column as DataGridTextColumn).Binding.StringFormat = "dd/MM/yyyy";

}

private void HistoryDriver\_dg\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

if (!(HistoryDriver\_dg.SelectedItem is DataRowView)) return;

DataRowView selected = HistoryDriver\_dg.SelectedItem as DataRowView;

selectedDriverOrderID = (int)selected[0];

}

public void newOrderUpdate()

{

newOrder\_dg.Columns.Clear();

newOrder\_dg.ItemsSource = session.selectTable($@"select ID\_Order as 'ID', [Order\_Date] as 'Дата заказа',[Order\_Time] as 'Время заказа',[Order\_Number] as 'Номер заказа',[Order\_Status] as 'Статус заказа' from [dbo].[Order] inner join [dbo].[Category] on [Category\_ID] = [ID\_Category] inner join [dbo].[Client] on [Client\_ID] = [ID\_Client] inner join [dbo].[Driver] on [Driver\_ID] = [ID\_Driver] where [Order\_Status] not like 'Закрыт'").DefaultView;

}

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

{

session.action($"update [dbo].[Order] set [Order\_Status] = 'Закрыт', [Driver\_ID] = {DriverID} where [ID\_Order] = '{newOrderID}'");

newOrderUpdate();

updateHistory();

}

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

{

new SendFeedbackWindow(selectedDriverOrderID).ShowDialog();

}

private void SearchHistory\_tb\_TextChanged(object sender, TextChangedEventArgs e)

{

updateHistory();

}

private void newOrder\_dg\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

if (!(newOrder\_dg.SelectedItem is DataRowView)) return;

DataRowView selected = newOrder\_dg.SelectedItem as DataRowView;

newOrderID = (int)selected[0];

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net;

using System.Net.Mail;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace DataSetClass\_Taxiapp

{

/// <summary>

/// Логика взаимодействия для SendMailWindow.xaml

/// </summary>

public partial class SendMailWindow : Window

{

Boolean quest = false;

public int authcode;

string name = "";

public SendMailWindow(string name)

{

InitializeComponent();

Background = App.Current.MainWindow.Background;

this.name = name;

}

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

{

quest = true;

if (authcode.ToString() == authCode\_tb.Text) DialogResult = true;

else ModernWpf.MessageBox.Show("Код не верный");

}

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

{

Thread thread = new Thread(() =>

{

authcode = 0;

//authcode = spamSend(name, "rutopruter@gmail.com");

});

thread.Start();

}

public int spamSend(string name, string sendMail)

{

int authcode = new Random().Next(100000000, 999999999);

try

{

MailAddress From = new MailAddress("taxipraktika@gmail.com", "Taxi");

MailAddress To = new MailAddress(sendMail);

MailMessage msg = new MailMessage(From, To);

msg.Subject = "Код авторизации";

msg.Body = $"{name}, ваш код авторизации в приложении: {authcode}";

SmtpClient smtp = new SmtpClient();

smtp.Host = "smtp.gmail.com";

smtp.Port = 587;

smtp.EnableSsl = true;

smtp.DeliveryMethod = SmtpDeliveryMethod.Network;

smtp.UseDefaultCredentials = false;

smtp.Credentials = new NetworkCredential(From.Address, "qwe123$$");

smtp.Send(msg);

}catch (Exception ex)

{

MainWindow.LogExcep(ex);

}

return authcode;

}

private void Window\_Closing(object sender, System.ComponentModel.CancelEventArgs e)

{

if (quest) return;

messageWindow msgwindow = new messageWindow("Отменить авторизацию?");

msgwindow.ShowDialog();

if (msgwindow.ex == true)

{

e.Cancel = false;

DialogResult = false;

}

else e.Cancel = true;

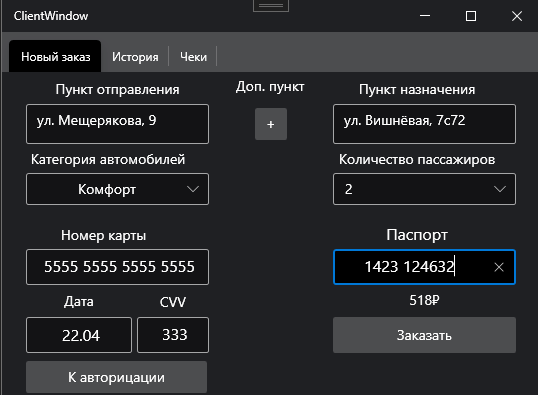
}

}

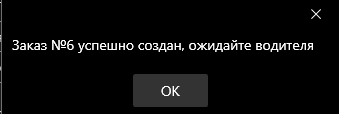
}

1. РЕЗУЛЬТАТ РАБОТЫ
   1. Описание функций приложения базы данных

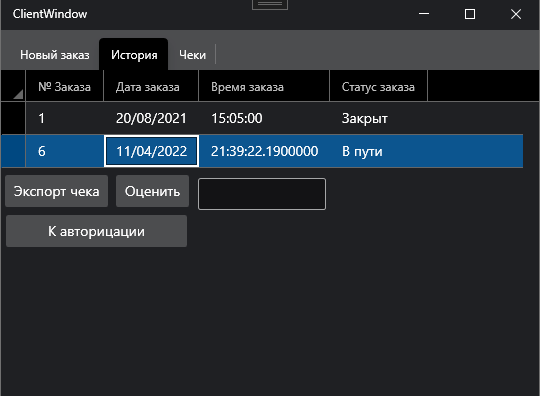
Для заказа клиенту нужно ввести в пункт назначения адрес назначения и в пункт отправления адрес отправления, выбрать категорию (Эконом, комфорт или бизнец), выбрать количество пассажиров, ввести номер и серию паспорта, номер карты, срок действия карты, и код безопасности карты. После нажать кнопку «Заказать».



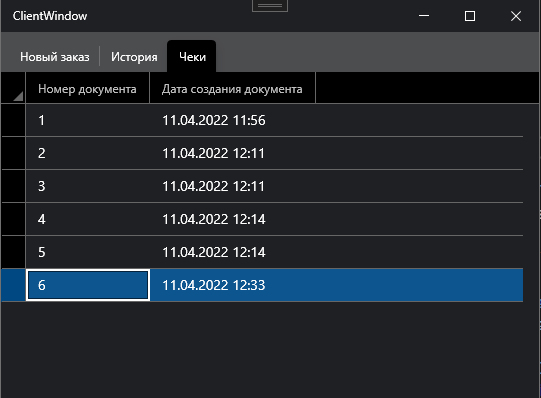
После на экране появится сообщение, что заказ создан и появится в истории заказов.



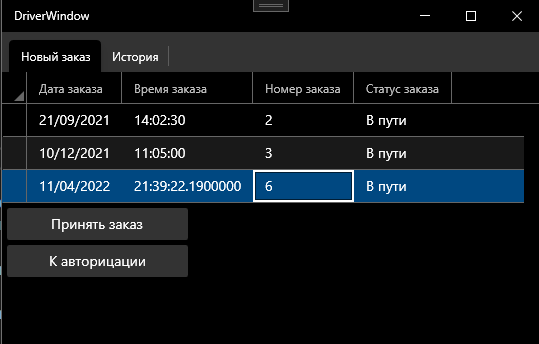
В истории заказов можно нажать на кнопку «Экспорт чека», что загрузит чек в виде Word документа в указанную директорию, также можно оценить заказ нажав кнопку «Оценить». Также историю заказа можно сортировать, введя в текстовое поле возле кнопки «Оценить» своё значение.



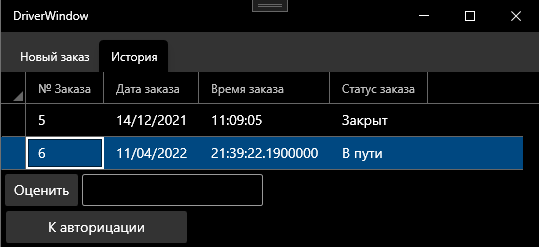
Во вкладке «Чеки» можно увидеть все распечатанные чеки пользователя.



Водитель может принять свободный заказ клиента нажав кнопку «Принять заказ», тогда он отобразиться в истории водителя.



Водитель может также оставить отзыв о заказе нажав на заказ и кнопку «Оценить» во вкладе «История». Также историю заказа можно сортировать, введя в текстовое поле возле кнопки «Оценить» своё значение.



* 1. Описание дополнительных функций

В данном разделе, необходимо указать иллюстрации дополнительных функций: валидация данных, сообщения об ошибках, анимация, работа с реестром, конфигурация и настройка приложения и так далее. (Перед каждой иллюстрацией указывается краткое описание действий пользователя или программы).

1. ВЫВОД

Было сделано приложение по предметной области «Службы такси».

Создана структура Базы данных. Создан класс «SQL» для работы с базой данных. Созданы функции добавления, изменения, удаления, сортировки и фильтрации базы данных.

Созданы методы для создания, считывания и шифрования записей в реестре.

Были получены навыки в разработке баз данных. Валидацией двухфакторной авторизации пользователей через генерацию и отправления случайного кода на указанный электронный адрес. Получения информации из системных датчиков. Разработка системы авторизации и регистрации пользователей.

Сильные стороны проекта состоят в разработке двух отдельных тем, полностью изменяющих всю цветовую палитру приложения. Проверке авторизации пользователя через случайно сгенерированный код, отправленный на электронную почту, в многопоточном режиме работы приложения. Выводе всех критических ошибок на экран и записи их в текстовый документ. В автоматическом перерасчёте всех числовых значений и мгновенно отображении их на экране.

Слабые стороны проекта состоят в отсутствии экспорта и импорта данных из Excel таблиц.

В проекте необходимо доработать экспорт и импорт в Excel таблицы, разработать большее количество тем приложения со сменой цветовой палитры через 3 ползунка RGB.

1. СПИСОК ИСПОЛЬЗУЕМЫХ МАТЕРИАЛОВ
2. ModernWpf, обновлённый интерфейс для приложения, <https://github.com/Kinnara/ModernWpf>.
3. LiveCharts, визуализация данных в виде графиков, <https://lvcharts.net>.
4. Microsoft.Office.Interop.Word, библиотека для создания документов Microsoft Word, <https://docs.microsoft.com/ru-ru/dotnet/api/microsoft.office.interop.word>.
5. ControlzEx, создание масок ввода для кнопок, <https://github.com/ControlzEx/ControlzEx>.
6. ModernWpf.MessageBox, улучшенный вид для всплывающего окна, <https://github.com/OpenByteDev/ModernWpf.MessageBox>.
7. Microsoft.Xaml.Behaviors.Wpf, расширенные настройки для визуальных элементов, <https://github.com/Microsoft/XamlBehaviorsWpf>.
8. Microsoft.Windows.SDK.Contracts, пакет, который добавляет поддержку старых API в приложение, <https://www.nuget.org/packages/Microsoft.Windows.SDK.Contracts>.
9. System.Net.Mail, класс, для отправки кода авторизации на указанную почту, <https://docs.microsoft.com/en-us/dotnet/api/system.net.mail?view=net-6.0>.
10. Microsoft.VisualBasic, использовался ComputerInfo Class, который выводит информацию с системных датчиков, <https://docs.microsoft.com/en-us/dotnet/api/microsoft.visualbasic.devices.computerinfo?view=windowsdesktop-6.0>.
11. System.Threading.Thread, класс, позволяющий запустить параллельный поток в приложении, <https://docs.microsoft.com/en-us/dotnet/api/system.threading.thread?view=net-6.0>.
12. Inno Setup, система создания инсталляторов для Windows-программ с открытым исходным кодом, <https://jrsoftware.org>.
13. Type.GetTypeFromProgID, метод, определяющий установлен Word и Excel или нет, <https://docs.microsoft.com/ru-ru/dotnet/api/system.type.gettypefromprogid?view=net-6.0>.
14. Random, генератор случайных чисел для кода двухэтапной авторизации, <https://docs.microsoft.com/ru-ru/dotnet/api/system.random?view=net-6.0>.
15. SHA-1, используемый метод шифрования данных, <https://upread.ru/art.php?id=42>.
16. System.Diagnostics.PerformanceCounter, класс, позволяющий получить информацию из системы об объёме и свободном количестве оперативной памяти, <https://docs.microsoft.com/ru-ru/dotnet/api/system.diagnostics.performancecounter?view=dotnet-plat-ext-6.0>.
17. CheckInternetSpeed, метод, определяющий скорость интернета, <https://stackoverflow.com/questions/13600604/how-to-get-accurate-download-upload-speed-in-c-net>.
18. System.Windows.Threading.DispatcherTimer, класс, позволяющий установить таймер, который будет с определённым периодом вызывать определённый код, <https://docs.microsoft.com/ru-ru/dotnet/api/system.windows.threading.dispatchertimer?view=windowsdesktop-6.0>.
19. System.Data.SqlClient, класс для работы со средой MS SQL, <https://docs.microsoft.com/en-us/dotnet/api/system.data.sqlclient?view=dotnet-plat-ext-6.0>.
20. Шаблон отчёта УП 01.01, шаблон, предоставленный Иваном Максимовичем Щаниковым, <https://classroom.google.com/u/1/c/NDc3MjIxNjIzMDMw/m/NDc3MjIxNjIzMDQy>.
21. C# DataSet, пример готовой работы, предоставленный Иваном Максимовичем Щаниковым, https://classroom.google.com/u/1/c/NDc3MjIxNjIzMDMw/m/NDc3MjIxNjIzMDUw.