Министерство науки и высшего образования РФ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Российский экономический университет им. Г.В. Плеханова»

Московский приборостроительный техникум

Специальность: 09.02.05 Прикладная информатика (по отраслям)

Профессиональный модуль: ПМ 02 Разработка, внедрение и адаптация программного обеспечения отраслевой направленности

Междисциплинарный курс: МДК 02.01 Разработка, внедрение и адаптация программного обеспечения отраслевой направленности

МПТ.И-3-17.МДК.02.01 00 20

ОТЧЁТ

по практическим работам:

Практическая работа № 28 «Организация модулей работы с источниками данных. Таблицы и представления.»

Практическая работа № 29 «Организация модулей работы с источниками данных. Хранимые процедуры.»

Практическая работа № 30 «Разработка модуля подключения к источнику данных.»

Практическая работа № 31 «Разработка интерфейсов справочников.»

Практическая работа № 32 «Разработка стандартных функций баз данных.»

Практическая работа № 33 «Разработка автоматизированных функций программного обеспечения.»

Практическая работа № 34 «Разработка экспорта данных в Microsoft Word, Excel и формат PDF.»

Практическая работа № 35 «Разработка системы аутентификации пользователей.»

Практическая работа № 36 «Публикация программного продукта в Git Hub.»

|  |  |
| --- | --- |
|  | Выполнил |
| Проверил: | Студент группы И-3-17 |
| И.М. Щаников \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ E. A. Сергеева |
| «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г. | «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г. |
| Оценка: \_\_\_\_\_ (\_\_\_\_\_\_\_\_\_\_) |  |

2020

СОДЕРЖАНИЕ

[1. ЦЕЛИ ПРАКТИЧЕСКОЙ РАБОТЫ 3](#_Toc21173925)

[2. ЭТАПЫ ВЫПОЛНЕНИЯ 5](#_Toc21173926)

[3. ВЫВОД 6](#_Toc21173927)

[4. СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 7](#_Toc21173928)

1. ЦЕЛИ ПРАКТИЧЕСКОЙ РАБОТЫ

Практическая работа № 28:

* Разрабатывать программное обеспечение с помощью языков программирования информационного контента;
* Программировать на встроенных алгоритмических языках.

Практическая работа № 29:

* Разрабатывать программное обеспечение с помощью языков программирования информационного контента;
* Программировать на встроенных алгоритмических языках.

Практическая работа № 30:

* Разрабатывать программное обеспечение с помощью языков программирования информационного контента;
* Программировать на встроенных алгоритмических языках.

Практическая работа № 31:

* Разрабатывать программное обеспечение с помощью языков программирования информационного контента;
* Создавать анимации в специализированных программных средах;
* Работать с мультимедийными инструментальными средствами.

Практическая работа № 32:

* Разрабатывать программное обеспечение с помощью языков программирования информационного контента.

Практическая работа № 33:

* Разрабатывать программное обеспечение с помощью языков программирования информационного контента;
* Программировать на встроенных алгоритмических языках.

Практическая работа № 34:

* Разрабатывать программное обеспечение с помощью языков программирования информационного контента;
* Программировать на встроенных алгоритмических языках.

Практическая работа № 35:

* Разрабатывать программное обеспечение с помощью языков программирования информационного контента;
* Программировать на встроенных алгоритмических языках.

Практическая работа № 36:

* Размещать информационный контент в глобальных и локальных сетях;
* Использовать инструментальные среды поддержки разработки, системы управления контентом;
* Использовать системы управления контентом для решения поставленных задач.

1. ЭТАПЫ ВЫПОЛНЕНИЯ
   1. Практическая работа № 28
      1. Описание структуры базы данных

Таблица 1 – Описание структуры базы данных

|  |  |
| --- | --- |
| Название элемента | Краткое описание назначения объекта |
| Sotrudniki | Таблица сотрудники хранит в себе данные про личные данные сотрудников, данные о авторизации |
| Otdel | Данная таблица хранит информацию название и номере отдела |
| Kachestvo | Данная таблица хранит информацию о название качестве и качестве товара |
| Spisok\_Tovarov | Данная таблица хранит информацию о название товара, его дате поставки, а также сроке годности |
| Zakaz | Данная таблица хранит информацию о личных данных о поставщике |

* + 1. Описание модулей работы с таблицами и представлениями

Таблица 2 – Описание модулей программы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название модуля | Краткое описание | Пространства имён | Классы | Количество строк |
| Table\_Class | Специальный модуль для заполнения “DataTable” в зависимости от запроса SQL | System | Table\_Class.cs | 51 |
| System.Collections |
| System.Generic |
| System.Linq |
| System.Text |
| System.Threading.Task |
| System.Data |
| System.Data.SqlClient |

* + 1. Код программы

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Data;

using System.Data.SqlClient;

namespace Sale\_App

{

public class Table\_Class

{

//Глобальный класс виртуальной таблицы

public DataTable table = new DataTable();

//Локальная переменная SQLCommand

private SqlCommand command = new SqlCommand("",Configuration\_class.connection);

//Глобальная переменная организации зависимости и прослушивания сервера

public SqlDependency Dependency = new SqlDependency();

/// <summary>

/// Заполение DataTable в зависимости от введённого SQL запроса

/// </summary>

/// <param name="SQL\_Select\_Query">Переменная запроса SQL</param>

public Table\_Class(string SQL\_Select\_Query)

{

command.Notification = null;//Отключение оповещений у команды

command.CommandText = SQL\_Select\_Query;//Присвение SQL запроса SQLCommand

Dependency.AddCommandDependency(command);//Присвоение команды в связку

//прослушивания

try

{

//Запуск прослушивания

SqlDependency.Start(Configuration\_class.connection.ConnectionString);

//Открытие подключения

Configuration\_class.connection.Open();

//Записть данных в табличном виде в виртулальную таблицу

table.Load(command.ExecuteReader());

}

catch(Exception ex)

{

//Вывод сообщения об ошибке

System.Windows.Forms.MessageBox.Show(ex.Message);

}

finally

{

//Закрытие подключения

Configuration\_class.connection.Close();

}

}

}

}

* 1. Практическая работа № 29
     1. Описание структуры базы данных

В данном пункте в виде таблицы необходимо дать описание хранимым процедурам в базе данных, их назначение.

Таблица 1 – Описание структуры базы данных

|  |  |
| --- | --- |
| Название элемента | Краткое описание назначения объекта |
| [dbo].[Sotrudniki\_insert] | Данная процедура предназначена для добавления данных в сущность сотрудники |
| [dbo].[ Sotrudniki \_update] | Данная процедура предназначена для изменения данных в сущности сотрудники |
| [dbo].[ Sotrudniki \_delete] | Данная процедура предназначена для удаления данных в сущности сотрудники |
| [dbo].[Otdel\_insert] | Данная процедура предназначена для добавления данных в сущность отдел |
| [dbo].[Otdel\_update] | Данная процедура предназначена для изменения данных в сущности отдел |
| [dbo].[Otdel\_delete] | Данная процедура предназначена для удаления данных в сущности отдел |
| [dbo].[Kachestvo\_insert] | Данная процедура предназначена для добавления данных в сущность качество |
| [dbo].[Kachestvo\_update] | Данная процедура предназначена для изменения данных в сущности качество |
| [dbo].[Kachestvo\_delete] | Данная процедура предназначена для удаления данных в сущности качество |
| [dbo].[Spisok\_Tovarov\_insert] | Данная процедура предназначена для добавления данных в сущность список товаров |
| [dbo].[ Spisok\_Tovarov\_update] | Данная процедура предназначена для изменения данных в сущности список товаров |
| [dbo].[ Spisok\_Tovarov\_delete] | Данная процедура предназначена для удаления данных в сущности список товаров |
| [dbo].[Zakaz\_insert] | Данная процедура предназначена для добавления данных в сущность заказ |
| [dbo].[Zakaz\_update] | Данная процедура предназначена для изменения данных в сущности заказ |
| [dbo].[Zakaz\_delete] | Данная процедура предназначена для удаления данных в сущности заказ |

* + 1. Описание модулей работы с хранимыми процедурами

Таблица 2 – Описание модулей программы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название модуля | Краткое описание | Пространства имён | Классы | Количество строк |
| Procedure\_Class | Обращение к хранимым процедурам MS SQL | System | Procedure\_Class | 71 |
| System.Collections |
| System.Generic |
| System.Linq |
| System.Text |
| System.Threading.Task |
| System.Data |
| System.Data.SqlClient |

* + 1. Код программы

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Data;

using System.Data.SqlClient;

using System.Collections;

namespace Sale\_App

{

public class Procedure\_Class

{

SqlCommand command = new SqlCommand("", Configuration\_class.connection);

/// <summary>

/// Метод обращения к любой хранимой процедуре Microsoft SQL Server

/// </summary>

/// <param name="Procedure\_name">Водимое название процедуры из БД</param>

/// <param name="fileld\_value">Не типизированная коллекция значений приложения</param>

public void procedure\_Execution(string Procedure\_name, ArrayList fileld\_value)

{

//Запрос на вывод списка параметров, конкретной хранимолй процедуры

//в зависимости от введённого разработчиком названия Stored Procedure

Table\_Class table = new Table\_Class(string.Format("select name from sys.parameters " +

"where " +

"object\_id =" +

" (select object\_id from sys.procedures where name = '{0}')", Procedure\_name));

try

{

//Настройка SQLCommand для работы с хранимыми процедурами

command.CommandType = CommandType.StoredProcedure;

//Присвоение в текст команды названия хранимой процедуры

command.CommandText = string.Format("[dbo].[{0}]", Procedure\_name);

//Отчистка параметров

command.Parameters.Clear();

for(int i = 0; i<table.table.Rows.Count; i++)

{

//Сопоставление значений коллекций в значения параметров

command.Parameters.AddWithValue(table.table.Rows[i][0].ToString(),

fileld\_value[i]);

}

//Объявление события на перехват сообщений из БД

Configuration\_class.connection.InfoMessage += Connection\_InfoMessage;

//Открытие подключения

Configuration\_class.connection.Open();

//Выполнение запроса процедуры

command.ExecuteNonQuery();

}

catch

{

}

finally

{

//Закрытие подключения

Configuration\_class.connection.Close();

}

}

/// <summary>

/// Обработчик события о получении собщения с сервера БД

/// </summary>

/// <param name="sender">Ссылка на объект</param>

/// <param name="e">Аргумет сообщения сервера</param>

private void Connection\_InfoMessage(object sender, SqlInfoMessageEventArgs e)

{

//Вывод сообщения с сервера в диалоговое окно

System.Windows.Forms.MessageBox.Show(e.Message);

//Снятиек с события обработчика метода

Configuration\_class.connection.InfoMessage -= Connection\_InfoMessage;

}

}

}

* 1. Практическая работа № 30
     1. Описание пространств имён

В данном пункте в виде таблицы необходимо описать название пространства имён, и назначение данного пространства в рамках выполняемой работы.

Таблица 1 – Описание пространств имён

|  |  |
| --- | --- |
| Название элемента | Краткое описание назначения элемента |
| SqlClient | Передача данных |
| Sql | Подключение к SQL server |
| SqlDependency | Предназначен для установки зависимости уведомления запросов. |
| Win32 | Предоставляет работу с регистром и с событиями ОС |
| System | Возможность обрабатывать события и стандартную работу с классами |
| Collections | Предназначен для использования коллекций |
| Forms | Использовать формы |
| RegistryKey | Предназначен для работы с реестром |
| DataTable | Предназначен для предоставления одной таблицы данных в памяти |
| SqlCommand | Предназначен для работы с процедурами над базой данных |

* + 1. Описание классов

Таблица 2 – Описание классов

|  |  |
| --- | --- |
| Название элемента | Краткое описание назначения элемента |
| Configuration\_class | Для подключения к БД |
| Table\_Class | Для работы с таблицами БД |
| Proceudre\_Class | Работа с хранимыми процедурами |

* + 1. Код программы

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Microsoft.Win32;

using System.Data;

using System.Data.Sql;

using System.Data.SqlClient;

using System.Net;

using System.Net.NetworkInformation;

namespace Sale\_App

{

public class Configuration\_class

{

public event Action<DataTable> Server\_Collection;

//Получает колекцию доступных серверов

public event Action<DataTable> Data\_Base\_Collection;

//Получает колекцию доступных БД на сервере

public event Action<bool> Conection\_Checked;

//Определяет статус подключения

public string DS = "Empty",// Переменная Data Source

IC = "Empty";//Переменная Initial Catalog

public string ds = "";//Проверка подключения Data Source

public static SqlConnection connection = new SqlConnection();

/// <summary>

/// Ментод получения информации о строке подключения к БД

/// свойств Data Source и Initial Catalog технологии долступа

/// к данным ADO.Net

/// </summary>

public void SQL\_Server\_Configuration\_Get()

{

//Создаёт каталог в одном из корней реестра ОС

RegistryKey registry = Registry.CurrentUser;

//Создаёт папку в выбраном коревом каталоге рееста ОС

RegistryKey key = registry.CreateSubKey("Server\_ConfigurationLena");

try

{

//Пытаюсь получить значения из переменных в реестре

DS = key.GetValue("DS").ToString();

IC = key.GetValue("IC").ToString();

}

catch

{

DS = "Empty";

IC = "Empty";

}

finally

{

//Обновление строки подкючения

connection.ConnectionString = "Data Source = "+DS+

"; Initial Catalog = "+IC+

"; Integrated Security = true;";

}

}

/// <summary>

/// Метод обновления информации о подкелючении к источнику данных

/// по технологии ADO.Net

/// </summary>

/// <param name="ds">Запись значения Data Source</param>

/// <param name="ic">Запись значения Initial Catalog</param>

public void SQL\_Server\_Configuration\_Set(string ds, string ic)

{

RegistryKey registry = Registry.CurrentUser;

RegistryKey key = registry.CreateSubKey("Server\_ConfigurationLena");

key.SetValue("DS", ds);//Запись значения в переменную реестра

key.SetValue("IC", ic);

SQL\_Server\_Configuration\_Get();

}

/// <summary>

/// Метод возвращает список доступных серверов в локаьном окружении

/// </summary>

public void SQL\_Server\_Enumurator()

{

//Полдучет сведения о доступных серверах

SqlDataSourceEnumerator sourceEnumerator

= SqlDataSourceEnumerator.Instance;

//Присвоение Event Action списка серверов ввиде таблицы

Server\_Collection(sourceEnumerator.GetDataSources());

}

/// <summary>

/// Метод проверки подключения к источнику данных

/// </summary>

public void SQL\_Data\_Base\_Checking()

{

connection.ConnectionString = "Data Source = " + ds + "; " +

"Initial Catalog = master; Integrated Security = True";

try

{

//Если подключение по источнеику данных открыть можно

// в Event Action присваиваю true

connection.Open();

Conection\_Checked(true);

}

catch

//В противном случае false

{

Conection\_Checked(false);

}

finally

{

connection.Close();

}

}

/// <summary>

/// Метод получает список доступных на сервере

/// баз данных по технологии доступа к данным ADO.Net

/// </summary>

public void SQL\_Data\_Base\_Collection()

{

//Запрос на выборку названия баз данных с конкретного сервера

//где база не назвается master, model, tempdb, msdb

//и имеет схожее название с Sale\_Data\_Base

SqlCommand command = new SqlCommand("select name from sys.databases " +

"where name not in ('master','tempdb','model','msdb') " +

"and name like 'Sale\_Data\_Base%'", connection);

try

{

connection.Open();

DataTable table = new DataTable();

table.Load(command.ExecuteReader());

Data\_Base\_Collection(table);

}

catch

{

}

finally

{

connection.Close();

}

}

public void Machine\_Name\_Get()

{

RegistryKey registry = Registry.CurrentUser;

RegistryKey key = registry.CreateSubKey("Server\_ConfigurationLena");

Machine\_Name = key.GetValue("Machine\_Name").ToString();

}

public void Machine\_Name\_Set(string machine\_name)

{

RegistryKey registry = Registry.CurrentUser;

RegistryKey key = registry.CreateSubKey("Server\_ConfigurationLena");

key.SetValue("Machine\_Name", machine\_name);

Machine\_Name\_Get();

}

//Строковые статические переменные

//Название организации, Путь сохранения документа,

//Название персонального компьютера

public static string Organization\_Name,

Save\_Files\_Path, Machine\_Name;

//Переменные отступа в документе

public static Int32 doc\_Left\_Merge,

doc\_Right\_Merg,

doc\_Top\_Merge, doc\_Bottom\_Merge;

/// <summary>

/// Получение данных о конфигурации документа

/// </summary>

public void Document\_Configuration\_Get()

{

RegistryKey registry = Registry.CurrentUser;

RegistryKey key

= registry.

CreateSubKey("Server\_ConfigurationLena");

try

{

Organization\_Name

= key.GetValue("Organization\_Name")

.ToString();

doc\_Left\_Merge

= Convert.ToInt32(key

.GetValue("doc\_Left\_Merge").ToString());

doc\_Right\_Merg

= Convert.ToInt32(key

.GetValue("doc\_Right\_Merg").ToString());

doc\_Top\_Merge

= Convert.ToInt32(

key.GetValue("doc\_Top\_Merge")

.ToString());

doc\_Bottom\_Merge

= Convert.ToInt32(

key.GetValue("doc\_Bottom\_Merge")

.ToString());

}

catch

{

Organization\_Name = "Empty";

doc\_Left\_Merge = 0;

doc\_Right\_Merg = 0;

doc\_Top\_Merge = 0;

doc\_Bottom\_Merge = 0;

}

RegistryKey remote\_key;

try

{

Machine\_Name\_Get();

remote\_key = RegistryKey.OpenRemoteBaseKey(

RegistryHive.CurrentUser, Machine\_Name);

}

catch

{

}

}

public void Document\_Configuration\_set(string organization\_Name, Int32 Doc\_Left\_Merge,

Int32 Doc\_Right\_Merg, Int32 Doc\_Top\_Merge, Int32 Doc\_Bottom\_Merge)

{

RegistryKey registry = Registry.CurrentUser;

RegistryKey key = registry.CreateSubKey("Server\_ConfigurationLena");

key.SetValue("Organization\_Name", organization\_Name);

key.SetValue("doc\_Left\_Merge", Doc\_Left\_Merge);

key.SetValue("doc\_Right\_Merg", Doc\_Right\_Merg);

key.SetValue("doc\_Top\_Merge", Doc\_Top\_Merge);

key.SetValue("doc\_Bottom\_Merge", Doc\_Bottom\_Merge);

Document\_Configuration\_Get();

}

}

}

* + 1. Результат работы

В данном пункте продемонстрировать скриншоты работы «от и до» окна подключения к источнику данных.

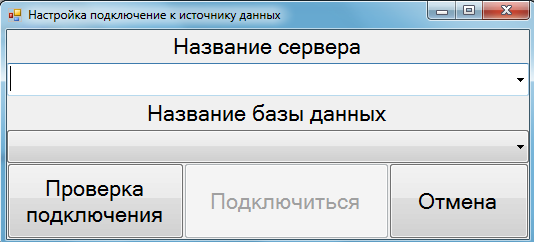


Рисунок 1 – 1 этап

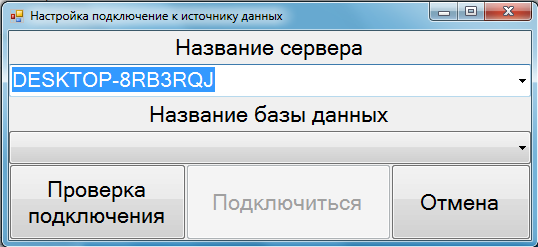


Рисунок 2 – 2 этап

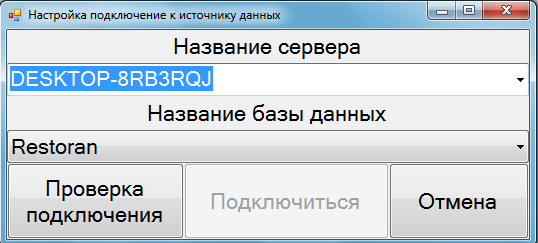


Рисунок 3 – 3 этап

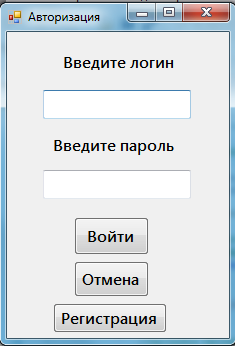


Рисунок 4 – открытие

* 1. Практическая работа № 31
     1. Описание окон приложения

В данном пункте в виде таблицы описать разрабатываемые окна справочных таблиц, с указанием названия и описанием тех таблиц, которые в них будут выводить.

Таблица 1 – Описание окон приложения

|  |  |
| --- | --- |
| Название окна | Функциональное описание окна |
| Заказ | Добавление, изменение, удаление, поиск, сортировка и фильтрация данных, а так же вывод информации из таблицы списка товаров и сотрудников |
| Список товаров | Добавление, изменение, удаление, поиск, сортировка и фильтрация данных, а также вывод информации из таблицы качества |
| Сотрудники | Добавление, изменение, удаление, поиск, сортировка и фильтрация данных, а также вывод информация из таблицы отделов |
| Качество | Добавление, изменение, удаление, поиск, сортировка и фильтрация данных |
| Отдел | Добавление, изменение, удаление, поиск, сортировка и фильтрация данных |

* + 1. Описание классов

В данном пункте в виде таблицы необходимо описать название классов, и назначение данного пространства в рамках выполняемой работы.

Таблица 2 – Описание классов

|  |  |
| --- | --- |
| Название элемента | Краткое описание назначения элемента |
| Document\_Сlass | Обработка данных из БД для создания документов |
| Program | Служит входной точкой приложения и хранит в себе глобальные переменные, свойства для всего проекта в целом |

* + 1. Описание визуальных компонентов

В данном пункте необходимо описать требования к визуальным компонентам, с помощью которых будут производиться манипулирования данных в приложении БД. Данный пункт является эталоном создания и реализации интерфейса, т.е. описывается обязательно внешний вид элементов, каких они будут размеров, их функциональное назначение (UI). Во второй таблице описать предполагаемые реакции элементов на взаимодействие с пользователями (UX).

Таблица 3 – UI описание

|  |  |
| --- | --- |
| Скриншот элемента | Описание свойств элемента |
|  | Все свойства стоят по умолчанию, кроме размера самого компонента и изменяется он в зависимости от размера окна. Служит для вывода данных из БД. |
|  | Label, стандартные свойства, служит для вывода текстовой информации из БД |
|  | Button, стандартные свойства. Изменяет свойства размера в зависимости от groupBox, служит для добавление или изменения данных в БД |
|  | TextBox, стандартные свойства, изменяет свойства размера в зависимости от groupBox, служит для внесения данных |

Таблица 4 – UX описание

|  |  |
| --- | --- |
| Скриншот элемента | Описание реакции элемента |
|  | Данные будут подсвечиваться при наведении |
|  | При изменении данных в БД, эти данные будут выводиться здесь |
|  | При наведении на кнопки компонента, происходит эффект падения, при нажатии кнопка выделится другим цветом |
|  | При наведении на область textBox курсор меняется, при нажатие textbox появиться pipe в строке |

* + 1. Код программы

public partial class Auth : Form

{

public Auth()

{

InitializeComponent();

}

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

{

string Login = tbLogin.Text;

string Password = tbPassword.Text;

var Proverka = new Table\_Class($"SELECT Otdel\_ID, ID\_Sotrudnika FROM dbo.Sotrudniki WHERE Sotrudnika\_Login = '{Login}' AND Sotrudnika\_Password = '{Password}'");

try

{

if (Proverka.table.Rows[0][0] != DBNull.Value)

{

Program.Dostup = Convert.ToInt32(Proverka.table.Rows[0][0].ToString());

Program.IDSotr = Convert.ToInt32(Proverka.table.Rows[0][1].ToString());

new MainPage().Show();

Visible = false;

ShowInTaskbar = false;

}

}

catch

{

MessageBox.Show("Не правильно введен логин или пароль!", "Ресторан", MessageBoxButtons.OK,

MessageBoxIcon.Warning);

}

}

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

{

Close();

}

private void button3\_Click(object sender, EventArgs e)

{

new Register().Show();

Visible = false;

ShowInTaskbar = false;

}

}

}

* + 1. Результат работы программы

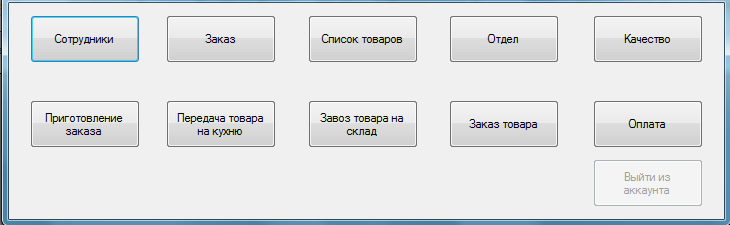


Рисунок 5 – наведение

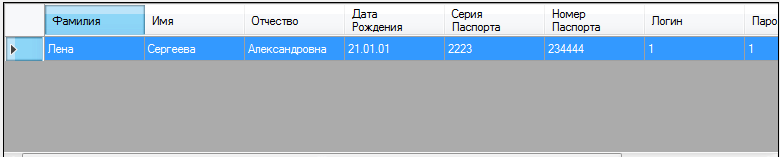


Рисунок 6 - выбор значения и “tabpage”

* 1. Практическая работа № 32
     1. Код программы
        1. Добавление данных

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

{

ArrayList arrayList = new ArrayList();

arrayList.Add(textBox1.Text);

arrayList.Add(textBox2.Text);

arrayList.Add(textBox3.Text);

arrayList.Add(textBox4.Text);

arrayList.Add(textBox5.Text);

arrayList.Add(textBox6.Text);

arrayList.Add(textBox7.Text);

arrayList.Add(textBox8.Text);

arrayList.Add(comboBox1.SelectedValue);

new Procedure\_Class().procedure\_Execution("Sotrudniki\_insert", arrayList);

DgFill();

}

* + - 1. Изменение данных

private void button3\_Click(object sender, EventArgs e)

{

ArrayList arrayList = new ArrayList();

arrayList.Add(sotrID);

arrayList.Add(textBox1.Text);

arrayList.Add(textBox2.Text);

arrayList.Add(textBox3.Text);

arrayList.Add(textBox4.Text);

arrayList.Add(textBox5.Text);

arrayList.Add(textBox6.Text);

arrayList.Add(textBox7.Text);

arrayList.Add(textBox8.Text);

arrayList.Add(comboBox1.SelectedValue);

new Procedure\_Class().procedure\_Execution("Sotrudniki\_update", arrayList);

DgFill();

}

* + - 1. Удаление данных (Физическое)

private void button4\_Click(object sender, EventArgs e)

{

ArrayList arrayList = new ArrayList();

arrayList.Add(sotrID);

new Procedure\_Class().procedure\_Execution("Sotrudniki\_delete", arrayList);

DgFill();

}

* + - 1. Удаление данных (Логическое)

public void ingDolgnost\_delete(Int32 ID\_Sotrudniki)

{

commandConfig("Sotrudniki\_delete");

command.Parameters.AddWithValue("@ID\_Sotrudniki", ID\_Sotrudniki);

DBConnection.connection.Open();

command.ExecuteNonQuery();

DBConnection.connection.Close();

}

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

{

DataRowView ID = (DataRowView)dgDolgnost.SelectedItems[0];

bool Log\_Del = Convert.ToBoolean(chbLogDel.IsChecked);

procedures.ingDolgnost\_update(Convert.ToInt32(ID["ID\_Sotrudniki"]), tbName\_Sotrudniki.Text.ToString(), Convert.ToInt32(tbOklad.Text.ToString()), Log\_Del);

* + - 1. Фильтрация данных

private void ChbFilter\_Checked(object sender, RoutedEventArgs e)

{

switch (chbFilter.IsChecked)

{

case (true):

btSearch.Content = "Фильтрация";

break;

case (false):

btSearch.Content = "Поиск";

dgFill(QR);

break;

}

}

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

{

switch (chbFilter.IsChecked)

{

case (true):

string newQR = QR + "where [Name\_Sotrudniki] like '%";

dgFill(newQR);

break;

case (false):

foreach (DataRowView dataRow in (DataView)dgDolgnost.ItemsSource)

{

if (dataRow.Row.ItemArray[1].ToString() == tbSearch.Text ||

dataRow.Row.ItemArray[2].ToString() == tbSearch.Text ||

dataRow.Row.ItemArray[3].ToString() == tbSearch.Text)

{

dgDolgnost.SelectedItem = dataRow;

}

}

break;

}

}

* + - 1. Поиск данных

for (int i = 0; i < dataGridView1.RowCount; i++)

{

dataGridView1.Rows[i].Selected = false;

for (int j = 0; j < dataGridView1.ColumnCount; j++)

if (dataGridView1.Rows[i].Cells[j].Value != null)

if (dataGridView1.Rows[i].Cells[j].Value.ToString().Contains(tbSearch.Text))

{

dataGridView1.Rows[i].Selected = true;

button6\_Click(sender, e);

break;

}

}

* + 1. Результат работы программы

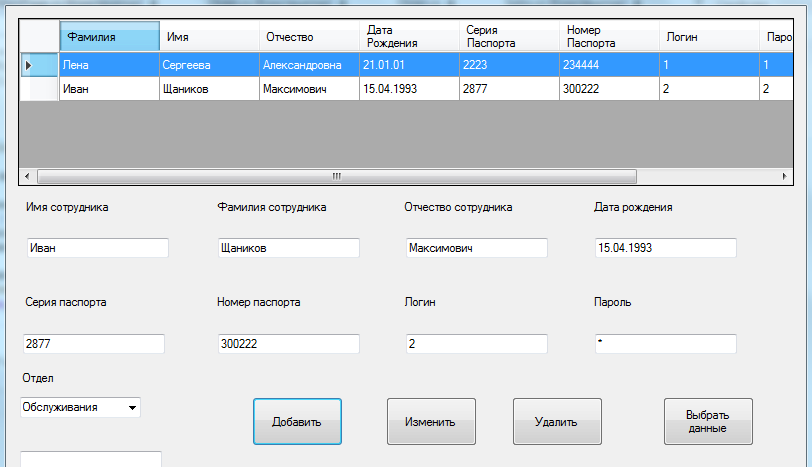


Рисунок 7 – добавление

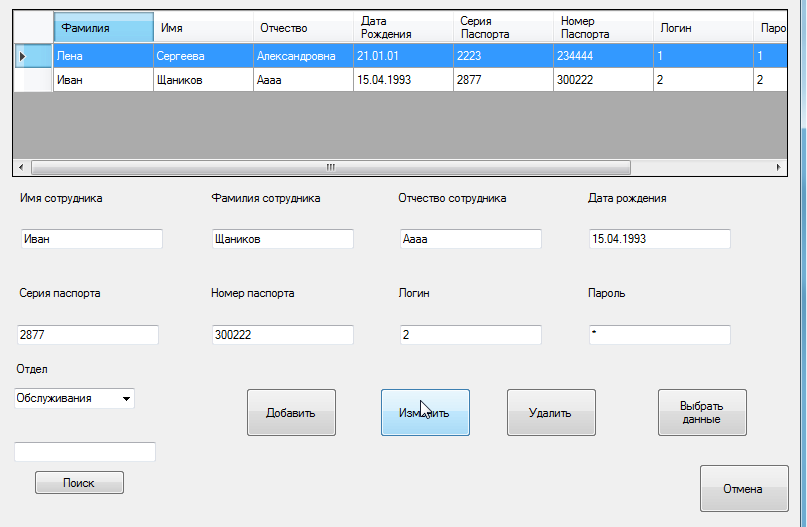


Рисунок 8 – изменение

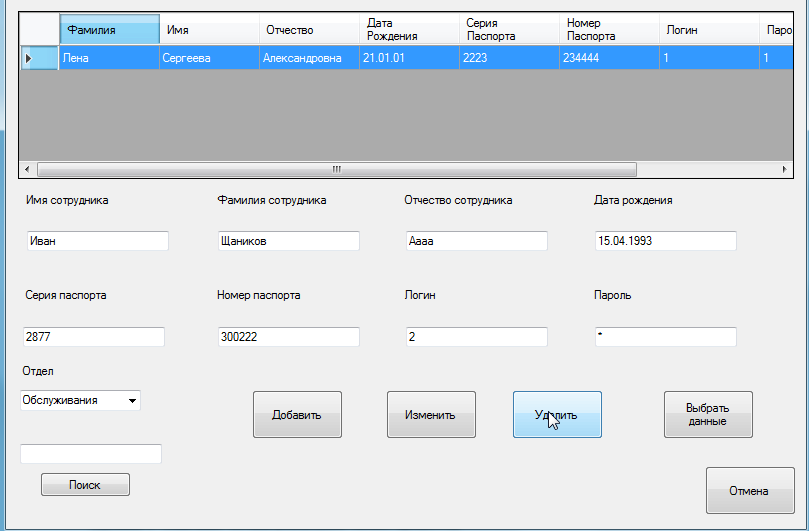


Рисунок 9 – удаление

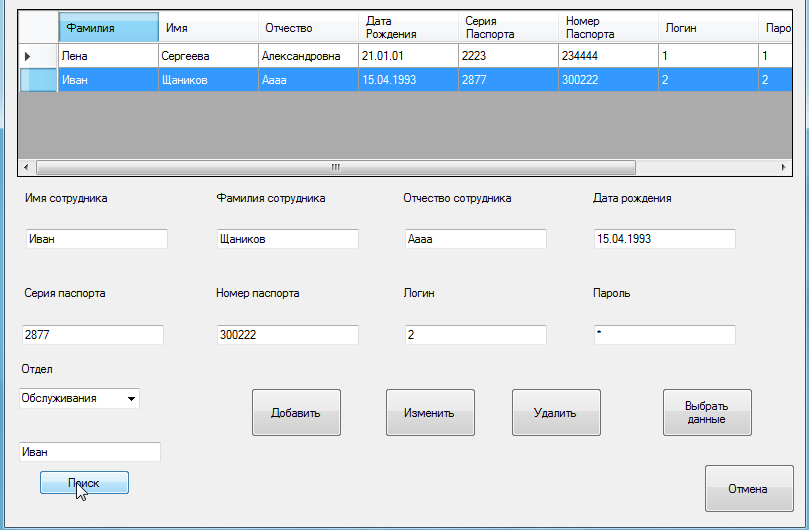


Рисунок 10 - поиск

* 1. Практическая работа № 33
     1. Описание блоков функций бизнес-процессов

В данном пункте необходимо описать порядок выполнения бизнес-процессов в приложении на основе бизнес-процессов DFD, в виде нумерованного списка на против каждый блок процесса.

Таблица 1- Описание логики реализации бизнес процессов

|  |  |
| --- | --- |
| Блок DFD | Порядок выполнения |
| Приготовление заказа | Забираются данные из таких сущностей как, заказ и список товаров, сумма заказа и номер стола |
| Передача товара на кухню | Забираются данные из сущности список товаров и выводится список товаров |
| Завоз товара на склад | Забираются данные из сущности список товаров и качества |
| Заказ товара | Забираются данные из сущности заказ и список товаров, откуда выводятся списки товара на складе и информация о клиенте и его заказе |
| Оплата | Забираются данные из сущностей заказ, откуда берется номер заказа и стола, также из сотрудников, где показывается какой сотрудник на оплате, и также данные из списков товара, а именно название товара |

* + 1. Код программы

public partial class PrigotovlenieZakaza : Form

{

public PrigotovlenieZakaza()

{

InitializeComponent();

}

private void PrigotovlenieZakaza\_Load(object sender, EventArgs e)

{

dgTovary.SelectionMode = DataGridViewSelectionMode.FullRowSelect;

DgFill();

CbFill();

}

private void DgFill()

{

var table\_class = new Table\_Class("SELECT [ID\_Tovara],[Name\_of\_Tovar] as \"ИмяТовара\",[Number\_of\_Tovar] as \"НомерТовара\"" +

",[Date\_Postavka] as \"ДатаПоставкиТовара\",[Srok\_Godnosti] as \"СрокГодности\", " +

"[Nazv\_Kachestvo] as \"НазваниеКачества\", [ID\_Kachestvo] FROM [Restoran].[dbo].[Spisok\_Tovarov] inner join" +

" [Restoran].[dbo].[Kachestvo] on [Restoran].[dbo].[Spisok\_Tovarov].[Kachestvo\_ID] = [Restoran].[dbo].[Kachestvo].[ID\_Kachestvo]");

dgTovary.DataSource = table\_class.table.DefaultView;

dgTovary.Columns[0].Visible = false;

dgTovary.Columns[6].Visible = false;

}

private void CbFill()

{

Table\_Class tableClass2 = new Table\_Class("SELECT ID\_Tovara, Name\_of\_Tovar FROM dbo.Spisok\_Tovarov");

comboBox2.DataSource = tableClass2.table.DefaultView;

comboBox2.ValueMember = "ID\_Tovara";

comboBox2.DisplayMember = "Name\_of\_Tovar";

}

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

{

ArrayList arrayList = new ArrayList();

arrayList.Add(textBox1.Text);

arrayList.Add(textBox2.Text);

arrayList.Add(comboBox2.SelectedValue);

arrayList.Add(Program.IDSotr);

new Procedure\_Class().procedure\_Execution("Zakaz\_insert", arrayList);

DgFill();

MessageBox.Show("Заказ успешно добавлен!", "Ресторан", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

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

{

new MainPage().Show();

Visible = false;

ShowInTaskbar = false;

}

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

{

for (int i = 0; i < dgTovary.RowCount; i++)

{

dgTovary.Rows[i].Selected = false;

for (int j = 0; j < dgTovary.ColumnCount; j++)

if (dgTovary.Rows[i].Cells[j].Value != null)

if (dgTovary.Rows[i].Cells[j].Value.ToString().Contains(tbSearch.Text))

{

dgTovary.Rows[i].Selected = true;

break;

}

}

}

}

}

public partial class PeredachaTovaraKuxnu : Form

{

private Table\_Class tableClass;

public PeredachaTovaraKuxnu()

{

InitializeComponent();

}

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

{

new MainPage().Show();

Visible = false;

ShowInTaskbar = false;

}

private void PeredachaTovaraKuxnu\_Load(object sender, EventArgs e)

{

dataGridView1.SelectionMode = DataGridViewSelectionMode.FullRowSelect;

DgFill();

}

private void DgFill()

{

tableClass = new Table\_Class("SELECT Name\_of\_Tovar as \"Имя Товара\", Number\_of\_Tovar as \"Номер Товара\", Srok\_Godnosti as \"Срок Годности\" " +

$"FROM dbo.Spisok\_Tovarov WHERE Spisok\_Tovarov.Date\_Postavka = '{DateTime.Now.ToShortDateString()}'");

dataGridView1.DataSource = tableClass.table.DefaultView;

}

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

{

Document\_Class documentClass = new Document\_Class();

documentClass.Document\_Create(Document\_Class.Document\_Type.Statistic, Document\_Class.Document\_Format.Word, "Товары на передачу на кухню", tableClass.table);

documentClass.Document\_Create(Document\_Class.Document\_Type.Statistic, Document\_Class.Document\_Format.Excel, "Товары на передачу на кухню", tableClass.table);

documentClass.Document\_Create(Document\_Class.Document\_Type.Statistic, Document\_Class.Document\_Format.PDF, "Товары на передачу на кухню", tableClass.table);

}

}

}

public partial class ZavozTovaraSklad : Form

{

public ZavozTovaraSklad()

{

InitializeComponent();

}

private void ZavozTovaraSklad\_Load(object sender, EventArgs e)

{

dataGridView1.SelectionMode = DataGridViewSelectionMode.FullRowSelect;

DgFill();

CbFill();

}

private void DgFill()

{

var table\_class = new Table\_Class("SELECT [ID\_Tovara],[Name\_of\_Tovar] as \"Имя Товара\",[Number\_of\_Tovar] as \"Номер Товара\"" +

",[Date\_Postavka] as \"Дата Поставки Товара\",[Srok\_Godnosti] as \"СрокГодности\", " +

"[Nazv\_Kachestvo] as \"Название Качества\", [ID\_Kachestvo] FROM [Restoran].[dbo].[Spisok\_Tovarov] inner join" +

" [Restoran].[dbo].[Kachestvo] on [Restoran].[dbo].[Spisok\_Tovarov].[Kachestvo\_ID] = [Restoran].[dbo].[Kachestvo].[ID\_Kachestvo]");

dataGridView1.DataSource = table\_class.table.DefaultView;

dataGridView1.Columns[0].Visible = false;

dataGridView1.Columns[6].Visible = false;

}

private void CbFill()

{

Table\_Class tableClass = new Table\_Class("SELECT [ID\_Kachestvo],[Nazv\_Kachestvo] FROM [Restoran].[dbo].[Kachestvo]");

comboBox1.DataSource = tableClass.table.DefaultView;

comboBox1.ValueMember = "ID\_Kachestvo";

comboBox1.DisplayMember = "Nazv\_Kachestvo";

}

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

{

new MainPage().Show();

Visible = false;

ShowInTaskbar = false;

}

private void button5\_Click(object sender, EventArgs e)

{

ArrayList arrayList = new ArrayList();

arrayList.Add(textBox1.Text);

arrayList.Add(textBox2.Text);

arrayList.Add(DateTime.Now.ToShortDateString());

arrayList.Add(textBox4.Text);

arrayList.Add(comboBox1.SelectedValue);

new Procedure\_Class().procedure\_Execution("Spisok\_Tovarov\_insert", arrayList);

DgFill();

}

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

{

for (int i = 0; i < dataGridView1.RowCount; i++)

{

dataGridView1.Rows[i].Selected = false;

for (int j = 0; j < dataGridView1.ColumnCount; j++)

if (dataGridView1.Rows[i].Cells[j].Value != null)

if (dataGridView1.Rows[i].Cells[j].Value.ToString().Contains(tbSearch.Text))

{

dataGridView1.Rows[i].Selected = true;

break;

}

}

}

} public partial class ZakazTovara : Form

{

public ZakazTovara()

{

InitializeComponent();

}

private void ZakazTovara\_Load(object sender, EventArgs e)

{

dataGridView1.SelectionMode = DataGridViewSelectionMode.FullRowSelect;

dataGridView2.SelectionMode = DataGridViewSelectionMode.FullRowSelect;

DgFill();

}

private void DgFill()

{

var table\_class1 = new Table\_Class("SELECT [ID\_Zakaza] as \"Код заказа\",[Summa\_Zakaza] as \"СуммаЗаказа\",[Number\_of\_Stol] as \"НомерСтола\"," +

" [Name\_Sotrudnika] as \"ФамилияСотрудника\", [Name\_of\_Tovar] as \"ИмяТовара\", [ID\_Tovara], [ID\_Sotrudnika]" +

" FROM [Restoran].[dbo].[Zakaz] inner join [Restoran].[dbo].[Sotrudniki] on" +

" [Restoran].[dbo].[Zakaz].[Sotrudnika\_ID] = [Restoran].[dbo].[Sotrudniki].[ID\_Sotrudnika] " +

"inner join [Restoran].[dbo].[Spisok\_Tovarov] on [Restoran].[dbo].[Zakaz].[Tovar\_ID] = [Restoran].[dbo].[Spisok\_Tovarov].[ID\_Tovara]");

dataGridView1.DataSource = table\_class1.table.DefaultView;

dataGridView1.Columns[5].Visible = false;

dataGridView1.Columns[6].Visible = false;

var table\_class = new Table\_Class("SELECT [ID\_Tovara],[Name\_of\_Tovar] as \"Имя Товара\",[Number\_of\_Tovar] as \"Номер Товара\"" +

",[Date\_Postavka] as \"Дата Поставки Товара\",[Srok\_Godnosti] as \"Срок Годности\", " +

"[Nazv\_Kachestvo] as \"Название Качества\", [ID\_Kachestvo] FROM [Restoran].[dbo].[Spisok\_Tovarov] inner join" +

" [Restoran].[dbo].[Kachestvo] on [Restoran].[dbo].[Spisok\_Tovarov].[Kachestvo\_ID] = [Restoran].[dbo].[Kachestvo].[ID\_Kachestvo]");

dataGridView2.DataSource = table\_class.table.DefaultView;

dataGridView2.Columns[0].Visible = false;

dataGridView2.Columns[6].Visible = false;

}

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

{

new MainPage().Show();

Visible = false;

ShowInTaskbar = false;

}

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

{

Program.GotovZakaz = Convert.ToInt32(numericUpDown1.Value);

new Oplata().Show();

Visible = false;

ShowInTaskbar = false;

}

}

}

}

public partial class Oplata : Form

{

private Table\_Class table\_class1;

public Oplata()

{

InitializeComponent();

}

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

{

new MainPage().Show();

Visible = false;

ShowInTaskbar = false;

}

private void Oplata\_Load(object sender, EventArgs e)

{

dataGridView1.SelectionMode = DataGridViewSelectionMode.FullRowSelect;

DgFill();

}

private void DgFill()

{

table\_class1 = new Table\_Class("SELECT [ID\_Zakaza] as \"Код заказа\",[Summa\_Zakaza] as \"СуммаЗаказа\",[Number\_of\_Stol] as \"НомерСтола\"," +

" [Name\_Sotrudnika] as \"ФамилияСотрудника\", [Name\_of\_Tovar] as \"ИмяТовара\", [ID\_Tovara], [ID\_Sotrudnika]" +

" FROM [Restoran].[dbo].[Zakaz] inner join [Restoran].[dbo].[Sotrudniki] on" +

" [Restoran].[dbo].[Zakaz].[Sotrudnika\_ID] = [Restoran].[dbo].[Sotrudniki].[ID\_Sotrudnika] " +

"inner join [Restoran].[dbo].[Spisok\_Tovarov] on [Restoran].[dbo].[Zakaz].[Tovar\_ID] = [Restoran].[dbo].[Spisok\_Tovarov].[ID\_Tovara]" +

$" WHERE [ID\_Zakaza] = {Program.GotovZakaz}");

dataGridView1.DataSource = table\_class1.table.DefaultView;

dataGridView1.Columns[5].Visible = false;

dataGridView1.Columns[6].Visible = false;

}

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

{

Document\_Class documentClass = new Document\_Class();

documentClass.Document\_Create(Document\_Class.Document\_Type.Report, Document\_Class.Document\_Format.Word, "Заказ на оплату", table\_class1.table);

documentClass.Document\_Create(Document\_Class.Document\_Type.Report, Document\_Class.Document\_Format.Excel, "Заказ на оплату", table\_class1.table);

documentClass.Document\_Create(Document\_Class.Document\_Type.Report, Document\_Class.Document\_Format.PDF, "Заказ на оплату", table\_class1.table);

}

}

* + 1. Результат работы программы

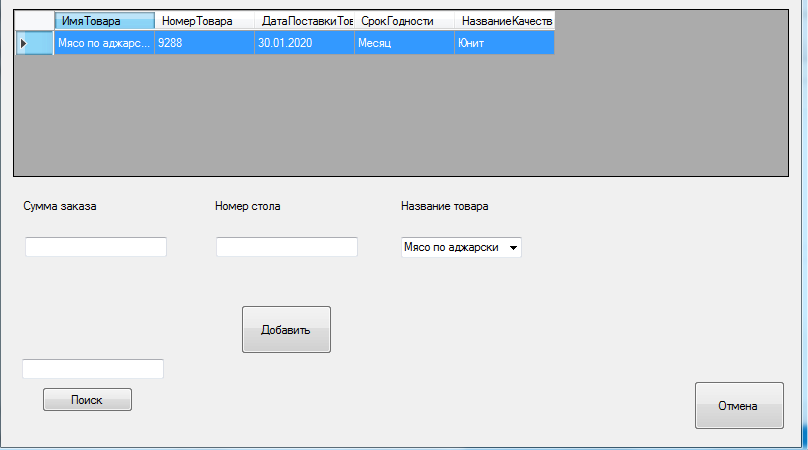


Рисунок 11 – приготовление заказа

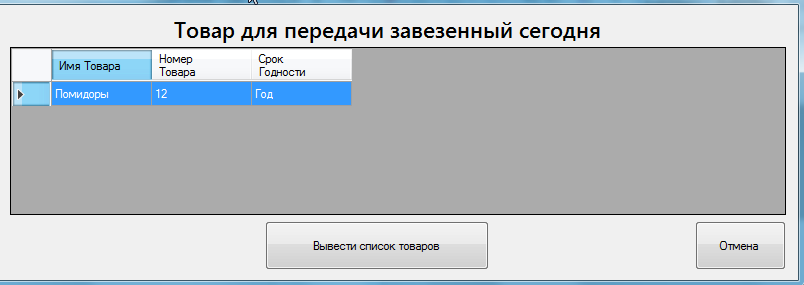


Рисунок 12 – передача товара на кухню

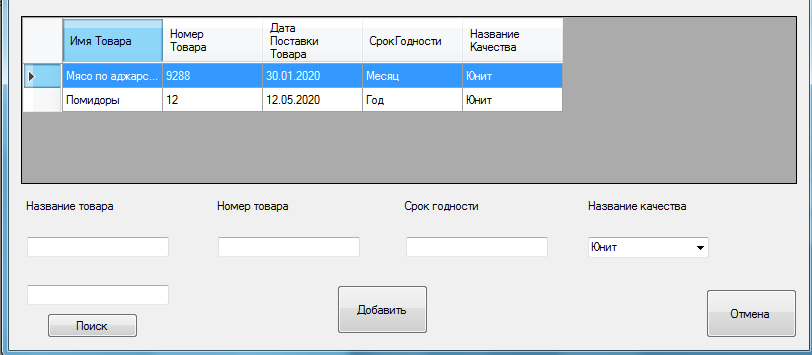


Рисунок 13 – завоз товара на склад

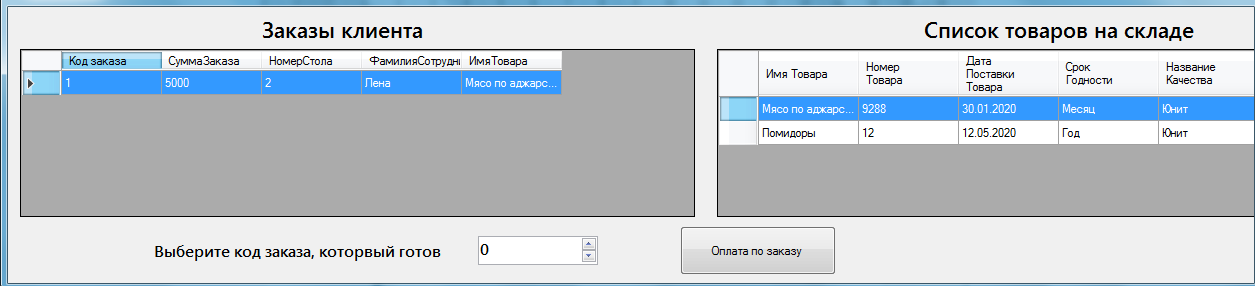


Рисунок 14 - заказ товара

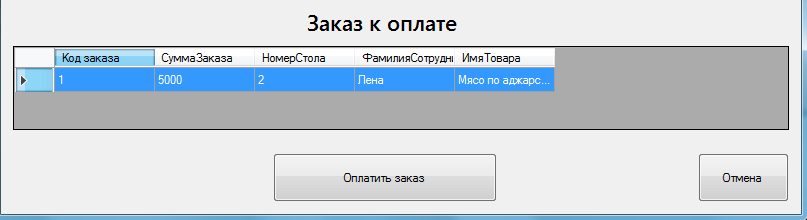


Рисунок 15 - оплата

* 1. Практическая работа № 34
     1. Требования к экспорту

В данном пункте в виде таблицы описать требования к реализации выходных документов на основе модели EPC.

Таблица 1 – Описание выходных документов

|  |  |
| --- | --- |
| Скриншот нотации EPC | Требования к содержанию выходного документа |
|  | Должны быть записи, в сущности. Форматы такие как .doc, .pdf, .xls. |
|  | Должны быть записи, в сущности. Форматы такие как .doc, .pdf, .xls. |

* + 1. Код программы

class Document\_Class

{

/// <summary>

/// Список доступных видов отчётов

/// </summary>

internal enum Document\_Type

{

Report, Statistic

}

/// <summary>

/// Итоговый выходной формат отчёта

/// </summary>

internal enum Document\_Format

{

Word, Excel, PDF

}

/// <summary>

/// Метод создания и сохранения документов

/// в форматах Microsoft Word (doc, PDF),

/// Excel (exls)

/// </summary>

/// <param name="type">Тип создаваемого документа

/// отчёт или статистика</param>

/// <param name="format">Формат сохранения

/// документ или таблица</param>

/// <param name="name">Название документа</param>

/// <param name="table">Входная таблица с данными</param>

public void Document\_Create(Document\_Type type,

Document\_Format format, string name,

DataTable table)

{

//Получение данных о конфигурации документа

Configuration\_class configuration\_Class

= new Configuration\_class();

configuration\_Class.Document\_Configuration\_Get();

//Проверка на пустоту названия

switch (name != "" || name != null)

{

case true:

//Выбор формата либо Word либо Excel

switch (format)

{

case Document\_Format.Word:

//Запуск процесса в дистпечере задач

word.Application application

= new word.Application();

//создание документа в процессе

word.Document document

//Присвоение документа процессу, Visible: true

//возможность редактирования документа

= application.Documents.Add(Visible: true);

try

{

//Объявление дипапазона для формирования текста

word.Range range = document.Range(0, 0);

//89Настройка отступов в документе

document.Sections.PageSetup.LeftMargin

= application.CentimetersToPoints(

(float)Configuration\_class.

doc\_Left\_Merge);

document.Sections.PageSetup.TopMargin

= application.CentimetersToPoints(

(float)Configuration\_class.

doc\_Top\_Merge);

document.Sections.PageSetup.RightMargin

= application.

CentimetersToPoints((float)

Configuration\_class.doc\_Right\_Merg);

document.Sections.PageSetup.BottomMargin

= application.CentimetersToPoints(

(float)Configuration\_class.

doc\_Bottom\_Merge);

//Присвоение текстового знеачения в дипазон

range.Text =

"Ресторан";

//Настройка выравнивания текста

range.ParagraphFormat.Alignment =

word.WdParagraphAlignment.

wdAlignParagraphCenter;

//Настройка интервала после абзаца

range.ParagraphFormat.SpaceAfter = 1;

//Настройка интервала перед абзаца

range.ParagraphFormat.SpaceBefore = 1;

//Настройка межстрочного интервала

range.ParagraphFormat.LineSpacingRule

= word.WdLineSpacing.wdLineSpaceSingle;

//Настройка названия шрифта

range.Font.Name = "Times New Roman";

//Настройка размера шрифта

range.Font.Size = 12;

//Добавление параграфов

document.Paragraphs.Add();//В конце текста

document.Paragraphs.Add();//Свободный

document.Paragraphs.Add();//Для будущего текста

//Параграф для названия документа

word.Paragraph Document\_Name

= document.Paragraphs.Add();

//Настройка параграфа через свойство диапазона

Document\_Name.Format.Alignment

= word.WdParagraphAlignment.wdAlignParagraphCenter;

Document\_Name.Range.Font.Name = "Times New Roman";

Document\_Name.Range.Font.Size = 16;

//Проверка на тип документа, отчёт или статистика

switch (type)

{

case Document\_Type.Report:

Document\_Name.Range.Text = "ОТЧЁТ";

break;

case Document\_Type.Statistic:

Document\_Name.Range.Text = "СТАТИСТИЧЕСКИЙ ОТЧЁТ";

break;

}

document.Paragraphs.Add();

document.Paragraphs.Add();

document.Paragraphs.Add();

word.Paragraph statparg = document.Paragraphs.Add();

//Создание области таблицы в документе

word.Table stat\_table

//Добавление таблицы в область документа

//Указывается параграф в котором документ создан

//Количество строк и столбцов

= document.Tables.Add(statparg.Range,

table.Rows.Count, table.Columns.Count);

//Настройка границ таблицы внутренние

stat\_table.Borders.InsideLineStyle

= word.WdLineStyle.wdLineStyleSingle;

//Настройка границ таблицы внешние

stat\_table.Borders.OutsideLineStyle

= word.WdLineStyle.wdLineStyleSingle;

//Выравнивание текста внутри ячеек по ширине

stat\_table.Rows.Alignment

= word.WdRowAlignment.wdAlignRowCenter;

//Выравнивание текста внутри ячеек по высоте

stat\_table.Range.Cells.VerticalAlignment =

word.WdCellVerticalAlignment.wdCellAlignVerticalCenter;

stat\_table.Range.Font.Size = 11;

stat\_table.Range.Font.Name = "Times New Roman";

//Индексация столбцов и строк в Word начинается с 1,1

for (int row = 1; row <= table.Rows.Count; row++)

for (int col = 1; col <= table.Columns.Count; col++)

{

stat\_table.Cell(row, col).Range.Text

= table.Rows[row - 1][col - 1].ToString();

}

document.Paragraphs.Add();

document.Paragraphs.Add();

//Парадграф с фиксациейц даты создания документа

word.Paragraph Footparg = document.Paragraphs.Add();

Footparg.Range.Text =

string.Format("Дата создания \t\t\t{0}",

DateTime.Now.ToString("dd.MM.yyyy"));

}

catch (Exception ex)

{

System.Windows.Forms.MessageBox.Show(ex.Message);

}

finally

{

switch (format)

{

case Document\_Format.Word:

//Сохранение документа с названием из метода,

//и в формате doc

document.SaveAs2(string.Format("{0}\\{1}", Environment.CurrentDirectory, name),

word.WdSaveFormat.wdFormatDocument);

break;

case Document\_Format.PDF:

//Сохранение документа в формате PDF

document.SaveAs2(string.Format("{0}\\{1}", Environment.CurrentDirectory, name),

word.WdSaveFormat.wdFormatPDF);

break;

}

//Закрываем документ

document.Close();

//Выходим из процесса с его закрытием

application.Quit();

}

break;

case Document\_Format.Excel:

//Создание процесса Excel

excel.Application application\_ex

= new excel.Application();

//Создание книги

excel.Workbook workbook

= application\_ex.Workbooks.Add();

//Создание страницы

excel.Worksheet worksheet

= (excel.Worksheet)workbook.ActiveSheet;

try

{

switch (type)

{

case Document\_Type.Report:

//Название страницы

worksheet.Name = "Отчёт";

for (int row = 0; row < table.Rows.Count; row++)

for (int col = 0; col < table.Columns.Count; col++)

{

//ЗАнесение данных в ячейку

worksheet.Cells[row + 1][col + 1]

= table.Rows[row][col].ToString();

}

//Указание диапазона работы с ячеёками листа

excel.Range border

//Начало диапазона

= worksheet.Range[worksheet.Cells[1, 1],

//Динамический конец диапазона в зависимости от

//выдодимых данных

worksheet.Cells[table.Rows.Count + 1]

[table.Columns.Count + 1]];

//Стиль линий границ ячеек

border.Borders.LineStyle = excel.XlLineStyle.xlContinuous;

//Выравнивание во высоте

border.VerticalAlignment = excel.XlHAlign.xlHAlignCenter;

//Выравнивание по ширине

border.HorizontalAlignment = excel.XlHAlign.xlHAlignCenter;

//Внесение даты создания документа

worksheet.Cells[table.Rows.Count + 3][2]

= string.Format("Дата создания {0}",

DateTime.Now.ToString());

//Объединение ячеек

worksheet.Range[worksheet.Cells[table.Rows.Count + 3, 2],

worksheet.Cells[table.Rows.Count + 2,

table.Columns.Count + 2]].Merge();

break;

case Document\_Type.Statistic:

worksheet.Name = "Статистический отчёт";

for (int row = 0; row < table.Rows.Count; row++)

for (int col = 0; col < table.Columns.Count; col++)

{

worksheet.Cells[row + 1][col + 1]

= table.Rows[row][col].ToString();

}

excel.Range border1

= worksheet.Range[worksheet.Cells[1, 1],

worksheet.Cells[table.Rows.Count + 1]

[table.Columns.Count + 1]];

border1.Borders.LineStyle

= excel.XlLineStyle.xlContinuous;

border1.VerticalAlignment

= excel.XlHAlign.xlHAlignCenter;

border1.HorizontalAlignment

= excel.XlHAlign.xlHAlignCenter;

worksheet.Cells[table.Rows.Count + 3][2]

= string.Format("Дата создания {0}",

DateTime.Now.ToString());

worksheet.Range[worksheet.Cells[table.Rows.Count + 3, 2],

worksheet.Cells[table.Rows.Count + 2,

table.Columns.Count + 2]].Merge();

//Класс области графиков

excel.ChartObjects chartObjects

= (excel.ChartObjects)worksheet.ChartObjects(

Type.Missing);

//Область размещения графиков: отступы слева сверху,

//размер ширина и высота

excel.ChartObject chartObject

= chartObjects.Add(300, 50, 250, 250);

//Объявление области графика

excel.Chart chart = chartObject.Chart;

//Объявление колекции построений графиков

excel.SeriesCollection seriesCollection

= (excel.SeriesCollection)chart.SeriesCollection(

Type.Missing);

//Объявление посторения графика

excel.Series series = seriesCollection.NewSeries();

//Тип графика

chart.ChartType = excel.XlChartType.xl3DColumn;

//Диапазон значений по оси X

series.XValues =

worksheet.get\_Range("B2", "B" + table.Rows.Count + 1);

//Диапазон значений по оси Y

series.Values =

worksheet.get\_Range("C2", "C" + table.Rows.Count + 1);

break;

}

}

catch

{

}

finally

{

//Сохранение книги

workbook.SaveAs(string.Format("{0}\\{1}", Environment.CurrentDirectory, name), application\_ex.DefaultSaveFormat);

//Закрытие книги

workbook.Close();

//Завершение процесса

application\_ex.Quit();

}

break;

}

break;

case false:

System.Windows.Forms.MessageBox.Show

("Введите название документп");

break;

}

}

}

}

* + 1. Результат работы программы

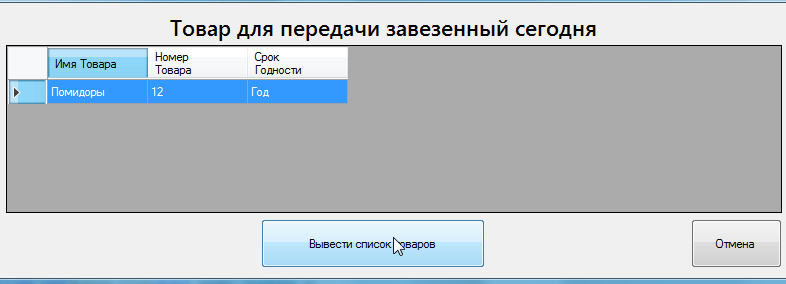


Рисунок 16 - начало

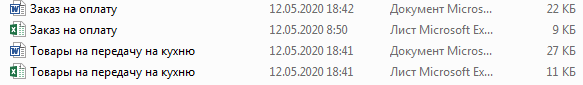


Рисунок 17 – документы

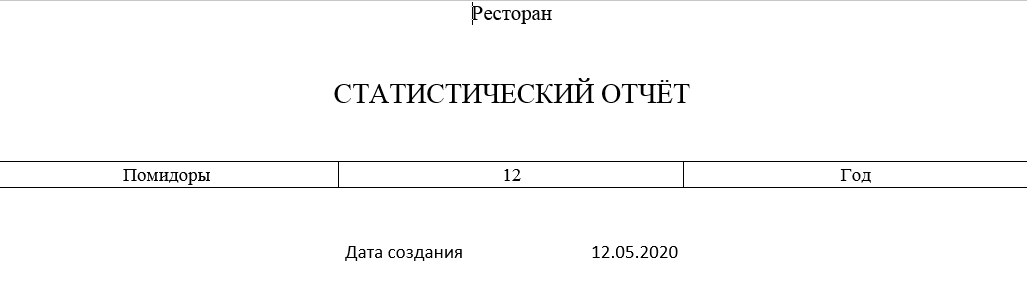


Рисунок 18 - результат

* 1. Практическая работа № 35
     1. Описание принципа аутентификации
        1. Авторизация

|  |  |
| --- | --- |
| Название роли | Функции роли |
| Администратор | * Заказ * Список товаров * Качество * Приготовление заказа * Передача товара на кухню * Завоз товара на склад * Оплата * Сотрудники * Заказ товара |
| Гость | * Оплата |
| Обслуживание | * Список товаров * Оплата * Заказ товара * Качество |
| Погрузки товара | * Заказ * Список товаров * Качество * Приготовление заказа * Передача товара на кухню * Завоз товара на склад |
| Грузчик | * Завоз товара на склад |

* + - 1. Регистрация

Требования для регистрации: нужно иметь доступ к ИС и знать к какой БД подключаться, если все это есть, то при регистрации выдаются минимальные права для пользователя.

* + 1. Код программы

public partial class Auth : Form

{

public Auth()

{

InitializeComponent();

}

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

{

string Login = tbLogin.Text;

string Password = tbPassword.Text;

var Proverka = new Table\_Class($"SELECT Otdel\_ID, ID\_Sotrudnika FROM dbo.Sotrudniki WHERE Sotrudnika\_Login = '{Login}' AND Sotrudnika\_Password = '{Password}'");

try

{

if (Proverka.table.Rows[0][0] != DBNull.Value)

{

Program.Dostup = Convert.ToInt32(Proverka.table.Rows[0][0].ToString());

Program.IDSotr = Convert.ToInt32(Proverka.table.Rows[0][1].ToString());

new MainPage().Show();

Visible = false;

ShowInTaskbar = false;

}

}

catch

{

MessageBox.Show("Не правильно введен логин или пароль!", "Ресторан", MessageBoxButtons.OK,

MessageBoxIcon.Warning);

}

}

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

{

Close();

}

private void button3\_Click(object sender, EventArgs e)

{

new Register().Show();

Visible = false;

ShowInTaskbar = false;

}

}

} public partial class Register : Form

{

public Register()

{

InitializeComponent();

}

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

{

Close();

}

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

{

ArrayList arrayList = new ArrayList();

arrayList.Add(textBox1.Text);

arrayList.Add(textBox2.Text);

arrayList.Add(textBox3.Text);

arrayList.Add(textBox4.Text);

arrayList.Add(textBox5.Text);

arrayList.Add(textBox6.Text);

arrayList.Add(textBox7.Text);

arrayList.Add(textBox8.Text);

arrayList.Add(2);

new Procedure\_Class().procedure\_Execution("Sotrudniki\_insert", arrayList);

MessageBox.Show("Вы успешно зарегистрированны!", "Ресторан", MessageBoxButtons.OK,

MessageBoxIcon.Information);

new Auth().Show();

Visible = false;

ShowInTaskbar = false;

}

}

}

* + 1. Результат работы программы

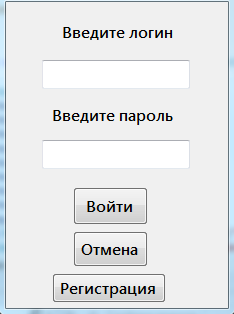


Рисунок 19 – авторизация начало

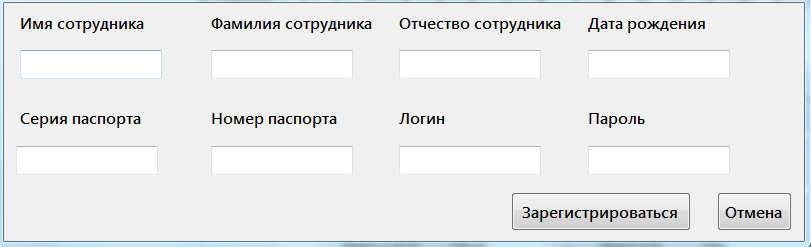


Рисунок 20 – регистрация

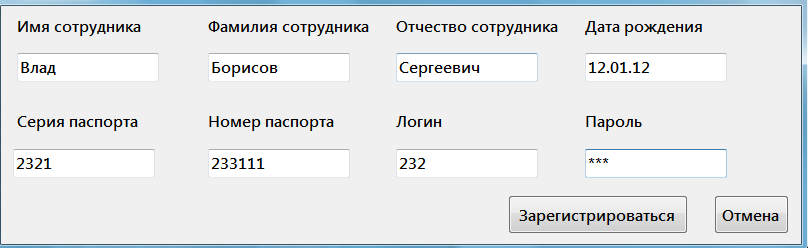


Рисунок 21 – регистрация после

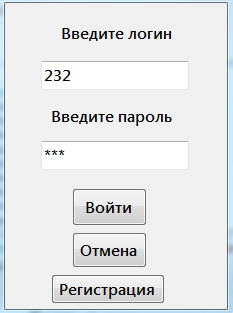


Рисунок 22 – после

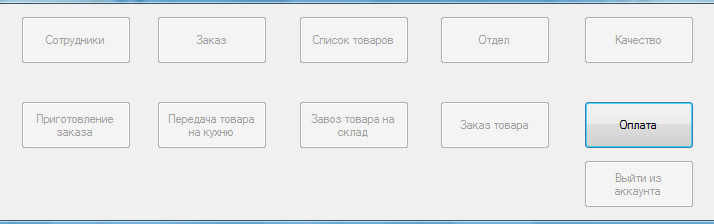


Рисунок 23 - результат

* 1. Практическая работа № 36
     1. Описание персонального репозитория
        1. Создание репозиория

В данном подпункте в виде скриншотов показать пошаговое создание удалённого репозитория к текущему проекту.

* + - 1. Настройка среды разработки

В данном пункте показать в виде скриншотов пошаговую настройку репозитория в среде разработки.

* + 1. Отправка проекта на репозиторий

В данном пункте показать в виде скриншотов механизм Push проекта на удалённый репозиторий, через локальный Commit.

* + 1. Скачивание проекта с репозитория

В данном пункте показать в виде скриншотов механизм Pull проекта из удалённого репозитория.

1. ВЫВОД

Разработали программное обеспечение с помощью языков программирования информационного контента. Программировал на встроенных алгоритмических языках.

1. СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Федорова Г. Н., Разработка, внедрение и адаптация программного обеспечения отраслевой направленности

2. Подбельский В. В., ПРОГРАММИРОВАНИЕ. БАЗОВЫЙ КУРС С#. Учебник для СПО, Юрайт, 2019, режим доступа

3. Гагарина Л. Г., Разработка и эксплуатация автоматизированных информационных систем, ИД ФОРУМ, 2018.

4. Казанский А.А., ПРОГРАММИРОВАНИЕ НА VISUAL C# 2013. Учебное пособие для СПО, Юрайт, 2019

5. Кувшинов Д.Р., ОСНОВЫ ПРОГРАММИРОВАНИЯ. Учебное пособие для СПО, Юрайт, 2019