**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ**

**РОССИЙСКОЙ ФЕДЕРАЦИИ**

федеральное государственное бюджетное образовательное учреждение высшего образования «Казанский национальный исследовательский

технический университет им. А.Н. Туполева-КАИ»

(КНИТУ-КАИ)

Институт компьютерных технологий и защиты информации

(наименование института(факультета), филиала)

Кафедра автоматизированных систем обработки информации и управления

(наименование кафедры)

09.03.02 Информационные системы и технологии

(шифр и наименование направления подготовки (специальности))

Курсовая работа

по дисциплине: Технологии программирования

на тему: Панель управления грузоперевозками в .NET Remoting и в ASP.NET.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Обучающийся: | 4307 |  |  |  | Сайфуллин Д.А. |
|  | (номер группы) |  | (подпись, дата) |  | (Ф.И.О.) |

|  |  |  |  |
| --- | --- | --- | --- |
| Руководитель: | старший преподаватель |  | Вафин Р. Р. |
|  | (должность) |  | (Ф.И.О.) |

|  |  |  |  |
| --- | --- | --- | --- |
| Курсовая работа зачтена с оценкой: |  |  |  |
|  |  |  | (подпись, дата) |

Казань 2023

**MINISTRY OF SCIENCE AND HIGHER EDUCATION**

**RUSSIAN FEDERATION**

Federal State Budgetary Educational Institution of Higher Education

"Kazan National Research Technical University A.N. Tupolev-KAI"

(KNRTU-KAI)

Institute of Computer Technology and Information Security

(name of institute (faculty), branch)

Department of Automated Processing and Control Systems

(name of the department)

09.03.02 Information Systems and Technologies

(code and name of the field of study (specialty))

COURSE WORK

by discipline: Programming techniques

on the topic: Cargo management dashboard in .NET Remoting and in ASP.NET.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Student: | 4307 |  |  |  | Sayfullin D.A. |
|  | (group number) |  | (signature, date) |  | (full name) |

|  |  |  |  |
| --- | --- | --- | --- |
| Supervisor: | senior lecturer |  | Vafin R. R. |
|  | (position) |  | (full name) |

|  |  |  |  |
| --- | --- | --- | --- |
| Coursework credited with assessment: |  |  |  |
|  |  |  | (signature, date) |

Kazan 2023

Данная пояснительная записка к курсовой работе по дисциплине «Технологии программирования» на тему «Панель управления грузоперевозками в .NET Remoting и в ASP.NET».

В результате выполнения курсовой работы были реализованы две программы:

1. Клиент-серверное приложение с использованием средств .NET Remoting и ADO.NET для ОС Windows.

2. Web-приложение с использованием фреймворка ASP.NET и используя технологию доступа к данным ADO.NET.

В процессе проектирования этих программ были подробно изучены технологии .NET Remoting, ASP.NET и ADO.NET.

This explanatory note to the course work on the discipline "Programming Technologies" on the topic “Cargo management dashboard in .NET Remoting and in ASP.NET.”

As a result of the course work, two programs were implemented:

1. Client-server application using .NET Remoting and ADO.NET tools for Windows OS.

2. Web application using the ASP.NET framework and using the ADO.NET data access technology.

During the design of these programs, .NET Remoting, ASP.NET, and ADO.NET technologies were studied in detail.

Содержание

[Введение 6](#_Toc133184120)

[Introduction 7](#_Toc133184121)

[1 Постановка задачи 8](#_Toc133184122)

[2 Описание решения задачи 8](#_Toc133184123)

[2.1. Общий алгоритм решения задачи 8](#_Toc133184124)

[2.2. Описание применяемых для решения задачи технологий 10](#_Toc133184125)

[3 Спецификация программы 11](#_Toc133184126)

[3.1. Инструкция по работе с приложением (.NET Remoting) 11](#_Toc133184127)

[3.2. Внутреннее представление программы (.NET Remoting) 13](#_Toc133184128)

[3.3. Инструкция по работе с приложением (ASP .NET) 15](#_Toc133184129)

[3.4. Внутреннее представление программы (ASP .NET) 16](#_Toc133184130)

[Заключение 17](#_Toc133184131)

[Список литературы 18](#_Toc133184132)

[References 19](#_Toc133184133)

[Приложение 1. Код реализации серверной части 20](#_Toc133184134)

[Приложение 2. Код реализации клиентской части 21](#_Toc133184135)

[Приложение 3. Код реализации удаленных объектов 34](#_Toc133184136)

[Приложение 4. Код реализации серверной части (ASP.NET) 36](#_Toc133184137)

# Введение

В работе системы для панели управления грузоперевозками используются два типа пользователей – логист и водитель. Каждый из них авторизуется в систему под своим уникальным именем (Login) и далее выполняет определенные действия.

Логист, войдя под своим идентификатором, имеет возможность просматривать, редактировать, добавлять и удалять грузы.

Водитель после входа в аккаунт имеет возможность принять груз, предварительно выбрав его из списка. Если груз уже назначен другому водителю или у водителя уже есть назначенный груз, то он будет проинформирован об этом. Также водитель может видеть свой текущий груз, отменить или завершить доставку.

**Introduction**

The system for the cargo transportation control panel involves two types of users - a logistics manager and a driver. Each user logs into the system using their unique login credentials and performs specific actions.

The logistics manager, upon logging in with their identifier, has the ability to view, edit, add, and delete cargoes.

The driver, after logging into their account, can accept a cargo by selecting it from the list. If the cargo is already assigned to another driver or if the driver already has a cargo assigned, they will be notified. The driver can also view their current cargo, decline it, or complete the delivery.

1. **Постановка задачи**

Разработать клиент-серверное и веб приложения для интернет-магазина канцелярии с использованием средств .NET Remoting для ОС Windows и фреймворка ASP.NET.

При запуске программы открывается стандартная веб-форма с пользовательским интерфейсом. При авторизации в приложении под именем логиста доступны просмотр, редактирование, добавление и удаление грузов, а под именем водителя – просмотр текущего груза, принятие нового груза, отказ от него или завершение доставки.

1. **Описание решения задачи**
   1. **Общий алгоритм решения задачи**

Для решения поставленной задачи используются технологии клиент-серверного взаимодействия .NET Remoting, а также технология для работы с базой данных – ADO.NET.

Последовательность действий для решения задачи (.NET Remoting):

1. Запускаем сервер;
2. Регистрируем каналы на сервере и привязываем их к удаленным объектам;
3. Запускаем клиент;
4. Проверяем подключение к серверу по каналам;
5. Вызываем методы удаленного объекта через интерфейс;
6. Записываем/получаем данные;
7. Отображаем данные клиенту.

Последовательность действий для решения задачи (ASP.NET):

1. Запускаем сервер;
2. Отображаем пользовательский интерфейс в окне браузера;
3. Делаем запрос на сервер;
4. Записываем/получаем данные;
5. Отображаем данные клиенту.

Схемы реализации, поставленной задачи для технологий представлены на рисунке 1 и рисунке 2.

**Сервер**

БД

Удаленный объект 1

Удаленный объект 2

Интерфейс

Клиент

Рис. 1 – Схема реализации, поставленной задачи для технологии .NET Remoting

Веб-приложение

БД

Клиент

Интернет

Рис. 2 – Схема реализации, поставленной задачи для технологии ASP.NET

* 1. **Описание применяемых для решения задачи технологий**

Remoting − технология взаимодействия между разными доменами приложений. NET Remoting применяется в организации взаимодействия доменов приложений внутри одного процесса, между процессами одной системы или же между процессами на разных системах.

В отличие от Web-службы XML, NET Remoting гарантирует сильную связность клиента и сервера, потому что они разделяют одни и те же типы объектов.

Удаленные сборки могут быть сконфигурированы для локальной работы в пределах одного домена приложений или как часть удаленного приложения.

Среда Remoting − универсальное средство доступа к удаленным объектам, которое может быть приспособлено к широкому классу задач взаимодействия компонент распределенного приложения.

ASP.NET − технология разработки Web-приложений. Web-приложения — это особый тип программ, которые построены по клиент-серверной архитектуре. Особенность технологии состоит в том, что Web-приложение находится и выполняется на сервере, а клиент получает только результаты работы.

Приложение обрабатывает полученные запросы от пользователя (клиента) и выдает результат.

Запросы и результаты их обработки передаются через Интернет.

1. **Спецификация программы**
   1. **Инструкция по работе с приложением (.NET Remoting)**

Для начала работы с приложение необходимо запустить Server.exe, регистрирующий каналы связи и привязывающий удаленные объекты. Сообщения об успешном запуске сервера и регистрации каналов показаны на рисунке 3.

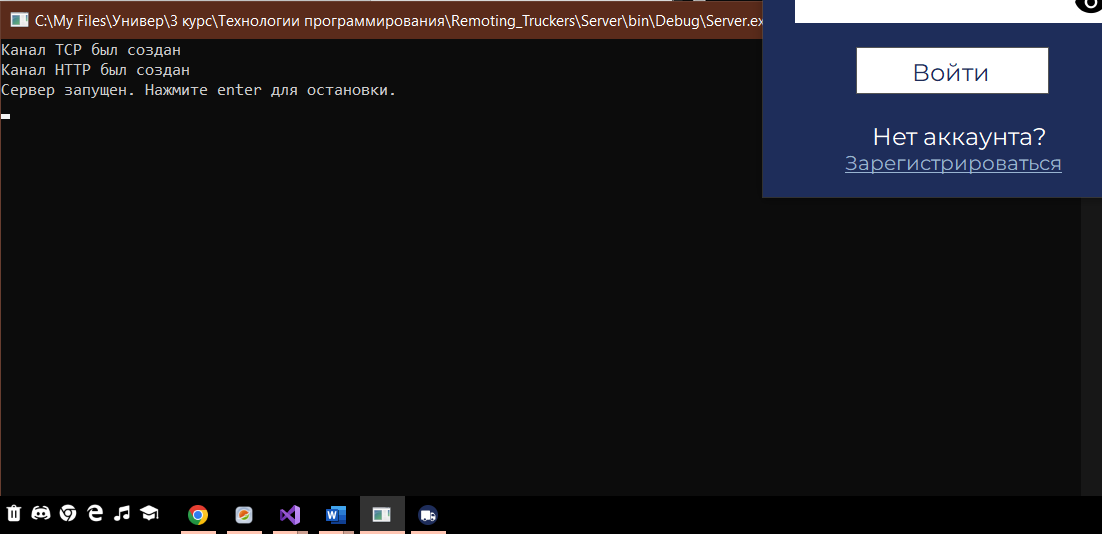


Рис. 3 – Результат запуска сервера

Затем запускается Truckers.exe, в результате чего пользователь видит окно авторизации, изображенное на рисунке 4, на котором размещаются следующие элементы:

1. поля авторизации: логин и пароль;
2. кнопка «Войти»;
3. кнопка «Зарегистрироваться»

При нажатии на кнопку «Зарегистрироваться» открывается окно регистрации, изображенное на рисунке 5, на котором размещаются следующие элементы:

1. поля: имя, логин, пароль и должность (логист или водитель);
2. кнопка «Отправить»;
3. кнопка «Авторизоваться»

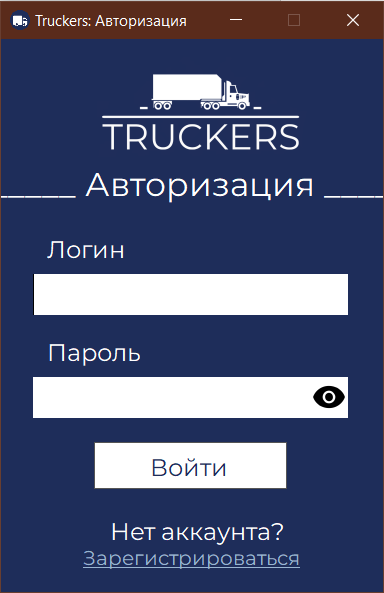


Рис. 4 – Запуск клиента. Окно авторизации

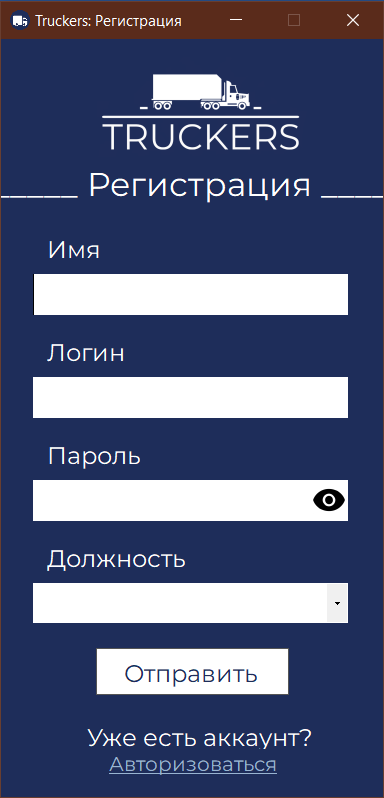


Рис. 5 – Окно регистрации

После ввода логина и пароля в окне авторизации и нажатия кнопки «Войти» открывается окно управления грузами, зависящее от должности. Окна изображены на рисунках 6-7. В окне логиста имеются следующие элементы:

1. кнопка «Выйти из системы»;
2. текст с именем логиста;
3. выпадающий список ID грузов;
4. поле ввода ID водителя;
5. выпадающий список статуса груза;
6. поле ввода наименования груза;
7. поле ввода массы груза;
8. поле ввода пункта отправления;
9. поле ввода пункта назначения;
10. кнопка «Обновить»;
11. кнопка «Сохранить»;
12. кнопка «Добавить»;
13. кнопка «Удалить».

В окне логиста водителя следующие элементы:

1. кнопка «Выйти из системы»;
2. текст с именем водителя и его ID;
3. выпадающий список ID грузов;
4. поле просмотра ID водителя;
5. поле просмотра статуса груза;
6. поле просмотра наименования груза;
7. поле просмотра массы груза;
8. поле просмотра пункта отправления;
9. поле просмотра пункта назначения;
10. кнопка «Текущий груз»;
11. кнопка «Принять»;
12. кнопка «Отказаться»;
13. кнопка «Доставить».

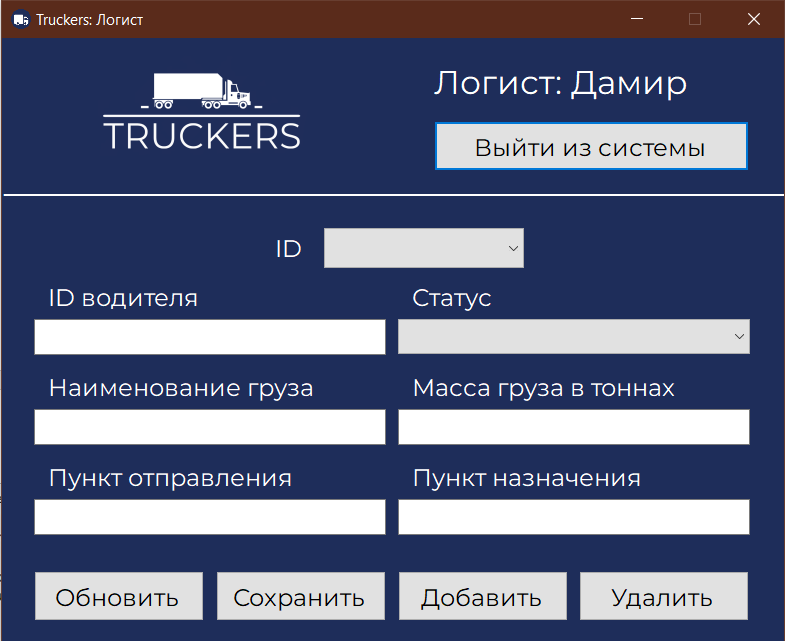


Рис. 6 – Окно логиста

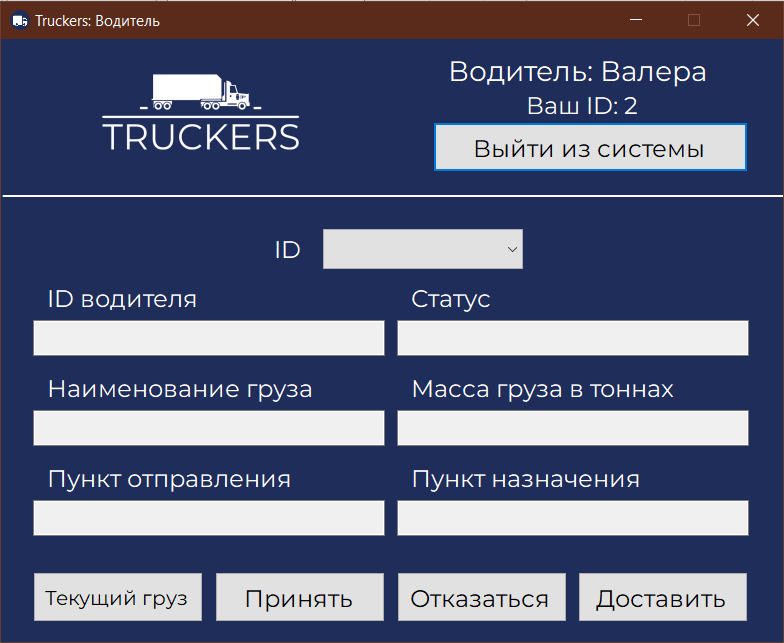


Рис. 7 – Окно водителя

При входе в систему у логиста появляются дополнительные возможности – просмотр имеющихся грузов, их редактирование, добавление и удаление. У водителя – просмотр текущего груза, принятие, отказ и доставка грузов.

* 1. **Внутреннее представление программы (.NET Remoting)**

**Сервер (листинг программы представлен в Приложении 1)**

В сервере происходит регистрация двух каналов – через код и конфигурационный файл.

Метод static void ServerTCP() реализовывает программную регистрацию первого канала через протокол TCP и привязывает к нему удалённый объект RemoteObjectTCP. Настройка реализована в коде.

Регистрация второго канала и привязка к нему удалённого объекта RemoteObjectHTTP осуществляется в конфигурационном файле через протокол HTTP.

Подключение конфигурационного файла происходит c помощью строки RemotingConfiguration.Configure("ServerConfig.config", true).

Строка Console.ReadLine() позволяет серверу находиться в рабочем состоянии.

**Клиент (листинг программы представлен в Приложении 2)**

Клиент представляет собой форму, реализующую пользовательский интерфейс. В нем происходит регистрация каналов для работы с удалёнными объектами, через которые производится запись или чтение данных из базы данных. Спонсор для продления времени аренды удалённого объекта также реализован в клиенте.

Метод private void ConnectToServer() осуществляет канала TCP через код.

Метод private void button1\_Click(object sender, EventArgs e) является обработчиком нажатий на кнопку Войти и выполняет авторизацию пользователя в системе по логину и паролю, введенными из полей Логин и Пароль. Если же пары логин/пароль нет в базе данных, выводится сообщение «Неверный логин или пароль!». Если введенные данные верны, запускается форма, в зависимости от должности.

Метод private void button2\_Click(object sender, EventArgs e) является обработчиком нажатий на кнопку скрытия/показа пароля. При нажатии введенный текст в поле пароля показывается, либо скрывается.

Событие private void linkLabel1\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e), которое срабатывает на нажатие кнопки «Зарегистрироваться». Открывает окно регистрации.

Метод private void CargoReload() обновляет таблицу с базы данных и добавляет значения ID груза в выпадающий список.

Событие private void comboBox\_ID\_SelectedValueChanged(object sender, EventArgs e) срабатывает при выборе из выпадающего списка ID груза. Заполняет остальные поля данными в соответствии с ID груза.

Событие private void buttonExit\_Click(object sender, EventArgs e) запускается при нажатии на кнопку «Выйти из системы». Закрывает текущее окно и открывает окно авторизации.

Методы private void buttonReload\_Click(object sender, EventArgs e), private void buttonSave\_Click(object sender, EventArgs e), private void buttonAdd\_Click(object sender, EventArgs e), private void buttonDelete\_Click(object sender, EventArgs e) вызываются при нажатии кнопок «Обновить», «Сохранить», «Добавить» и «Удалить» соответственно в окне логиста.

Методы private void buttonCurrent\_Click(object sender, EventArgs e), private void buttonAccept\_Click(object sender, EventArgs e), private void buttonCancel\_Click(object sender, EventArgs e), private void buttonDelivery\_Click(object sender, EventArgs e) вызываются при нажатии кнопок «Текущий груз», «Принять», «Отказаться» и «Доставить» соответственно в окне водителя.

* 1. **Инструкция по работе с приложением (ASP .NET)**

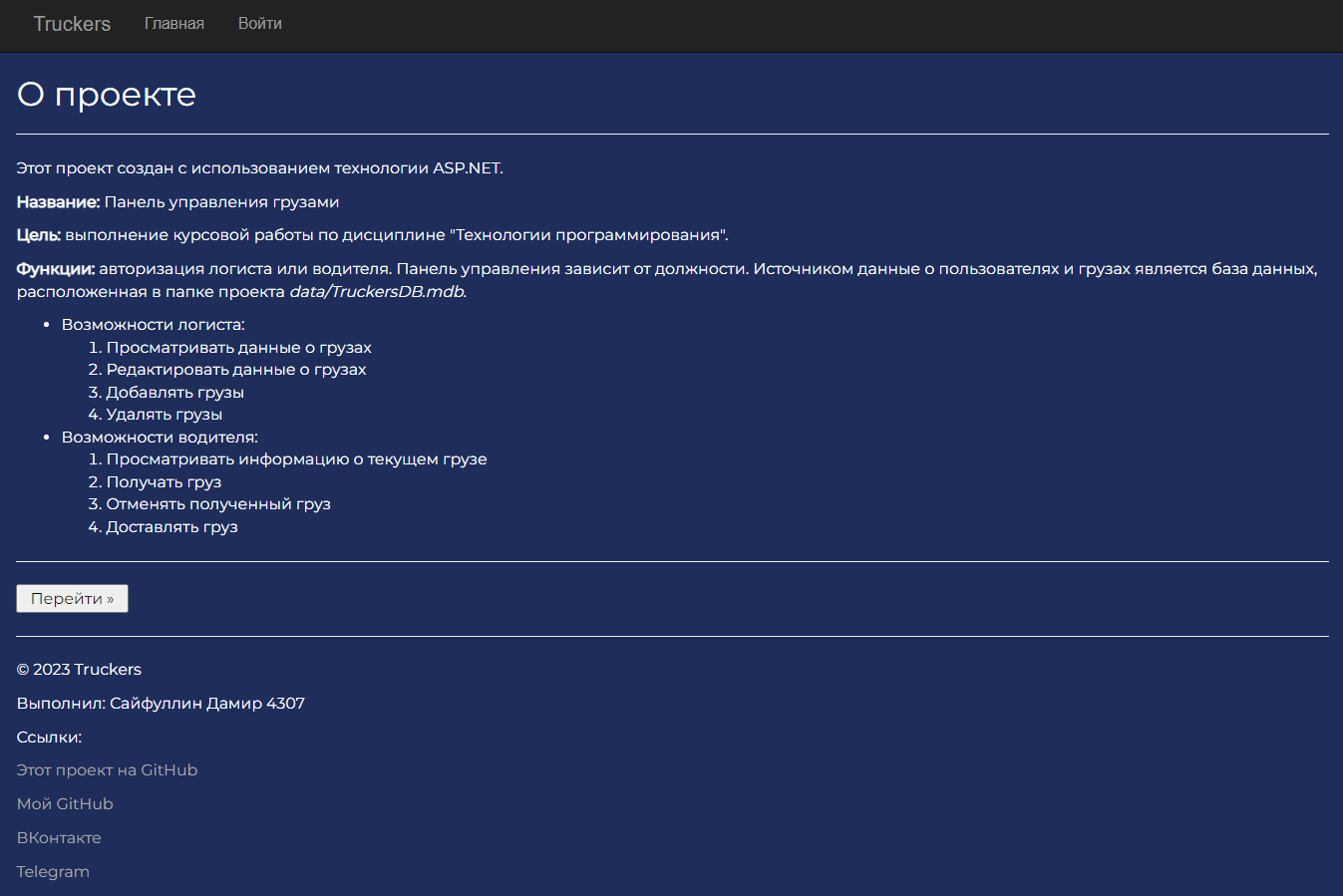
При запуске приложения открывается стартовое окно – рисунок 8. 

Рис. 8 – Стартовый экран при запуске приложения

При нажатии кнопки «Перейти» на начальной странице или «Войти» на шапке сайта открывается страница авторизации – рисунок 9.

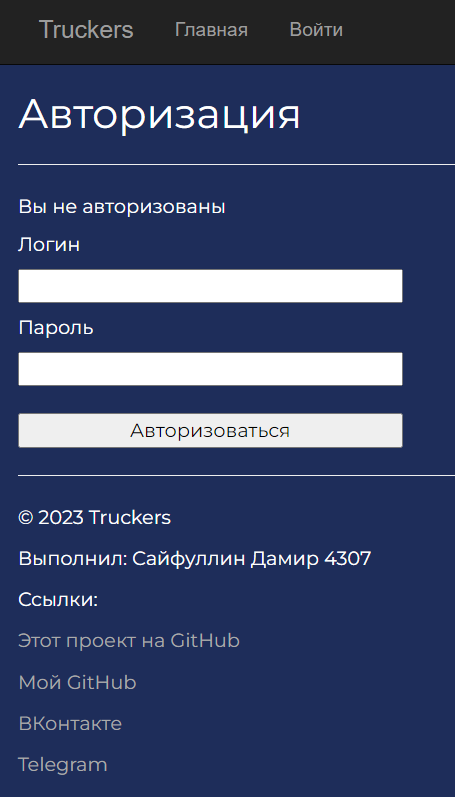


Рис. 9 – Страница авторизации

При входе в систему появляются дополнительные возможности в зависимости от должности. У логиста просмотр имеющихся грузов, их редактирование, добавление и удаление – рисунок 10. У водителя – просмотр текущего груза, принятие, отказ и доставка грузов.

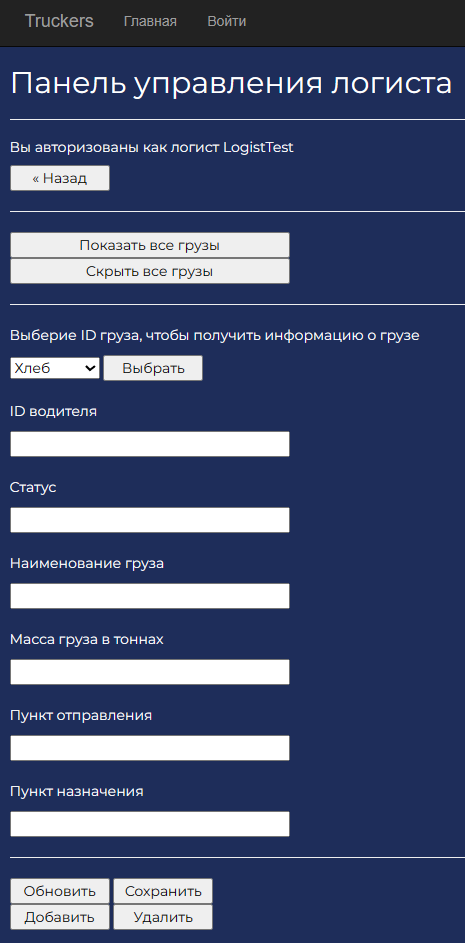


Рис. 10 – Экран при входе в систему логиста

* 1. **Внутреннее представление программы (ASP .NET)**

Программа на технологии ASP.NET состоит только из сервера. Клиентом будет являться окно браузера (исходный код в Приложении 4). Опишем внутреннее представление сервера.

Logist.aspx.cs:

* метод protected void Page\_Load(object sender, EventArgs e): он вызывается при полной загрузке страницы.
* метод public void CargoID\_Reload(): обновляет список ID грузов
* событие public void buttonChoose\_Click(object sender, EventArgs e): оно срабатывает при нажатии кнопки «Выбрать». Используется для показа информации о грузе по ID.
* событие protected void gridshow\_Click(object sender, EventArgs e): срабатывает при нажатии кнопки «Показать все грузы». Используется для показа всех грузов.
* событие protected void gridhide\_Click(object sender, EventArgs e): срабатывает при нажатии кнопки «Скрыть все грузы». Используется для скрытия всех грузов.

Logist.aspx.cs:

* метод protected void Page\_Load(object sender, EventArgs e): он вызывается при полной загрузке страницы.
* событие public void Authorize(object sender, EventArgs e): является обработчиком нажатий на кнопку «Авторизоваться» и выполняет авторизацию пользователя в системе по логину и паролю, введенными из полей Логин и Пароль. Если же пары логин/пароль нет в базе данных, выводится сообщение «Неверный логин или пароль!».
* событие protected void button1\_Click(object sender, EventArgs e): срабатывает при нажатии кнопки «Открыть панель управления».
* событие protected void button2\_Click(object sender, EventArgs e): срабатывает при нажатии кнопки «Сменить пользователя».

**Заключение**

В результате выполнения курсовой работы были реализованы две программы:

1. Клиент-серверное приложение с использованием средств .NET Remoting и ADO.NET для ОС Windows.

2. Web-приложение с использованием фреймворка ASP.NET и используя технологию доступа к данным ADO.NET.

В процессе проектирования этих программ были подробно изучены технологии .NET Remoting, ASP.NET и ADO.NET.

При выполнении работы были изучены и применены принципы поэтапной разработки и отладки программ средней сложности. Получены знания по разработке собственных классов, созданы из них разнообразные объекты и применены совместно для реализации требуемых алгоритмов функционирования программы. Также были использованы необходимые классы пакетов C#. Разработан собственный интерфейс программы, применены необходимые интерфейсные элементы на базе пакетов C#. Получены умения по описанию разработанного программного продукта как с точки зрения его пользователя, отражая его внешнюю сторону, так и с точки зрения программиста, отражая особенности его реализации.

**Список литературы**

1. Хабибуллин И.Ш. Самоучитель C#. – СПб.: БХВ-Петербург, 2008.
2. Медведев В.И. .NET компонентно-ориентированное программирование – Казань: РИЦ «Школа», 2013. – 248 c.: ил. – (Серия «Современная прикладная математика и информатика»).
3. П. Наутон, Г. Шилдт. C# 2. Наиболее полное руководство в подлиннике. – СПб.: БХВ-Петербург, 2005. – 1072 с.: ил.
4. <https://habr.com/ru/hub/csharp/>- форум C# программистов.
5. Эндрю Стелман Head first C#, 679с., 2010.

**References**

1. Habibullin I.Sh. Teach Yourself C#. - St. Petersburg.: BHV-Petersburg, 2008.
2. Medvedev V.I .NET Component-oriented programming - Kazan: RIP "School", 2013. - 248 c .: silt. - (Series "Modern Applied Mathematics and Computer Science").
3. P. Nauton G. Shildt. C# . The most comprehensive guide to podlinnike.- SPb .: BHV-Petersburg, 2005. - 1072 p .: silt.
4. <https://habr.com/ru/hub/csharp/>- forum of C# programmers.
5. Andrew Stellman Head first C#, 679p., 2010.

# Приложение 1. Код реализации серверной части

Server.cs

using System;

using System.Collections;

using System.Runtime.Remoting;

using System.Runtime.Remoting.Channels;

using System.Runtime.Remoting.Channels.Tcp;

using System.Runtime.Serialization.Formatters;

namespace Server

{

class Server

{

static void Main(string[] args)

{

ServerTCP();

ServerHTTP();

Console.WriteLine("Сервер запущен. Нажмите enter для остановки.");

Console.ReadLine();

}

static void ServerTCP()

{

BinaryServerFormatterSinkProvider serverProv = new BinaryServerFormatterSinkProvider();

serverProv.TypeFilterLevel = TypeFilterLevel.Full;

IDictionary props = new Hashtable();

props["port"] = 8080;

props["name"] = "TCPChannel";

props["typeFilterLevel"] = TypeFilterLevel.Full;

props["impersonate"] = false;

props["secure"] = true;

props["protectionLevel"] = System.Net.Security.ProtectionLevel.EncryptAndSign;

TcpChannel tcpChannel = new TcpChannel(props, null, serverProv);

ChannelServices.RegisterChannel(tcpChannel, true);

RemotingConfiguration.ApplicationName = "tcp";

RemotingConfiguration.RegisterActivatedServiceType(typeof(RemoteObjectTCP));

Console.WriteLine("Канал TCP был создан");

}

static void ServerHTTP()

{

RemotingConfiguration.Configure("C:/My Files/Универ/3 курс/Технологии программирования/Remoting\_Truckers/Server/ServerConfig.config", true);

Console.WriteLine("Канал HTTP был создан");

}

}

}

**Приложение 2.** **Код реализации клиентской части**

FormLogin.cs

using System;

using System.Collections.Generic;

using System.Runtime.Remoting.Channels;

using System.Runtime.Remoting.Channels.Http;

using System.Runtime.Remoting.Lifetime;

using System.Runtime.Serialization.Formatters;

using System.Windows.Forms;

using Server;

namespace Truckers

{

public partial class FormLogin : Form

{

RemoteObjectHTTP remoteHTTP; // удаленный объект

HttpChannel channel = new HttpChannel(new Dictionary<string, string> { { "port", "0" } }, new BinaryClientFormatterSinkProvider(), new BinaryServerFormatterSinkProvider { TypeFilterLevel = TypeFilterLevel.Full });

bool password\_show = false;

public FormLogin()

{

InitializeComponent();

ConnectToServer();

this.TopMost = true;

}

private void ConnectToServer() // установка соединения с сервером

{

ChannelServices.RegisterChannel(channel, false);

remoteHTTP = new RemoteObjectHTTP();

ILease lease = (ILease)remoteHTTP.InitializeLifetimeService();

ClientSponsor clientHTTPSponsor = new ClientSponsor();

lease.Register(clientHTTPSponsor);

}

private void button1\_Click(object sender, EventArgs e) // нажатие на кнопку "Войти"

{

// проверка на пустные поля

if (textBox1.Text == "" || textBox2.Text == "")

{

MessageBox.Show(

"Введите логин и пароль!",

"Ошибка!",

MessageBoxButtons.OK,

MessageBoxIcon.Error,

MessageBoxDefaultButton.Button1,

MessageBoxOptions.DefaultDesktopOnly);

}

else

{

// вызов метода Autorization у удаленного объекта

string result = remoteHTTP.Autorization(textBox1.Text, textBox2.Text);

// если пользователей с таким логином и паролем нет

if (result == "1")

{

MessageBox.Show(

"Неверный логин или пароль!",

"Ошибка!",

MessageBoxButtons.OK,

MessageBoxIcon.Error,

MessageBoxDefaultButton.Button1,

MessageBoxOptions.DefaultDesktopOnly);

}

// если такой пользователь логист

else if (result == "logist")

{

// получение имени пользователя

string username = remoteHTTP.GetUsername(textBox1.Text);

// переход к форме логиста

FormLogist formLogist = new FormLogist(this, username);

formLogist.Show();

this.Hide();

}

// если такой пользователь водитель

else if (result == "driver")

{

// получение имени пользователя

string username = remoteHTTP.GetUsername(textBox1.Text);

int ID = remoteHTTP.GetID(textBox1.Text);

// переход к форме водителя

FormDriver formDriver = new FormDriver(this, username, ID);

formDriver.Show();

this.Hide();

}

}

}

private void button2\_Click(object sender, EventArgs e) // нажатие на кнопку скрытия пароля

{

password\_show = !password\_show;

if (password\_show)

{

button2.BackgroundImage = Properties.Resources.EyeClosed;

textBox2.UseSystemPasswordChar = false;

}

else

{

button2.BackgroundImage = Properties.Resources.EyeOpened;

textBox2.UseSystemPasswordChar = true;

}

}

private void linkLabel1\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e) // нажатие на кнопку "Зарегистрироваться"

{

// открытие формы регистрации

FormRegister formRegister = new FormRegister(this, remoteHTTP);

formRegister.Show();

// скрытие это формы авторизации

this.Hide();

}

}

}

FormRegister.cs

using System;

using System.Collections.Generic;

using System.Runtime.Remoting.Channels;

using System.Runtime.Remoting.Channels.Http;

using System.Runtime.Serialization.Formatters;

using System.Windows.Forms;

using Server;

namespace Truckers

{

public partial class FormRegister : Form

{

HttpChannel channel = new HttpChannel(new Dictionary<string, string> { { "port", "0" } }, new BinaryClientFormatterSinkProvider(), new BinaryServerFormatterSinkProvider { TypeFilterLevel = TypeFilterLevel.Full });

bool password\_show = false; // показывать пароль

RemoteObjectHTTP remoteHTTP; // удаленный объект

FormLogin formLogin; // ссылка на форму авторизации

public FormRegister(FormLogin formLogin, RemoteObjectHTTP remoteHTTP)

{

InitializeComponent();

this.formLogin = formLogin; // присваивание ссылки на форму авторизации

this.remoteHTTP = remoteHTTP;

this.TopMost = true;

}

private void linkLabel1\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e) // нажатие на кнопку "Авторизоваться"

{

// открытие формы авторизации

formLogin.Show();

// закрытие этой формы регистрации

this.Close();

// освобождение памяти

this.Dispose();

}

private void button1\_Click(object sender, EventArgs e) // нажатие на кнопку "Отправить"

{

// проверка на пустые поля

if (textBox1.Text == "" || textBox2.Text == "" || textBox3.Text == "" || comboBox1.Text == "")

{

MessageBox.Show(

"Не все поля заполнены!",

"Ошибка!",

MessageBoxButtons.OK,

MessageBoxIcon.Warning,

MessageBoxDefaultButton.Button1,

MessageBoxOptions.DefaultDesktopOnly);

}

else

{

// вызов метода Registration у удаленного объекта

string result = remoteHTTP.Registration(textBox1.Text, textBox2.Text, textBox3.Text, comboBox1.Text);

// если регистрация успешна

if (result == "0")

{

MessageBox.Show(

"Регистрация прошла успешно!",

"Успех!",

MessageBoxButtons.OK,

MessageBoxIcon.Information,

MessageBoxDefaultButton.Button1,

MessageBoxOptions.DefaultDesktopOnly);

}

// если пользователь с таким логином уже существует

else if (result == "1")

{

MessageBox.Show(

"Пользователь с таким логином уже существует!",

"Ошибка!",

MessageBoxButtons.OK,

MessageBoxIcon.Warning,

MessageBoxDefaultButton.Button1,

MessageBoxOptions.DefaultDesktopOnly);

}

}

}

private void button2\_Click(object sender, EventArgs e) // нажатие на кнопку скрытия пароля

{

password\_show = !password\_show;

if (password\_show)

{

button2.BackgroundImage = Properties.Resources.EyeClosed;

textBox3.UseSystemPasswordChar = false;

}

else

{

button2.BackgroundImage = Properties.Resources.EyeOpened;

textBox3.UseSystemPasswordChar = true;

}

}

}

}

FormDriver.cs

using Server;

using System;

using System.Data;

using System.Runtime.Remoting;

using System.Runtime.Remoting.Lifetime;

using System.Windows.Forms;

namespace Truckers

{

public partial class FormDriver : Form

{

int ID; // ID водителя

RemoteObjectTCP remoteTCP; // удаленный объект

FormLogin formLogin; // ссылка на форму авторизации

DataTable cargoDataTable; // таблица с грузами

public FormDriver(FormLogin formLogin, string username, int ID)

{

InitializeComponent();

this.ID = ID;

this.formLogin = formLogin; // присваивание ссылки на форму авторизации

label1.Text = "Водитель: " + username; // вывод имени пользователя на форму

label2.Text = "Ваш ID: " + ID.ToString(); // вывод ID пользователя на форму

ConnectToServer();

CargoReload();

this.TopMost = true;

}

private void ConnectToServer() // установка соединения с сервером

{

RemotingConfiguration.Configure("C:/My Files/Универ/3 курс/Технологии программирования/Remoting\_Truckers/Truckers/ClientConfig.config", false);

remoteTCP = new RemoteObjectTCP();

ILease lease = (ILease)remoteTCP.InitializeLifetimeService();

ClientSponsor clientTCPSponsor = new ClientSponsor();

lease.Register(clientTCPSponsor);

}

private void CargoReload() // обновление таблицы

{

// получение данных

cargoDataTable = remoteTCP.Logist\_CargoReload();

// очистка выпадающего списка

comboBox\_ID.Items.Clear();

// проходим по каждой строке в cargoDataTable

foreach (DataRow row in cargoDataTable.Rows)

{

// получение значения поля ID из текущей строки

object value = row["ID"];

// добавление значения в comboBox\_ID

comboBox\_ID.Items.Add(value);

}

}

private void comboBox\_ID\_SelectedValueChanged(object sender, EventArgs e) // при выборе ID в выпадающем списке

{

// поиск строк с выбранным ID

DataRow[] IDRow = cargoDataTable.Select(String.Format("ID = '{0}'", comboBox\_ID.SelectedItem.ToString()));

foreach (DataRow row in IDRow)

{

// изменение полей в соответсвии с выбраным ID

textBoxDriverID.Text = row["DriverID"].ToString();

textBoxStatus.Text = row["Status"].ToString();

textBoxCargo.Text = row["Cargo"].ToString();

textBoxWeight.Text = row["Weight"].ToString();

textBoxFrom.Text = row["From"].ToString();

textBoxTo.Text = row["To"].ToString();

}

}

private void buttonExit\_Click(object sender, EventArgs e) // при нажатии на кнопку "Выйти из системы"

{

// открытие формы авторизации

formLogin.Show();

// закрытие этой формы водителя

this.Close();

// освобождение памяти

this.Dispose();

}

private void buttonCurrent\_Click(object sender, EventArgs e) // при нажатии на кнопку "Текущий груз"

{

// получение данных о текущем грузе

DataTable driverCargo = remoteTCP.Driver\_GetCargo(ID.ToString());

// проверка на наличие груза у водителя

if (driverCargo.Rows.Count != 0)

{

foreach (DataRow row in driverCargo.Rows)

{

// изменение полей в соответсвии с ID водителя

comboBox\_ID.Text = row["ID"].ToString();

textBoxDriverID.Text = row["DriverID"].ToString();

textBoxStatus.Text = row["Status"].ToString();

textBoxCargo.Text = row["Cargo"].ToString();

textBoxWeight.Text = row["Weight"].ToString();

textBoxFrom.Text = row["From"].ToString();

textBoxTo.Text = row["To"].ToString();

}

}

else

{

MessageBox.Show(

string.Format("У вас нет активного груза!", textBoxDriverID.Text),

"Ошибка!",

MessageBoxButtons.OK,

MessageBoxIcon.Information,

MessageBoxDefaultButton.Button1,

MessageBoxOptions.DefaultDesktopOnly);

}

}

private void buttonAccept\_Click(object sender, EventArgs e)

{

// проверка на пустое ID

if (comboBox\_ID.Text == "")

{

MessageBox.Show(

"Выберите ID!",

"Ошибка!",

MessageBoxButtons.OK,

MessageBoxIcon.Warning,

MessageBoxDefaultButton.Button1,

MessageBoxOptions.DefaultDesktopOnly);

}

else

{

// получение данных о текущем грузе

DataTable driverCargo = remoteTCP.Driver\_GetCargo(ID.ToString());

// проверка на наличие груза у водителя

if (driverCargo.Rows.Count == 0)

{

// проверка на наличие водителей у выбранного груза

if (remoteTCP.Driver\_GetDriverID(comboBox\_ID.Text) == 0)

{

// принятие груза

int result = remoteTCP.Driver\_CargoAccept(comboBox\_ID.Text, ID.ToString());

// если все прошло успешно

if (result == 0)

{

MessageBox.Show(

string.Format("Груз принят!\n ID = {0}\n Груз: {1}\n Из: {2}\n В: {3}", comboBox\_ID.Text, textBoxCargo.Text, textBoxFrom.Text, textBoxTo.Text),

"Успех!",

MessageBoxButtons.OK,

MessageBoxIcon.Information,

MessageBoxDefaultButton.Button1,

MessageBoxOptions.DefaultDesktopOnly);

}

else

{

MessageBox.Show(

string.Format("Этот груз уже был доставлен!", textBoxDriverID.Text),

"Ошибка!",

MessageBoxButtons.OK,

MessageBoxIcon.Information,

MessageBoxDefaultButton.Button1,

MessageBoxOptions.DefaultDesktopOnly);

}

// обновление таблицы

CargoReload();

}

else

{

MessageBox.Show(

string.Format("У этого груза уже есть водитель!", textBoxDriverID.Text),

"Ошибка!",

MessageBoxButtons.OK,

MessageBoxIcon.Information,

MessageBoxDefaultButton.Button1,

MessageBoxOptions.DefaultDesktopOnly);

}

}

else

{

MessageBox.Show(

string.Format("У вас уже есть активный груз!", textBoxDriverID.Text),

"Ошибка!",

MessageBoxButtons.OK,

MessageBoxIcon.Information,

MessageBoxDefaultButton.Button1,

MessageBoxOptions.DefaultDesktopOnly);

}

}

}

private void buttonCancel\_Click(object sender, EventArgs e)

{

// получение данных о текущем грузе

DataTable driverCargo = remoteTCP.Driver\_GetCargo(ID.ToString());

// проверка на наличие груза у водителя

if (driverCargo.Rows.Count != 0)

{

// требование подтверждения отказа от груза

if (MessageBox.Show(

string.Format("Вы действительно хотите отказаться от груза?\n ID = {0}\n Груз: {1}\n Из: {2}\n В: {3}", driverCargo.Rows[0]["ID"].ToString(), driverCargo.Rows[0]["Cargo"], driverCargo.Rows[0]["From"], driverCargo.Rows[0]["To"]),

"Внимание!",

MessageBoxButtons.YesNo,

MessageBoxIcon.Warning,

MessageBoxDefaultButton.Button1,

MessageBoxOptions.DefaultDesktopOnly) == DialogResult.Yes)

{

// отказ от груза

int result = remoteTCP.Driver\_CargoCancel(ID.ToString());

// если все прошло успешно

if (result == 0)

{

MessageBox.Show(

string.Format("Ваш груз отменен!"),

"Успех!",

MessageBoxButtons.OK,

MessageBoxIcon.Information,

MessageBoxDefaultButton.Button1,

MessageBoxOptions.DefaultDesktopOnly);

}

// обновление таблицы

CargoReload();

}

}

else

{

MessageBox.Show(

string.Format("У вас нет активного груза!", textBoxDriverID.Text),

"Ошибка!",

MessageBoxButtons.OK,

MessageBoxIcon.Information,

MessageBoxDefaultButton.Button1,

MessageBoxOptions.DefaultDesktopOnly);

}

}

private void buttonDelivery\_Click(object sender, EventArgs e)

{

// получение данных о текущем грузе

DataTable driverCargo = remoteTCP.Driver\_GetCargo(ID.ToString());

// проверка на наличие груза у водителя

if (driverCargo.Rows.Count != 0)

{

// требование подтверждения доставки груза

if (MessageBox.Show(

string.Format("Вы действительно хотите завершить доставку?\n ID = {0}\n Груз: {1}\n Из: {2}\n В: {3}", driverCargo.Rows[0]["ID"].ToString(), driverCargo.Rows[0]["Cargo"], driverCargo.Rows[0]["From"], driverCargo.Rows[0]["To"]),

"Внимание!",

MessageBoxButtons.YesNo,

MessageBoxIcon.Warning,

MessageBoxDefaultButton.Button1,

MessageBoxOptions.DefaultDesktopOnly) == DialogResult.Yes)

{

// завершение доставки груза

int result = remoteTCP.Driver\_CargoDelivery(ID.ToString());

// если все прошло успешно

if (result == 0)

{

MessageBox.Show(

string.Format("Ваш груз доставлен!"),

"Успех!",

MessageBoxButtons.OK,

MessageBoxIcon.Information,

MessageBoxDefaultButton.Button1,

MessageBoxOptions.DefaultDesktopOnly);

}

// обновление таблицы

CargoReload();

}

}

else

{

MessageBox.Show(

string.Format("У вас нет активного груза!", textBoxDriverID.Text),

"Ошибка!",

MessageBoxButtons.OK,

MessageBoxIcon.Information,

MessageBoxDefaultButton.Button1,

MessageBoxOptions.DefaultDesktopOnly);

}

}

}

}

FormLogist.cs

using System;

using System.Data;

using System.Runtime.Remoting;

using System.Runtime.Remoting.Lifetime;

using System.Windows.Forms;

using Server;

namespace Truckers

{

public partial class FormLogist : Form

{

RemoteObjectTCP remoteTCP; // удаленный объект

FormLogin formLogin; // ссылка на форму авторизации

DataTable cargoDataTable; // таблица с грузами

public FormLogist(FormLogin formLogin, string username)

{

InitializeComponent();

this.formLogin = formLogin; // присваивание ссылки на форму авторизации

label1.Text = "Логист: " + username; // вывод имени пользователя на форму

ConnectToServer();

CargoReload();

this.TopMost = true;

}

private void ConnectToServer() // установка соединения с сервером

{

RemotingConfiguration.Configure("C:/My Files/Универ/3 курс/Технологии программирования/Remoting\_Truckers/Truckers/ClientConfig.config", false);

remoteTCP = new RemoteObjectTCP();

ILease lease = (ILease)remoteTCP.InitializeLifetimeService();

ClientSponsor clientTCPSponsor = new ClientSponsor();

lease.Register(clientTCPSponsor);

}

private void CargoReload() // обновление таблицы

{

// получение данных

cargoDataTable = remoteTCP.Logist\_CargoReload();

// очистка выпадающего списка

comboBox\_ID.Items.Clear();

// проходим по каждой строке в cargoDataTable

foreach (DataRow row in cargoDataTable.Rows)

{

// получение значения поля ID из текущей строки

object value = row["ID"];

// добавление значения в comboBox\_ID

comboBox\_ID.Items.Add(value);

}

}

private void buttonReload\_Click(object sender, EventArgs e) // при нажатии на кнопку "Обновить"

{

// обновление таблицы

CargoReload();

}

private void comboBox\_ID\_SelectedValueChanged(object sender, EventArgs e) // при выборе ID в выпадающем списке

{

// поиск строк с выбранным ID

DataRow[] IDRow = cargoDataTable.Select(String.Format("ID = '{0}'", comboBox\_ID.SelectedItem.ToString()));

foreach (DataRow row in IDRow)

{

// изменение полей в соответсвии с выбраным ID

textBoxDriverID.Text = row["DriverID"].ToString();

comboBoxStatus.Text = row["Status"].ToString();

textBoxCargo.Text = row["Cargo"].ToString();

textBoxWeight.Text = row["Weight"].ToString();

textBoxFrom.Text = row["From"].ToString();

textBoxTo.Text = row["To"].ToString();

}

}

private void buttonExit\_Click(object sender, EventArgs e) // при нажатии на кнопку "Выйти из системы"

{

// открытие формы авторизации

formLogin.Show();

// закрытие этой формы логиста

this.Close();

// освобождение памяти

this.Dispose();

}

private void buttonSave\_Click(object sender, EventArgs e) // при нажатии на кнопку "Сохранить"

{

// проверка на пустые поля

if (comboBox\_ID.Text == "" || textBoxDriverID.Text == "" || comboBoxStatus.Text == "" || textBoxCargo.Text == "" || textBoxWeight.Text == "" || textBoxFrom.Text == "" || textBoxTo.Text == "")

{

MessageBox.Show(

"Не все поля заполнены!",

"Ошибка!",

MessageBoxButtons.OK,

MessageBoxIcon.Warning,

MessageBoxDefaultButton.Button1,

MessageBoxOptions.DefaultDesktopOnly);

}

// если все поля заполнены

else

{

string driversCount = remoteTCP.Logist\_GetDriversCount(textBoxDriverID.Text);

// проверка на наличие груза у водителя

if (driversCount == "0")

{

// вызов метода Logist\_CargoSave у удаленного объекта

string result = remoteTCP.Logist\_CargoSave(comboBox\_ID.Text, textBoxDriverID.Text, comboBoxStatus.Text, textBoxCargo.Text, textBoxWeight.Text, textBoxFrom.Text, textBoxTo.Text);

// если изменение прошло успешно

if (result == "0")

{

MessageBox.Show(

"Данные успешно сохранены!",

"Успех!",

MessageBoxButtons.OK,

MessageBoxIcon.Information,

MessageBoxDefaultButton.Button1,

MessageBoxOptions.DefaultDesktopOnly);

}

// обновление таблицы

CargoReload();

}

else

{

MessageBox.Show(

string.Format("У водителя с ID = {0} уже есть груз!", textBoxDriverID.Text),

"Ошибка!",

MessageBoxButtons.OK,

MessageBoxIcon.Information,

MessageBoxDefaultButton.Button1,

MessageBoxOptions.DefaultDesktopOnly);

}

}

}

private void buttonAdd\_Click(object sender, EventArgs e) // при нажатии на кнопку "Добавить"

{

// проверка на пустые поля

if (textBoxDriverID.Text == "" || comboBoxStatus.Text == "" || textBoxCargo.Text == "" || textBoxWeight.Text == "" || textBoxFrom.Text == "" || textBoxTo.Text == "")

{

MessageBox.Show(

"Не все поля заполнены!",

"Ошибка!",

MessageBoxButtons.OK,

MessageBoxIcon.Warning,

MessageBoxDefaultButton.Button1,

MessageBoxOptions.DefaultDesktopOnly);

}

// если все поля заполнены

else

{

// проверка на наличие груза у водителя

string driversCount = remoteTCP.Logist\_GetDriversCount(textBoxDriverID.Text);

if (driversCount == "0")

{

// вызов метода Logist\_CargoAdd у удаленного объекта

string result = remoteTCP.Logist\_CargoAdd(textBoxDriverID.Text, comboBoxStatus.Text, textBoxCargo.Text, textBoxWeight.Text, textBoxFrom.Text, textBoxTo.Text);

// если изменение прошло успешно

if (result == "0")

{

MessageBox.Show(

"Данные успешно добавлены!",

"Успех!",

MessageBoxButtons.OK,

MessageBoxIcon.Information,

MessageBoxDefaultButton.Button1,

MessageBoxOptions.DefaultDesktopOnly);

}

// обновление таблицы

CargoReload();

}

else

{

MessageBox.Show(

string.Format("У водителя с ID = {0} уже есть груз!", textBoxDriverID.Text),

"Ошибка!",

MessageBoxButtons.OK,

MessageBoxIcon.Information,

MessageBoxDefaultButton.Button1,

MessageBoxOptions.DefaultDesktopOnly);

}

}

}

private void buttonDelete\_Click(object sender, EventArgs e) // при нажатии на кнопку "Удалить"

{

// проверка на пустое ID

if (comboBox\_ID.Text == "")

{

MessageBox.Show(

"Выберите ID!",

"Ошибка!",

MessageBoxButtons.OK,

MessageBoxIcon.Warning,

MessageBoxDefaultButton.Button1,

MessageBoxOptions.DefaultDesktopOnly);

}

// если ID выбран

else

{

// требование подтверждения удаления от пользователя

if (MessageBox.Show(

string.Format("Вы действительно хотите удалить груз?\n ID = {0}\n Статус = {1}\n Груз: {2}\n Из: {3}\n В: {4}", comboBox\_ID.Text, comboBoxStatus.Text, textBoxCargo.Text, textBoxFrom.Text, textBoxTo.Text),

"Внимание!",

MessageBoxButtons.YesNo,

MessageBoxIcon.Warning,

MessageBoxDefaultButton.Button1,

MessageBoxOptions.DefaultDesktopOnly) == DialogResult.Yes)

{

// вызов метода Logist\_CargoDelete у удаленного объекта

string result = remoteTCP.Logist\_CargoDelete(comboBox\_ID.Text);

// если удаление прошло успешно

if (result == "0")

{

MessageBox.Show(

string.Format("Груз с ID = {0} успешно удален!", comboBox\_ID.Text),

"Успех!",

MessageBoxButtons.OK,

MessageBoxIcon.Information,

MessageBoxDefaultButton.Button1,

MessageBoxOptions.DefaultDesktopOnly);

}

// обновление таблицы

CargoReload();

}

}

}

}

}

**Приложение 3.** **Код реализации удаленных объектов**

RemoteObjectTCP.cs

using System;

using System.Data;

using System.Data.OleDb;

using System.Runtime.Remoting.Lifetime;

namespace Server

{

[Serializable]

public class RemoteObjectTCP : MarshalByRefObject

{

// ссылка на базу данных

private string connectionString = "Provider=Microsoft.Jet.OleDb.4.0;Data Source=C:/My Files/Универ/3 курс/Технологии программирования/Remoting\_Truckers/Truckers/data/TruckersDB.mdb";

public RemoteObjectTCP()

{

Console.WriteLine("Удаленный объект TCP создан!");

}

~RemoteObjectTCP()

{

Console.WriteLine("Удаленный объект TCP уничтожен!");

}

/// <summary>

/// Получение базы данных

/// </summary>

/// <returns>

/// Таблица DataTable, полученная из базы данных

/// </returns>

public DataTable Logist\_CargoReload()

{

using (OleDbConnection connection = new OleDbConnection(connectionString))

{

// открытие соединения

connection.Open();

DataTable dataTable = new DataTable(); // создание таблицы

OleDbDataAdapter adapter = new OleDbDataAdapter("SELECT \* FROM Cargo", connection);

adapter.Fill(dataTable); // запись результатов выполнения запроса в таблицу

// закрытие соединения

connection.Close();

// возвращение таблицы

return dataTable;

}

}

/// <summary>

/// Изменение значений в базе данных по ID

/// </summary>

/// <returns>

/// "0" - если изменение прошло успешно

/// </returns>

public string Logist\_CargoSave(string ID, string DriverID, string Status, string Cargo, string Weight, string From, string To)

{

using (OleDbConnection connection = new OleDbConnection(connectionString))

{

// открытие соединения

connection.Open();

// обновление данных по ID

string sql = string.Format("UPDATE Cargo SET DriverID={1}, Status='{2}', Cargo='{3}', Weight={4}, [From]='{5}', [To]='{6}' WHERE ID={0}", ID, DriverID, Status, Cargo, Weight, From, To);

OleDbCommand cmd = new OleDbCommand(sql, connection);

cmd.ExecuteNonQuery();

// закрытие соединения

connection.Close();

return "0";

}

}

/// <summary>

/// Добавление записей в базу данных

/// </summary>

/// <returns>

/// "0" - если добавление прошло успешно

/// </returns>

public string Logist\_CargoAdd(string DriverID, string Status, string Cargo, string Weight, string From, string To)

{

using (OleDbConnection connection = new OleDbConnection(connectionString))

{

// открытие соединения

connection.Open();

// добавление данных

string sql = string.Format("INSERT INTO Cargo (DriverID, Status, Cargo, Weight, [From], [To]) VALUES ({0}, '{1}', '{2}', {3}, '{4}', '{5}')", DriverID, Status, Cargo, Weight, From, To);

OleDbCommand cmd = new OleDbCommand(sql, connection);

cmd.ExecuteNonQuery();

// закрытие соединения

connection.Close();

return "0";

}

}

/// <summary>

/// Удаление записи из базы данных

/// </summary>

/// <returns>

/// "0" - если удаление прошло успешно

/// </returns>

public string Logist\_CargoDelete(string ID)

{

using (OleDbConnection connection = new OleDbConnection(connectionString))

{

// открытие соединения

connection.Open();

// удаление данных по ID

string sql = string.Format("DELETE FROM Cargo WHERE ID = {0}", ID);

OleDbCommand cmd = new OleDbCommand(sql, connection);

cmd.ExecuteNonQuery();

// закрытие соединения

connection.Close();

return "0";

}

}

/// <summary>

/// Поиск количества грузов по ID водителя

/// </summary>

/// <returns>

/// Количество грузов

/// </returns>

public string Logist\_GetDriversCount(string DriverID)

{

if (DriverID == "0")

return "0";

using (OleDbConnection connection = new OleDbConnection(connectionString))

{

// открытие соединения

connection.Open();

// обновление данных по ID

string sql = string.Format("SELECT \* FROM Cargo WHERE DriverID = {0}", DriverID);

DataTable dataTable = new DataTable(); // создание таблицы

OleDbDataAdapter adapter = new OleDbDataAdapter(sql, connection);

adapter.Fill(dataTable); // запись результатов выполнения запроса в таблицу

// закрытие соединения

connection.Close();

return dataTable.Rows.Count.ToString();

}

}

/// <summary>

/// Получение данных о грузе по ID водителя

/// </summary>

/// <returns>

/// Данные о грузе

/// </returns>

public DataTable Driver\_GetCargo(string DriverID)

{

using (OleDbConnection connection = new OleDbConnection(connectionString))

{

// открытие соединения

connection.Open();

// обновление данных по ID

string sql = string.Format("SELECT \* FROM Cargo WHERE DriverID = {0}", DriverID);

DataTable dataTable = new DataTable(); // создание таблицы

OleDbDataAdapter adapter = new OleDbDataAdapter(sql, connection);

adapter.Fill(dataTable); // запись результатов выполнения запроса в таблицу

// закрытие соединения

connection.Close();

return dataTable;

}

}

/// <summary>

/// Получение ID водителя груза по ID груза

/// </summary>

/// <returns>

/// ID водителя - если у груза есть водитель

/// и 0 - если у водителя нет груза

/// </returns>

public int Driver\_GetDriverID(string ID)

{

using (OleDbConnection connection = new OleDbConnection(connectionString))

{

// открытие соединения

connection.Open();

// обновление данных по ID

string sql = string.Format("SELECT \* FROM Cargo WHERE ID = {0}", ID);

DataTable dataTable = new DataTable(); // создание таблицы

OleDbDataAdapter adapter = new OleDbDataAdapter(sql, connection);

adapter.Fill(dataTable); // запись результатов выполнения запроса в таблицу

// закрытие соединения

connection.Close();

return Convert.ToInt32(dataTable.Rows[0]["DriverID"]);

}

}

/// <summary>

/// Взятие груза по ID груза и ID водителя

/// </summary>

/// <returns>

/// 0 - если все прошло успешно

/// и 1 - если груз уже был доставлен

/// </returns>

public int Driver\_CargoAccept(string ID, string DriverID)

{

using (OleDbConnection connection = new OleDbConnection(connectionString))

{

// открытие соединения

connection.Open();

DataTable dataTable = new DataTable(); // создание таблицы

OleDbDataAdapter adapter = new OleDbDataAdapter("SELECT \* FROM Cargo WHERE ID=" + ID, connection);

adapter.Fill(dataTable); // запись результатов выполнения запроса в таблицу

// проверка статуса груза

if (dataTable.Rows[0]["Status"].ToString() == "ready for unloading")

{

// обновление данных по ID

string sql = string.Format("UPDATE Cargo SET DriverID={1}, Status='on the way' WHERE ID={0}", ID, DriverID);

OleDbCommand cmd = new OleDbCommand(sql, connection);

cmd.ExecuteNonQuery();

// закрытие соединения

connection.Close();

return 0;

}

else

{

// закрытие соединения

connection.Close();

return 1;

}

}

}

/// <summary>

/// Отказ от груза по ID груза и ID водителя

/// </summary>

/// <returns>

/// 0 - если все прошло успешно

/// </returns>

public int Driver\_CargoCancel(string DriverID)

{

using (OleDbConnection connection = new OleDbConnection(connectionString))

{

// открытие соединения

connection.Open();

// обновление данных по ID

string sql = string.Format("UPDATE Cargo SET DriverID=0, Status='ready for unloading' WHERE DriverID={0}", DriverID);

OleDbCommand cmd = new OleDbCommand(sql, connection);

cmd.ExecuteNonQuery();

// закрытие соединения

connection.Close();

return 0;

}

}

/// <summary>

/// Доставка груза по ID груза и ID водителя

/// </summary>

/// <returns>

/// 0 - если все прошло успешно

/// </returns>

public int Driver\_CargoDelivery(string DriverID)

{

using (OleDbConnection connection = new OleDbConnection(connectionString))

{

// открытие соединения

connection.Open();

// обновление данных по ID

string sql = string.Format("UPDATE Cargo SET DriverID=0, Status='delivered' WHERE DriverID={0}", DriverID);

OleDbCommand cmd = new OleDbCommand(sql, connection);

cmd.ExecuteNonQuery();

// закрытие соединения

connection.Close();

return 0;

}

}

public override object InitializeLifetimeService()

{

ILease lease = (ILease)base.InitializeLifetimeService();

if (lease.CurrentState == LeaseState.Initial)

{

lease.InitialLeaseTime = TimeSpan.FromSeconds(3);

lease.SponsorshipTimeout = TimeSpan.FromSeconds(10);

lease.RenewOnCallTime = TimeSpan.FromSeconds(2);

}

return lease;

}

}

}

RemoteObjectHTTP.cs

using System;

using System.Data;

using System.Data.OleDb;

using System.Runtime.Remoting.Lifetime;

namespace Server

{

[Serializable]

public class RemoteObjectHTTP : MarshalByRefObject

{

// ссылка на базу данных

private string connectionString = "Provider=Microsoft.Jet.OleDb.4.0;Data Source=C:/My Files/Универ/3 курс/Технологии программирования/Remoting\_Truckers/Truckers/data/TruckersDB.mdb";

public RemoteObjectHTTP()

{

Console.WriteLine("Удаленный объект HTTP создан!");

}

~RemoteObjectHTTP()

{

Console.WriteLine("Удаленный объект HTTP уничтожен!");

}

/// <summary>

/// Поиск пользователей по логину и паролю

/// </summary>

/// <returns>

/// Количество пользователей с таким логином и паролем

/// </returns>

public string Autorization(string login, string password)

{

using (OleDbConnection connection = new OleDbConnection(connectionString))

{

// открытие соединения

connection.Open();

DataTable dataTable = new DataTable(); // создание таблицы

OleDbDataAdapter adapter = new OleDbDataAdapter(string.Format("SELECT \* FROM Users WHERE Login='{0}' AND Password='{1}'", login, password), connection);

adapter.Fill(dataTable); // запись результатов выполнения запроса в таблицу

// закрытие соединения

connection.Close();

// если пользователь с таким логином и паролем не найден

if (dataTable.Rows.Count == 0)

return "1";

else

{

// если пользователь водитель

if (dataTable.Rows[0][2].ToString() == "Водитель")

return "driver";

// если пользователь логист

else if (dataTable.Rows[0][2].ToString() == "Логист")

return "logist";

else

return "1";

}

}

}

/// <summary>

/// Регистрация пользователя

/// </summary>

/// <returns>

/// "0" - если регистрация успешна

/// "1" - если пользователь с таким логином уже существует

/// </returns>

public string Registration(string name, string login, string password, string post)

{

using (OleDbConnection connection = new OleDbConnection(connectionString))

{

// открытие соединения

connection.Open();

// проверка на повтор логина

DataTable dataTable = new DataTable(); // создание таблицы

OleDbDataAdapter adapter = new OleDbDataAdapter("SELECT \* FROM Users WHERE Login='" + login + "'", connection);

adapter.Fill(dataTable); // запись результатов выполнения запроса в таблицу

// если пользователей с таким логином не существует

if (dataTable.Rows.Count == 0)

{

// регистрация нового пользователя

string sql = string.Format("INSERT INTO Users ([Username], [Post], [Login], [Password]) VALUES ('{0}', '{1}', '{2}', '{3}')", name, post, login, password);

OleDbCommand cmd = new OleDbCommand(sql, connection);

cmd.ExecuteNonQuery();

connection.Close();

return "0";

}

// если пользователь с таким логином уже существует

else

{

connection.Close();

return "1";

}

}

}

/// <summary>

/// Вывод имени пользователя по логину

/// </summary>

/// <returns>

/// Имя пользователя

/// </returns>

public string GetUsername(string login)

{

using (OleDbConnection connection = new OleDbConnection(connectionString))

{

// открытие соединения

connection.Open();

DataTable dataTable = new DataTable(); // создание таблицы

OleDbDataAdapter adapter = new OleDbDataAdapter(string.Format("SELECT \* FROM Users WHERE Login='{0}'", login), connection);

adapter.Fill(dataTable); // запись результатов выполнения запроса в таблицу

// закрытие соединения

connection.Close();

// вывод имени пользователя

return dataTable.Rows[0][1].ToString();

}

}

/// <summary>

/// Вывод ID пользователя по логину

/// </summary>

/// <returns>

/// ID пользователя

/// </returns>

public int GetID(string login)

{

using (OleDbConnection connection = new OleDbConnection(connectionString))

{

// открытие соединения

connection.Open();

DataTable dataTable = new DataTable(); // создание таблицы

OleDbDataAdapter adapter = new OleDbDataAdapter(string.Format("SELECT \* FROM Users WHERE Login='{0}'", login), connection);

adapter.Fill(dataTable); // запись результатов выполнения запроса в таблицу

// закрытие соединения

connection.Close();

// вывод имени пользователя

return Convert.ToInt32(dataTable.Rows[0][0]);

}

}

public override object InitializeLifetimeService()

{

ILease lease = (ILease)base.InitializeLifetimeService();

if (lease.CurrentState == LeaseState.Initial)

{

lease.InitialLeaseTime = TimeSpan.FromSeconds(3);

lease.SponsorshipTimeout = TimeSpan.FromSeconds(10);

lease.RenewOnCallTime = TimeSpan.FromSeconds(2);

}

return lease;

}

}

}

**Приложение 4.** **Код реализации серверной части (ASP.NET)**

Default.aspx

<%@ Page Title="Главная" Language="C#" MasterPageFile="~/Site.Master" AutoEventWireup="true" CodeBehind="Default.aspx.cs" Inherits="ASP.NET\_Truckers.\_Default" %>

<asp:Content ID="BodyContent" ContentPlaceHolderID="MainContent" runat="server">

<div class="whitefont">

<h2>О проекте</h2>

<hr />

<p>Этот проект создан с использованием технологии ASP.NET.</p>

<p><b>Название:</b> Панель управления грузами</p>

<p><b>Цель:</b> выполнение курсовой работы по дисциплине "Технологии программирования".</p>

<p><b>Функции:</b> авторизация логиста или водителя. Панель управления зависит от должности. Источником данные о пользователях и грузах является база данных, расположенная в папке проекта <i>data/TruckersDB.mdb</i>.</p>

<ul>

<li>Возможности логиста:</li>

<ol>

<li>Просматривать данные о грузах</li>

<li>Редактировать данные о грузах</li>

<li>Добавлять грузы</li>

<li>Удалять грузы</li>

</ol>

<li>Возможности водителя:</li>

<ol>

<li>Просматривать информацию о текущем грузе</li>

<li>Получать груз</li>

<li>Отменять полученный груз</li>

<li>Доставлять груз</li>

</ol>

</ul>

<hr />

<asp:Button runat="server" class="blackfont" Text="Перейти &raquo;" Width="100px" OnClick="Button\_Click"/>

</div>

</asp:Content>

Default.aspx.cs

using System;

using System.Web.UI;

namespace ASP.NET\_Truckers

{

public partial class \_Default : Page

{

protected void Page\_Load(object sender, EventArgs e)

{

}

protected void Button\_Click(object sender, EventArgs e)

{

Response.Redirect("~/Login.aspx");

}

}

}

Login.aspx

<%@ Page Title="Авторизация" Language="C#" MasterPageFile="~/Site.Master" AutoEventWireup="true" CodeBehind="Login.aspx.cs" Inherits="ASP.NET\_Truckers.Login" %>

<asp:Content ID="BodyContent" ContentPlaceHolderID="MainContent" runat="server">

<div class="whitefont">

<h2><%: Title %></h2>

<hr />

<p id="responseLabel" runat="server">Вы не авторизованы</p>

<h5>Логин</h5>

</div>

<input id="loginField" runat="server" type="text" class="blackfont" style="width: 400px"/>

<h5 class="whitefont">Пароль</h5>

<input id="passwordField" runat="server" type="text" class="blackfont" style="width: 400px"/>

<div runat="server" class="blackfont" id="button0"><br /><asp:Button runat="server" OnClick="Authorize" Text="Авторизоваться" Width="400px"/></div>

<div runat="server" class="blackfont" id="button1">

<br /><asp:Button runat="server" OnClick="button2\_Click" Text="Сменить пользователя" Width="400px"/>

<br /><br /><asp:Button runat="server" OnClick="button1\_Click" Text="Открыть панель управления" Width="400px"/>

</div>

</asp:Content>

Login.aspx.cs

using System;

using System.Data;

namespace ASP.NET\_Truckers

{

public partial class Login : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

if (Session["visibilitydb"] == null)

{

button1.Visible = false;

}

else

{

bool visible = (bool)Session["visibilitydb"];

button0.Visible = !visible;

button1.Visible = visible;

}

if (Session["responseLabel"] != null)

{

responseLabel.InnerHtml = (string)Session["responseLabel"];

if (Session["userName"] != null)

{

loginField.Disabled = true;

passwordField.Disabled = true;

}

}

else

{

responseLabel.InnerHtml = "Вы не авторизованы";

Session["visibilitydb"] = null;

Session["userID"] = null;

Session["userPost"] = null;

Session["userName"] = null;

loginField.Disabled = false;

passwordField.Disabled = false;

}

}

protected void Authorize(object sender, EventArgs e)

{

button1.Visible = false;

string login = loginField.Value.ToString();

string Password = passwordField.Value.ToString();

if (login == "" || Password == "")

{

Session["responseLabel"] = "Введите логин и пароль!";

Server.TransferRequest("Login.aspx");

return;

}

DataTable response = SqlResponses.GetSqlFromDB("SELECT \* FROM Users WHERE Login='" + login + "' AND Password='" + Password + "'");

if (response.Rows.Count == 0)

{

Session["responseLabel"] = "Неверный логин или пароль!";

Server.TransferRequest("Login.aspx");

return;

}

foreach (DataRow dataRow in response.Rows)

{

Session["visibilitydb"] = true;

Session["userID"] = (int)dataRow["ID"];

Session["userPost"] = (string)dataRow["Post"];

Session["userName"] = (string)dataRow["Username"];

Session["responseLabel"] = "Вы авторизованы как " + ((string)dataRow["Post"]).ToLower() + " " + (string)dataRow["Username"];

}

Server.TransferRequest("Login.aspx");

}

protected void button1\_Click(object sender, EventArgs e)

{

if (Session["userPost"].ToString() == "Логист")

Response.Redirect("~/Logist.aspx");

else if (Session["userPost"].ToString() == "Водитель")

Response.Redirect("~/Logist.aspx");

}

protected void button2\_Click(object sender, EventArgs e)

{

Session["responseLabel"] = null;

Session["visibilitydb"] = null;

Session["userID"] = null;

Session["userPost"] = null;

Session["nauserName"] = null;

Response.Redirect("~/Login.aspx");

}

}

}

Logist.aspx

<%@ Page Title="Логист" Language="C#" MasterPageFile="~/Site.Master" AutoEventWireup="true" CodeBehind="Logist.aspx.cs" Inherits="ASP.NET\_Truckers.Logist" %>

<asp:Content ID="BodyContent" ContentPlaceHolderID="MainContent" runat="server">

<div class="whitefont">

<h2>Панель управления логиста</h2>

<hr />

<h5 id="responseLabel" runat="server">Вы авторизованы как логист </h5>

</div>

<asp:Button id="buttonExit" OnClick="buttonExit\_Click" runat="server" Text="« Назад" class="blackfont" Width="100px" /><br/>

<hr class="whitefont"/>

<asp:Button id="gridshow" runat="server" Text="Показать все грузы" class="blackfont" Width="400px" OnClick="gridshow\_Click"/><br />

<asp:Button id="gridhide" runat="server" Text="Скрыть все грузы" class="blackfont" Width="400px" OnClick="gridhide\_Click" />

<div id="GridViewShow" runat="server" class="blackfont">

<br /><asp:GridView id="GridView1" class="whitefont" runat="server"></asp:GridView>

</div>

<hr class="whitefont"/>

<div class="whitefont">

<p>Выберие ID груза, чтобы получить информацию о грузе</p>

<select id="cargoID" class="blackfont" runat="server"></select>

<asp:Button id="buttonChoose" runat="server" Text="Выбрать" class="blackfont" Width="100px" OnClick="buttonChoose\_Click" /><br /><br />

<p runat="server">ID водителя</p>

<input id="cargoDriverID" runat="server" type="text" class="blackfont" style="width: 400px"/><br /><br />

<p runat="server">Статус</p>

<input id="cargoStatus" runat="server" type="text" class="blackfont" style="width: 400px"/><br /><br />

<p runat="server">Наименование груза</p>

<input id="cargoName" runat="server" type="text" class="blackfont" style="width: 400px"/><br /><br />

<p runat="server">Масса груза в тоннах</p>

<input id="cargoWeight" runat="server" type="text" class="blackfont" style="width: 400px"/><br /><br />

<p runat="server">Пункт отправления</p>

<input id="cargoFrom" runat="server" type="text" class="blackfont" style="width: 400px"/><br /><br />

<p runat="server">Пункт назначения</p>

<input id="cargoTo" runat="server" type="text" class="blackfont" style="width: 400px"/>

</div>

<hr class="whitefont" />

<div class="blackfont">

<asp:Button id="buttonReload" runat="server" Text="Обновить" Width="100px" OnClick="gridhide\_Click" />

<asp:Button id="buttonSave" runat="server" Text="Сохранить" Width="100px" OnClick="gridhide\_Click" /><br />

<asp:Button id="buttonAdd" runat="server" Text="Добавить" Width="100px" OnClick="gridhide\_Click" />

<asp:Button id="buttonDelete" runat="server" Text="Удалить" Width="100px" OnClick="gridhide\_Click" />

</div>

</asp:Content>

Logist.aspx.cs

using System;

using System.Data;

using System.Data.OleDb;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace ASP.NET\_Truckers

{

class SqlResponses

{

static string connectionString = @"Provider=Microsoft.ACE.OLEDB.16.0;Data Source = C:/My Files/Универ/3 курс/Технологии программирования/ASP.NET\_Truckers/data/TruckersDB.mdb";

public static void SqlFromDB(string sql)

{

OleDbConnection connection = new OleDbConnection(connectionString);

connection.Open();

OleDbCommand cmd = new OleDbCommand(sql, connection);

cmd.ExecuteNonQuery();

connection.Close();

}

public static DataTable GetSqlFromDB(string sql)

{

OleDbConnection connection = new OleDbConnection(connectionString);

connection.Open();

DataTable dataTable = new DataTable();

OleDbDataAdapter adapter = new OleDbDataAdapter(sql, connection);

adapter.Fill(dataTable);

connection.Close();

return dataTable;

}

}

public partial class Logist : System.Web.UI.Page

{

DataSet dst;

OleDbConnection myCnt;

OleDbDataAdapter dbAdpt1;

static string connectionString = @"Provider=Microsoft.ACE.OLEDB.16.0;Data Source = C:/My Files/Универ/3 курс/Технологии программирования/ASP.NET\_Truckers/data/TruckersDB.mdb";

protected void Page\_Load(object sender, EventArgs e)

{

dst = new DataSet();

myCnt = new OleDbConnection();

myCnt.ConnectionString = connectionString;

string slct1 = "SELECT \* From Cargo";

dbAdpt1 = new OleDbDataAdapter(slct1, myCnt);

dbAdpt1.Fill(dst, "Cargo");

GridView1.DataSource = dst.Tables["Cargo"];

Page.DataBind();

if (Session["GridVisibility"] == null)

{

GridViewShow.Visible = false;

}

else

{

bool visible = (bool)Session["GridVisibility"];

GridViewShow.Visible = visible;

}

if (Session["responseLabel"] != null)

{

responseLabel.InnerHtml = (string)Session["responseLabel"];

CargoID\_Reload();

}

else

{

Response.Redirect("~/Login.aspx");

}

}

public void CargoID\_Reload()

{

cargoID.Items.Clear();

foreach (DataRow dataRow in dst.Tables["Cargo"].Rows)

{

cargoID.Items.Add(new ListItem { Text = dataRow["Cargo"].ToString(), Value = dataRow["ID"].ToString()});

}

if (Session["cargoDriverID"] != null)

{

cargoDriverID.Value = Session["cargoDriverID"].ToString();

cargoStatus.Value = Session["cargoStatus"].ToString();

cargoName.Value = Session["cargoName"].ToString();

cargoWeight.Value = Session["cargoWeight"].ToString();

cargoFrom.Value = Session["cargoFrom"].ToString();

cargoTo.Value = Session["cargoTo"].ToString();

}

}

public void buttonChoose\_Click(object sender, EventArgs e)

{

string currentID = cargoID.Items[cargoID.SelectedIndex].Value;

DataTable response = SqlResponses.GetSqlFromDB("SELECT \* FROM Cargo WHERE ID=" + currentID);

foreach (DataRow dataRow in response.Rows)

{

Session["cargoDriverID"] = dataRow["DriverID"].ToString();

Session["cargoStatus"] = dataRow["Status"].ToString();

Session["cargoName"] = dataRow["Cargo"].ToString();

Session["cargoWeight"] = dataRow["Weight"].ToString();

Session["cargoFrom"] = dataRow["From"].ToString();

Session["cargoTo"] = dataRow["To"].ToString();

}

Server.TransferRequest("Logist.aspx");

}

protected void gridshow\_Click(object sender, EventArgs e)

{

Session["GridVisibility"] = true;

Server.TransferRequest("Logist.aspx");

}

protected void gridhide\_Click(object sender, EventArgs e)

{

Session["GridVisibility"] = false;

Server.TransferRequest("Logist.aspx");

}

protected void buttonExit\_Click(object sender, EventArgs e)

{

Response.Redirect("~/Login.aspx");

}

}

}