**МИНОБРНАУКИ РОССИИ**

**Федеральное государственное бюджетное образовательное**

**учреждение высшего образования**

**«Чувашский государственный университет имени И.Н. Ульянова»**

**(ФГБОУ ВО «ЧГУ им. И.Н. Ульянова»)**

**Факультет информатики и вычислительной техники**

**Кафедра компьютерных технологий**

ОТЧЕТ

ОБ УЧЕБНОЙ ПРАКТИКЕ

(ОЗНАКОМИТЕЛЬНОЙ ПРАКТИКЕ)

на базе кафедры компьютерных технологий

(наименование профильной организации/ структурного подразделения университета)

|  |  |  |
| --- | --- | --- |
| Обучающийся 1 курса, направление подготовки «Программная инженерия» | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Киселев М.А. |
|  | подпись, дата | ФИО |
|  |  |  |
|  |  |  |
| Руководитель,  доцент кафедры  компьютерных технологий,  к. пед. н., доцент | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ подпись, дата | Давыдова О.В. |
| уч. степень, уч. звание | ФИО |
|  |  |  |
|  |  |  |
| Заведующий кафедрой  компьютерных технологий,  д.пед. н., профессор | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ подпись, дата | Лавина Т.А. |
| уч. степень, уч. звание | ФИО |
|  |  |  |

Чебоксары 2025

**Содержание**

[Введение 3](#_Toc206263765)

[1. Проектирование и разработка базы данных в СУБД MS Access 4](#_Toc206263766)

[1.1 Постановка задачи 4](#_Toc206263767)

[1.2 Разработка таблиц базы данных 4](#_Toc206263768)

[2. Разработка приложения Windows Form ИС-системы 7](#_Toc206263769)

[2.1 Подключение БД Access к приложению Visual Studio 7](#_Toc206263770)

[2.2 Разработка интерфейса приложения 8](#_Toc206263771)

[2.3 Работа с базой данных из DataGridView. Select, Insert, Update, Delete 10](#_Toc206263772)

[Заключение 14](#_Toc206263773)

[Список использованных источников 15](#_Toc206263774)

[Приложение 16](#_Toc206263775)

# **Введение**

Целями прохождения учебной практики являются:

* получение первичных профессиональных умений и навыков;
* закрепление, расширение и углубление теоретических и практических знаний умений и навыков, полученных обучающимися ранее при изучении дисциплин (модулей) учебного плана.

Задачи учебной практики:

* применение естественнонаучных и общеинженерных знаний, методов математического анализа и моделирования, теоретического и экспериментального исследования в профессиональной деятельности классификация информационных технологий, в том числе отечественного производства, по назначению и характеру использования, по степени охвата задач управления;
* применение информационных технологий, в том числе отечественного производства, при решении теоретических и прикладных задач профессиональной деятельности;
* рациональный поиск информации в соответствии с потребностями в глобальных информационных сетях с использованием современных поисковых систем;
* применение информации, полученной из глобальных информационных сетей, с соблюдением законодательства в области информации, информационных технологий, защиты информации и авторского права;
* применение антивирусных программных средств и других методов защиты информации в профессиональной деятельности;
* формализация и разработка алгоритмов для поставленных задач;
* разработка программного кода с использованием языков программирования;
* оформление программного кода в соответствии с установленными требованиями.
* проверка и отладка программного кода;
* применение технологий программирования.

# **Проектирование и разработка базы данных в СУБД MS Access**

## **Постановка задачи**

Разработать информационно справочную систему для работы с базой данных.

Входные данные:

База данных состоит из записей. (Номер склада, тип складируемых товаров, адрес, количество мест, код ответственного сотрудника в отдельном файле справочная информация об ответственном сотруднике)

Обеспечить:  
- просмотр содержимого базы данных;  
- редактирование базы данных:   
а) удаление записи;

б) добавление записи;

в) редактирование записи;

г) сохранение данных в базу данных.  
- поиск данных по номеру склада;   
- сортировка данных по количеству мест;  
- получение справочной информации об ответственном сотруднике.

## **Разработка таблиц базы данных**

Для создания базы данных использовалась СУБД MS Access. Была создана база данных с именем «DatabaseP», где были созданы две таблицы: «worker» (см. рисунок 1) и «warehouse» (см. рисунок 2) для хранения информации о сотрудниках и складах соответственно. Полями таблицы worker являются: id (числового формата) для записи индивидуального номера сотрудника, w\_name (текстового формата) для записи имение сотрудника, birth\_date (формата даты и времени) для записи дня рождения сотрудника, phone\_number (текстового формата) для записи номера телефона сотрудника, hire\_date (формата даты и времени) для записи даты приема на работу, salary (числового формата) для записи заработной платы сотрудника. Полями таблицы warehouse являются: id (числового формата) для записи индивидуального номера склада, product\_type (текстового формата) для записи типа хранящегося продукта, address (текстового формата) для записи адреса склада, places\_count (числового формата) для записи количество места на складе, worker\_id (числовой формат) для записи прикрепленного сотрудника.

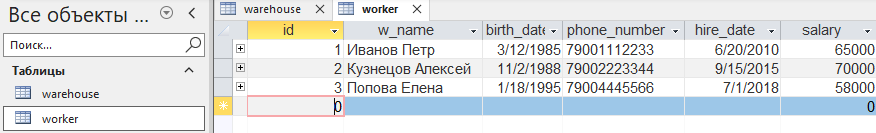


Рисунок 1 – Таблица «worker» в СУБД MS Access

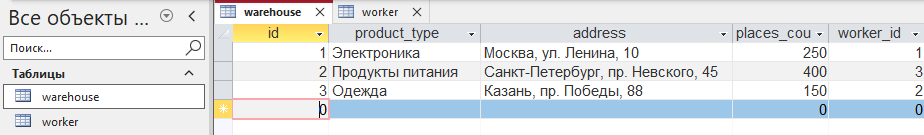


Рисунок 2 – Таблица «warehouse» в СУБД MS Access

Поле «id» в таблице worker связано с полем «worker\_id» в таблице warehouse по принципу «один-ко-многим» (см. рисунок 4), что обеспечивает непрерывную связь между таблицами worker и warehouse (см. рисунок 3). Это означает, что каждый склад из таблицы warehouse должен быть связан с определённым работником, информация о котором уже содержится в таблице worker. Попытка нарушить это правило приведёт к возникновению ошибки.

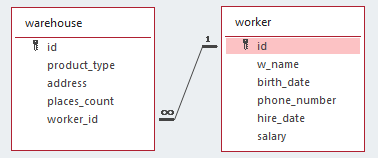


Рисунок 3 – Связь между таблицами «worker» и «warehouse» в СУБД MS Access

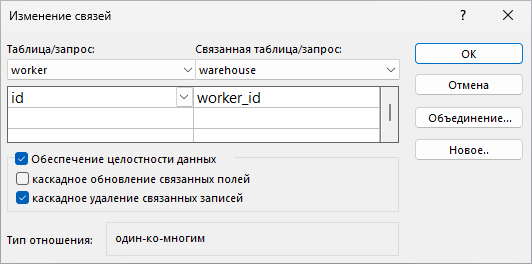


Рисунок 4 – Специфика связи таблиц «worker» и «warehouse»

# **Разработка приложения Windows Form ИС-системы**

## **2.1 Подключение БД Access к приложению Visual Studio**

Чтобы подключить БД Access к приложению Visual Studio был изначально выбран шаблон для работы с Windows Forms, использующими .NET Framework (см. рисунок 5).

**Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание**

Рисунок 5 – Шаблон, на основе которого будет создано приложение

В качестве инструмента был использован встроенный функционал Visual Studio ­­­­­– интерфейс «Источники данных», в рамках которого был добавлен новый источник данных, представляющий собой заранее подготовленную базу данных MS Access (см. рисунок 6).

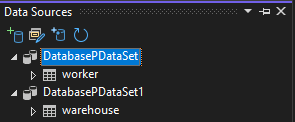


Рисунок 6 – Интерфейс «Источники данных»

Путь к базе данных был задан относительный для облегчения работы с информационной системой с любого компьютера. Предполагается, что файл базы данных будет находиться в той же директории, что и исполняемый exe-файл (см. рисунок 7).

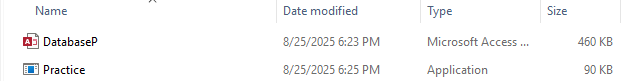


Рисунок 7 – Расположение файла базы данных в одной директории с exe-файлом

## **2.2 Разработка интерфейса приложения**

В процессе разработки интерфейса приложения было принято разделить приложение на три логических блока: главное меню, форма работы с базой сотрудников, форма работы с базой складов.

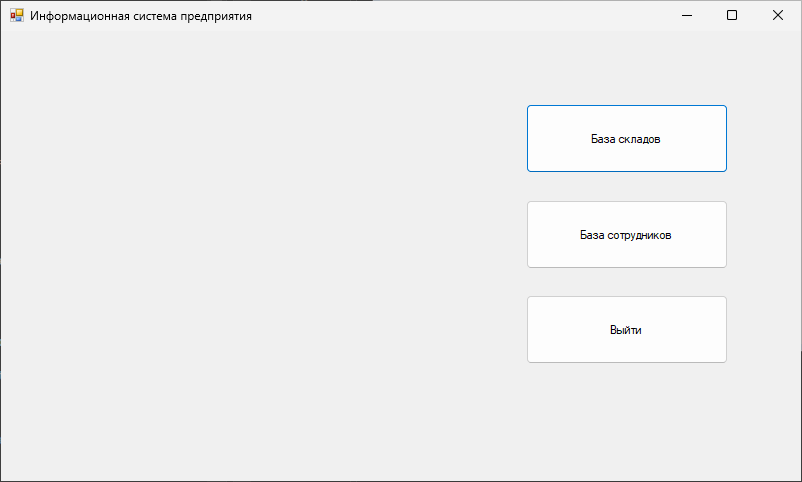


Рисунок 8 – Интерфейс Главного меню

При нажатии на кнопку «База складов» открывается форма для просмотра и управления складами (см. рисунок 9). Можно добавлять, удалять, изменять склады, а также сортировать по количеству мест (по возрастанию и убыванию). Кнопка «Выйти» возвращает в главное меню.

При нажатии на кнопку «База Работников» открывается форма для просмотра и управления бригадами (см. рисунок 10). Можно добавлять, удалять, изменять сотрудников. Кнопка «Выход» возвращает в главное меню.

При нажатии на кнопку «Выйти» в главном меню приложение завершает работу и закрывается.

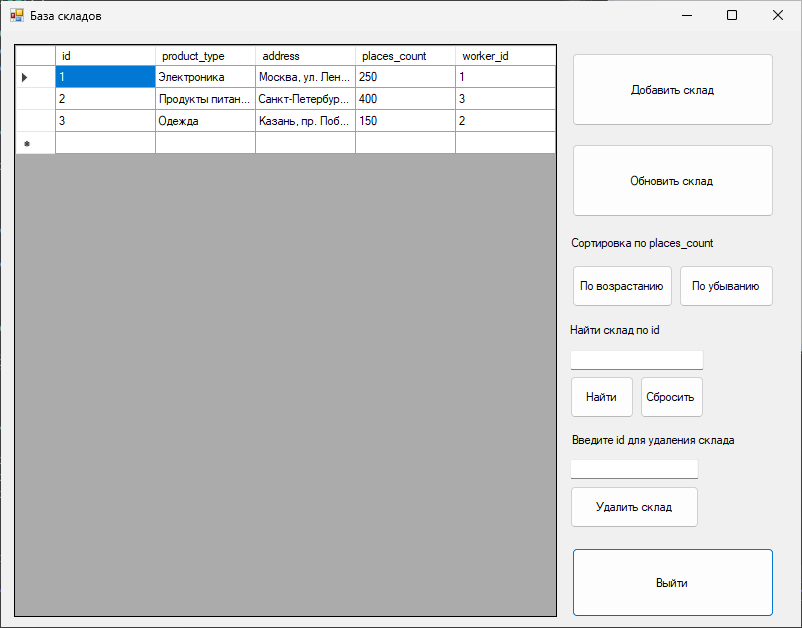


Рисунок 9 – Интерфейс «Базы складов»

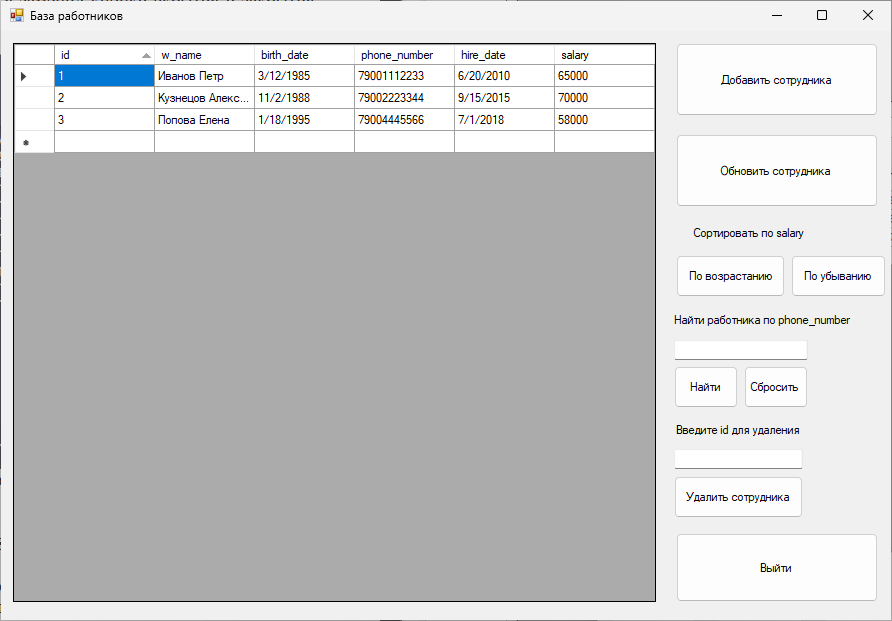


Рисунок 10 – Интерфейс «Базы сотрудников»

## **2.3 Работа с базой данных из DataGridView. Select, Insert, Update, Delete**

Метод INSERT INTO позволяет добавлять новые записи в таблицу. Важно отметить, что таблицы связаны и не получиться добавить в таблицу «warehouse» id сотрудника, которого нет в таблице «worker». Поэтому вначале стоит добавить нового сотрудника в таблицу «worker» (см. рисунок 11). Нужно нажать кнопку «Добавить сотрудника», заполнить все поля и нажать кнопку «Добавить» после этого добавиться новый сотрудник в таблицу «worker». После высветится окно (см. рисунок 12) об успешно добавлении сотрудника. При закрытии формы добавления сотрудника, информация автоматически обновиться до актуальной. Для обновления записей используется UPDATE.

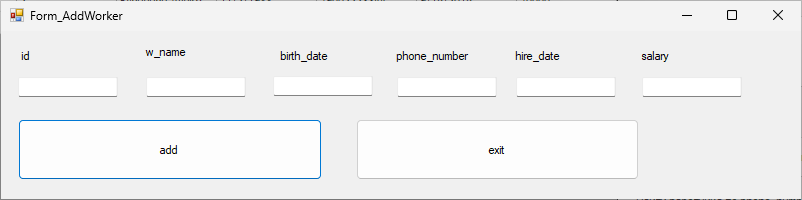


Рисунок 11 – Интерфейс добавления нового сотрудника

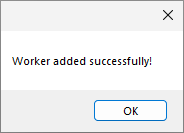


Рисунок 12 – Интерфейс сообщения успешного добавления сотрудника

Теперь добавим информацию о складе также при помощи метода INSERT INTO. Нужно нажать кнопку «Добавить склад» в форме «База складов» и заполнить поля новой формы (cм рисунок 13).

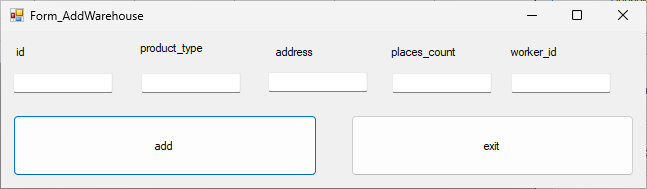


Рисунок 13 – ­­Интерфейс добавления нового склада

Для удаления записей используется sql-выражение DELETE. Удаление происходит по номеру бригады (см рисунок 14). Также связные данные удаляются и из складов.

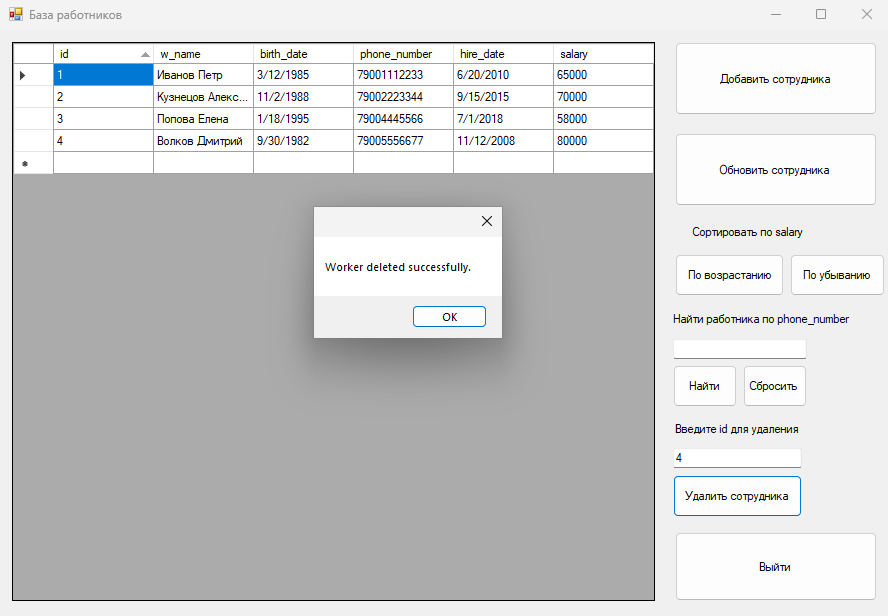


Рисунок 14 – Удаление данных о сотруднике

Для поиска записи в бд используем SELECT. В форме «База сотрудников» вводим номер телефона сотрудника и нажимаем «Найти» (см. рисунок 15). Аналогично можно сделать поиск в форме «База складов» (см. рисунок 16).

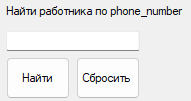
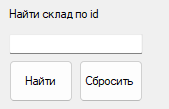
 

Рисунок 15 – Интерфейс поиска по номеру телефона

Рисунок 16 – Интерфейс поиска склада по id

Для сортировки будем использовать кнопки «возрастанию» для сортировки по возрастанию заработной платы, «убыванию» по убыванию заработной платы (см. рисунок 17).

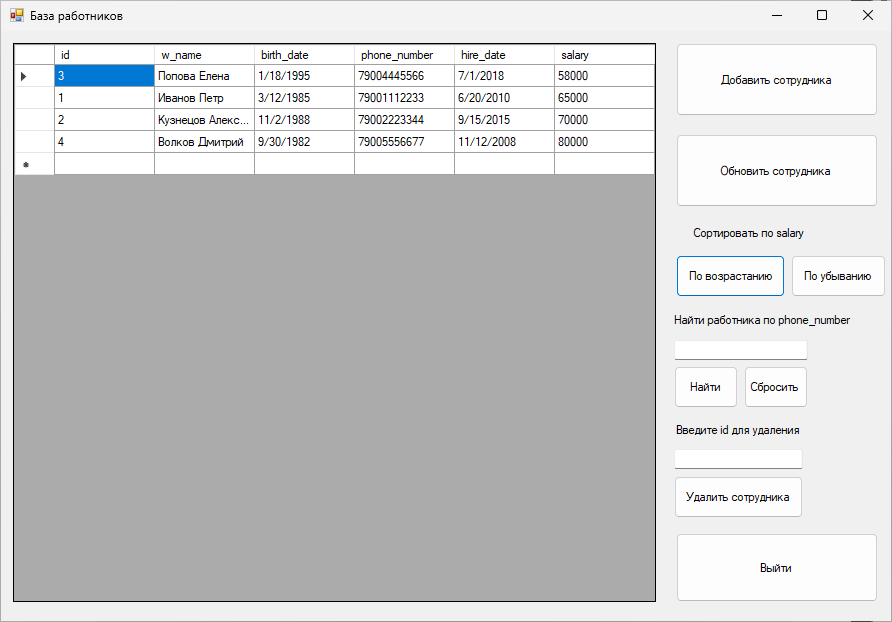


Рисунок 17 – Сортировка сотрудников по возрастанию заработной платы

# **Заключение**

В ходе учебной практики:

1. Приобретены начальные профессиональные умения и навыки.

2. Углублены и расширены теоретические и практические знания, а также навыки, полученные ранее в ходе изучения дисциплин.

3. Применены информационные технологии при расширении теоретических и прикладных задач профессиональной деятельности.

4. Эффективно осуществлялся поиск информации в глобальных информационных сетях с использованием современных поисковых систем, соответствуя потребностям.

5. Полученная из глобальных информационных систем информация использовалась с соблюдением законодательства в области информации, информационных технологий, защиты информации и авторского права.

6. Разработан программный код для решения задач с использованием языков программирования С# и создана база данных в приложении Microsoft Access.

7. Программный код оформлен в соответствии с установленными требованиями.

8. Проверен и отлажен программный код.

# **Список использованных источников**

1. Разработка Windows-приложений в среде программирования Visual Studio.Net [Электронный ресурс]: учебно-методическое пособие по дисциплине Информатика и программирование /. — Электрон. текстовые данные. — М.: Московский технический университет связи и информатики, 2016. — 20 c. — 2227-8397. — Режим доступа: http://www.iprbookshop.ru/61536.html
2. Казанский, А. А. Программирование на visual c# 2013: учебное пособие для прикладного бакалавриата / А. А. Казанский. — М.: Издательство Юрайт, 2018. — 191 с. [Электронный ресурс]. URL: https:// https://urait.ru/book/programmirovanie-na-visual-c-2013-414194
3. Гаспариан М.С. Информационные системы и технологии [Электронный ресурс]: учебное пособие / М.С. Гаспариан, Г.Н. Лихачева. — Электрон. текстовые данные. — М.: Евразийский открытый институт, 2011. — 370 c. — 978-5-374-00192-1. — Режим доступа: <http://www.iprbookshop.ru/10680.html>
4. Курипта О.В. Основы программирования и алгоритмизации [Электронный ресурс]: практикум / О.В. Курипта, О.В. Минакова, Д.К. Проскурин. – Электрон. текстовые данные. – Воронеж: Воронежский государственный архитектурно-строительный университет, ЭБС АСВ, 2015. – 133 c. – 978-5-89040-575-3. – Режим доступа: http://www.iprbookshop.ru/59123.html
5. ГОСТ 7.32-2017. Система стандартов по информации, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления [Электронный ресурс]. URL: <http://protect.gost.ru/document.aspx?control=7&id=218998>
6. Общероссийский классификатор стандартов. Программное обеспечение [Электронный ресурс]. URL: <http://standartgost.ru/0/757-programmnoe_obespechenie>.
7. Официальная документация майкрософт по языку С# [Электронный ресурс]. - URL: https://learn.microsoft.com/ru-ru/dotnet/csharp/
8. Официальная документация майкрософт по Windows Forms [Электронный ресурс]. URL: https://learn.microsoft.com/en-us/dotnet/desktop/winforms/

# **ПРИЛОЖЕНИЕ**

**Form\_Menu (Главное меню)**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Practice

{

public partial class Form\_Menu : Form

{

public Form\_Menu()

{

InitializeComponent();

}

private void warehouse\_button\_Click(object sender, EventArgs e)

{

this.Hide();

Form\_Warehouse f\_w = new Form\_Warehouse();

f\_w.Owner = this;

f\_w.StartPosition = FormStartPosition.Manual;

f\_w.Location = this.Location;

f\_w.Show();

}

private void worker\_button\_Click(object sender, EventArgs e)

{

this.Hide();

Form\_Worker f\_w = new Form\_Worker();

f\_w.Owner = this;

f\_w.StartPosition = FormStartPosition.Manual;

f\_w.Location = this.Location;

f\_w.Show();

}

private void exit\_button\_Click(object sender, EventArgs e)

{

this.Close();

}

private void Form\_Menu\_Load(object sender, EventArgs e)

{

}

}

}

**Form\_Warehouse (База складов)**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.Common;

using System.Data.OleDb;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Practice

{

public partial class Form\_Warehouse : Form

{

public static string connection = "Provider=Microsoft.ACE.OLEDB.16.0;Data Source=DatabaseP.accdb";

public OleDbConnection dbConnection;

public Form\_Warehouse()

{

InitializeComponent();

dbConnection = new OleDbConnection(connection);

dbConnection.Open();

}

private void Form\_Warehouse\_Load(object sender, EventArgs e)

{

this.warehouseTableAdapter.Fill(this.databasePDataSet1.warehouse);

}

private void Form\_Warehouse\_FormClosing(object sender, FormClosingEventArgs e)

{

dbConnection.Close();

}

public void UpdateDataGridView()

{

this.warehouseTableAdapter.Fill(this.databasePDataSet1.warehouse);

}

private void SaveChanges()

{

this.dataGridView1.EndEdit();

this.warehouseBindingSource.EndEdit();

this.warehouseTableAdapter.Update(this.databasePDataSet1.warehouse);

}

private void exit\_button\_Click(object sender, EventArgs e)

{

SaveChanges();

if (this.Owner != null)

this.Owner.Show();

this.Close();

}

private void add\_warehouse\_button\_Click(object sender, EventArgs e)

{

Form\_AddWarehouse faw = new Form\_AddWarehouse();

faw.Owner = this;

faw.MainForm = this;

faw.StartPosition = FormStartPosition.Manual;

faw.Location = this.Location;

faw.Show();

}

private void update\_warehouse\_button\_Click(object sender, EventArgs e)

{

Form\_UpdateWarehouse faw = new Form\_UpdateWarehouse();

faw.Owner = this;

faw.MainForm = this;

faw.StartPosition = FormStartPosition.Manual;

faw.Location = this.Location;

faw.Show();

}

private void delete\_warehouse\_button\_Click(object sender, EventArgs e)

{

try

{

int id = Convert.ToInt32(delete\_warehouse\_textBox.Text);

string query = "DELETE FROM warehouse WHERE [id] = @id";

OleDbCommand command = new OleDbCommand(query, dbConnection);

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

command.ExecuteNonQuery();

MessageBox.Show("Warehouse deleted successfully.");

this.warehouseTableAdapter.Fill(this.databasePDataSet1.warehouse);

}

catch (Exception ex)

{

MessageBox.Show("Error: " + ex.Message);

}

}

private void find\_button\_Click(object sender, EventArgs e)

{

try

{

int worker\_id = Convert.ToInt32(find\_textBox.Text);

string query = "SELECT \* FROM warehouse WHERE worker\_id = @worker\_id";

OleDbCommand command = new OleDbCommand(query, dbConnection);

command.Parameters.AddWithValue("@worker\_id", worker\_id);

OleDbDataAdapter dataAdapter = new OleDbDataAdapter(command);

DataTable dt = new DataTable();

dataAdapter.Fill(dt);

dataGridView1.DataSource = dt;

}

catch (Exception ex)

{

MessageBox.Show("Error: " + ex.Message);

}

}

private void reset\_button\_Click(object sender, EventArgs e)

{

dataGridView1.DataSource = databasePDataSet1.warehouse;

}

private void by\_ascending\_button\_Click(object sender, EventArgs e)

{

string query = "SELECT \* FROM warehouse ORDER BY 4 ASC";

OleDbDataAdapter command = new OleDbDataAdapter(query, dbConnection);

DataTable dt = new DataTable();

command.Fill(dt);

dataGridView1.DataSource = dt;

}

private void by\_descending\_button\_Click(object sender, EventArgs e)

{

string query = "SELECT \* FROM warehouse ORDER BY 4 DESC";

OleDbDataAdapter command = new OleDbDataAdapter(query, dbConnection);

DataTable dt = new DataTable();

command.Fill(dt);

dataGridView1.DataSource = dt;

}

}

}

**Form\_AddWarehouse (добавление склада)**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.OleDb;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Practice

{

public partial class Form\_AddWarehouse : Form

{

public Form\_Warehouse MainForm { get; set; }

public static string connection = "Provider=Microsoft.ACE.OLEDB.16.0;Data Source=DatabaseP.accdb";

private OleDbConnection dbConnection;

public Form\_AddWarehouse()

{

InitializeComponent();

dbConnection = new OleDbConnection(connection);

dbConnection.Open();

}

private void exit\_button\_Click(object sender, EventArgs e)

{

this.Close();

}

private void add\_button\_Click(object sender, EventArgs e)

{

try

{

int id = Convert.ToInt32(id\_textBox.Text);

string product\_type = product\_type\_textBox.Text;

string address = address\_textBox.Text;

int places\_count = Convert.ToInt32(places\_count\_textBox.Text);

int worker\_id = Convert.ToInt32(worker\_id\_textBox.Text);

string query = "INSERT INTO warehouse ([id], product\_type, address, places\_count, worker\_id) " + "VALUES (@id, @product\_type, @address, @places\_count, @worker\_id)";

OleDbCommand command = new OleDbCommand(query, dbConnection);

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

command.Parameters.AddWithValue("@product\_type", product\_type);

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

command.Parameters.AddWithValue("@places\_count", places\_count);

command.Parameters.AddWithValue("@worker\_id", worker\_id);

command.ExecuteNonQuery();

MessageBox.Show("Warehouse added successfully!");

MainForm.UpdateDataGridView();

this.Close();

}

catch (Exception ex)

{

MessageBox.Show("Error: " + ex.Message);

}

}

}

}

**Form\_UpdateWarehouse(обновление склада)**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.Common;

using System.Data.OleDb;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Practice

{

public partial class Form\_UpdateWarehouse : Form

{

public Form\_Warehouse MainForm { get; set; }

public static string connection = "Provider=Microsoft.ACE.OLEDB.16.0;Data Source=DatabaseP.accdb";

private OleDbConnection dbConnection;

public Form\_UpdateWarehouse()

{

InitializeComponent();

dbConnection = new OleDbConnection(connection);

dbConnection.Open();

}

private void update\_button\_Click(object sender, EventArgs e)

{

try

{

int id = Convert.ToInt32(id\_textBox.Text);

string product\_type = product\_type\_textBox.Text;

string address = address\_textBox.Text;

int places\_count = Convert.ToInt32(places\_count\_textBox.Text);

int worker\_id = Convert.ToInt32(worker\_id\_textBox.Text);

string query = "UPDATE warehouse SET product\_type = @product\_type, address = @address, places\_count = @places\_count, worker\_id = @worker\_id WHERE [id] = @id";

OleDbCommand command = new OleDbCommand(query, dbConnection);

command.Parameters.AddWithValue("@product\_type", product\_type);

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

command.Parameters.AddWithValue("@places\_count", places\_count);

command.Parameters.AddWithValue("@worker\_id", worker\_id);

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

command.ExecuteNonQuery();

MessageBox.Show("Warehouse updated!");

MainForm.UpdateDataGridView();

this.Close();

}

catch (Exception ex)

{

MessageBox.Show("Error: " + ex.Message);

}

}

private void exit\_button\_Click(object sender, EventArgs e)

{

this.Close();

}

}

}

**Form\_Worker (База работников)**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.OleDb;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Practice

{

public partial class Form\_Worker : Form

{

public static string connectString = "Provider=Microsoft.ACE.OLEDB.16.0;Data Source=DatabaseP.accdb";

private OleDbConnection dbConnection;

public Form\_Worker()

{

InitializeComponent();

dbConnection = new OleDbConnection(connectString);

dbConnection.Open();

}

private void Form\_Worker\_Load(object sender, EventArgs e)

{

this.workerTableAdapter.Fill(this.databasePDataSet.worker);

}

private void Form\_Worker\_FormClosing(object sender, FormClosingEventArgs e)

{

dbConnection.Close();

}

public void UpdateDataGridView()

{

this.workerTableAdapter.Fill(this.databasePDataSet.worker);

}

private void SaveChanges()

{

this.dataGridView1.EndEdit();

this.workerBindingSource.EndEdit();

this.workerTableAdapter.Update(this.databasePDataSet.worker);

}

private void exit\_button\_Click(object sender, EventArgs e)

{

SaveChanges();

if (this.Owner != null)

this.Owner.Show();

this.Close();

}

private void add\_worker\_button\_Click(object sender, EventArgs e)

{

Form\_AddWorker faw = new Form\_AddWorker();

faw.Owner = this;

faw.MainForm = this;

faw.StartPosition = FormStartPosition.Manual;

faw.Location = this.Location;

faw.Show();

}

private void update\_worker\_button\_Click(object sender, EventArgs e)

{

Form\_UpdateWorker faw = new Form\_UpdateWorker();

faw.Owner = this;

faw.MainForm = this;

faw.StartPosition = FormStartPosition.Manual;

faw.Location = this.Location;

faw.Show();

}

private void by\_ascending\_button\_Click(object sender, EventArgs e)

{

string query = "SELECT \* FROM worker ORDER BY 6 ASC";

OleDbDataAdapter command = new OleDbDataAdapter(query, dbConnection);

DataTable dt = new DataTable();

command.Fill(dt);

dataGridView1.DataSource = dt;

}

private void by\_descending\_button\_Click(object sender, EventArgs e)

{

string query = "SELECT \* FROM worker ORDER BY 6 DESC";

OleDbDataAdapter command = new OleDbDataAdapter(query, dbConnection);

DataTable dt = new DataTable();

command.Fill(dt);

dataGridView1.DataSource = dt;

}

private void find\_button\_Click(object sender, EventArgs e)

{

try

{

string phone\_number = find\_textBox.Text;

string query = "SELECT \* FROM worker WHERE phone\_number = @phone\_number";

OleDbCommand command = new OleDbCommand(query, dbConnection);

command.Parameters.AddWithValue("@phone\_number", phone\_number);

OleDbDataAdapter dataAdapter = new OleDbDataAdapter(command);

DataTable dt = new DataTable();

dataAdapter.Fill(dt);

dataGridView1.DataSource = dt;

}

catch (Exception ex)

{

MessageBox.Show("Error: " + ex.Message);

}

}

private void reset\_button\_Click(object sender, EventArgs e)

{

dataGridView1.DataSource = databasePDataSet.worker;

}

private void delete\_worker\_button\_Click(object sender, EventArgs e)

{

try

{

int id = Convert.ToInt32(delete\_worker\_textBox.Text);

string query = "DELETE FROM worker WHERE [id] = @id";

OleDbCommand command = new OleDbCommand(query, dbConnection);

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

command.ExecuteNonQuery();

MessageBox.Show("Worker deleted successfully.");

this.workerTableAdapter.Fill(this.databasePDataSet.worker);

}

catch (Exception ex)

{

MessageBox.Show("Error: " + ex.Message);

}

}

}

}

**Form\_AddWorker (добавление работника)**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.OleDb;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Practice

{

public partial class Form\_AddWorker : Form

{

public Form\_Worker MainForm { get; set; }

public static string connection = "Provider=Microsoft.ACE.OLEDB.16.0;Data Source=DatabaseP.accdb";

private OleDbConnection dbConnection;

public Form\_AddWorker()

{

InitializeComponent();

dbConnection = new OleDbConnection(connection);

dbConnection.Open();

}

private void add\_button\_Click(object sender, EventArgs e)

{

try

{

int id = Convert.ToInt32(id\_textBox.Text);

string w\_name = w\_name\_textBox.Text;

DateTime birth\_date = DateTime.Parse(birth\_date\_textBox.Text);

string phone\_number = phone\_number\_textBox.Text;

DateTime hire\_date = DateTime.Parse(hire\_date\_textBox.Text);

int salary = Convert.ToInt32(salary\_textBox.Text);

string query = "INSERT INTO worker ([id], w\_name, birth\_date, phone\_number, hire\_date, salary) " +

"VALUES (@id, @w\_name, @birth\_date, @phone\_number, @hire\_date, @salary)";

OleDbCommand command = new OleDbCommand(query, dbConnection);

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

command.Parameters.AddWithValue("@w\_name", w\_name);

command.Parameters.AddWithValue("@birth\_date", birth\_date);

command.Parameters.AddWithValue("@phone\_number", phone\_number);

command.Parameters.AddWithValue("@hire\_date", hire\_date);

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

command.ExecuteNonQuery();

MessageBox.Show("Worker added successfully!");

MainForm.UpdateDataGridView();

this.Close();

}

catch (Exception ex)

{

MessageBox.Show("Error: " + ex.Message);

}

}

private void exit\_button\_Click(object sender, EventArgs e)

{

this.Close();

}

}

}

**Form\_UpdateWorker (обновление работника)**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.OleDb;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Practice

{

public partial class Form\_UpdateWorker : Form

{

public Form\_Worker MainForm { get; set; }

public static string connection = "Provider=Microsoft.ACE.OLEDB.16.0;Data Source=DatabaseP.accdb";

private OleDbConnection dbConnection;

public Form\_UpdateWorker()

{

InitializeComponent();

dbConnection = new OleDbConnection(connection);

dbConnection.Open();

}

private void update\_button\_Click(object sender, EventArgs e)

{

try

{

int id = Convert.ToInt32(id\_textBox.Text);

string w\_name = w\_name\_textBox.Text;

DateTime birth\_date = DateTime.Parse(birth\_date\_textBox.Text);

string phone\_number = phone\_number\_textBox.Text;

DateTime hire\_date = DateTime.Parse(hire\_date\_textBox.Text);

int salary = Convert.ToInt32(salary\_textBox.Text);

string query = "UPDATE worker " +

"SET w\_name = @w\_name, birth\_date = @birth\_date, phone\_number = @phone\_number, hire\_date = @hire\_date, salary = @salary " +

"WHERE [id] = @id";

OleDbCommand command = new OleDbCommand(query, dbConnection);

command.Parameters.AddWithValue("@w\_name", w\_name);

command.Parameters.AddWithValue("@birth\_date", birth\_date);

command.Parameters.AddWithValue("@phone\_number", phone\_number);

command.Parameters.AddWithValue("@hire\_date", hire\_date);

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

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

command.ExecuteNonQuery();

MessageBox.Show("Worker updated!");

MainForm.UpdateDataGridView();

this.Close();

}

catch (Exception ex)

{

MessageBox.Show("Error: " + ex.Message);

}

}

private void exit\_button\_Click(object sender, EventArgs e)

{

this.Close();

}

}

}