Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Институт информационных технологий и

анализа данных

|  |
| --- |
| наименование института |
| **Отчет** по лабораторной работе №1  по дисциплине «Объектно-ориентированные базы данных»  «Databases 4 Objects» | | |  |

наименование темы

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | |
| Выполнил студент |  | ИСМб-19-1 | |  |  |  | Ю.А. Михейко |
| Проверил |  | шифр группы | |  | подпись |  | И.О. Фамилия  В.А. Харахинов |
|  |  |  | |  | подпись |  | И.О. Фамилия |
| Работа защищена с оценкой | | |  | |  | | |

**Содержание**

[**1 Задание 3**](#_Toc119675403)

[**2 Классы 4**](#_Toc119675404)

[**3 Работа программы 34**](#_Toc119675405)

# **1 Задание**

**Сборка ПК**

Предметная область включает в себя Процессор, Блок питания, Материнскую плату, Видеокарту, ОЗУ и ПЗУ

Необходимо реализовать CRUD функциональность для данной предметной области.

**Выходные документы:**

* Самый востребованный производитель материнских плат.
* Список видеокарт определенной мощности

# **2 Классы**

В данной лабораторной работе необходимо было разработать 7 классов: CPU, MotherBoard, PowerBlock, RAM, Storarge, VideoCArd и DataBase на языке C# c использованием библиотеки db4o.

Код классов представлен ниже.

*Класс CPU:*

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace db.Classes

{

internal class CPU

{

[DisplayName("id")]

public string id { get; set; }

[DisplayName("Тактовая частота")]

public float frequency { get; set; }

[DisplayName("Энергопотребление")]

public int power { get; set; }

[DisplayName("Сокет")]

public string socket { get; set; }

[DisplayName("Архитектура")]

public string archetype { get; set; }

[DisplayName("Название")]

public string title { get; set; }

[DisplayName("Производитель")]

public string manufacturer { get; set; }

public CPU()

{

id = Guid.NewGuid().ToString();

}

public CPU(float frequency, int power, string socket, string archetype, string title, string manufacturer)

{

id = Guid.NewGuid().ToString();

this.frequency = frequency;

this.power = power;

this.socket = socket;

this.archetype = archetype;

this.title = title;

this.manufacturer = manufacturer;

}

}

}

*Класс MotherBoard:*

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace db.Classes

{

internal class MotherBoard

{

[DisplayName("id")]

public string id { get; set; }

[DisplayName("Сокет")]

public string socket { get; set; }

[DisplayName("Поддерживаемые видеокарты")]

public string GCPUtype { get; set; }

[DisplayName("Тип поддерживаемой RAM")]

public string RAMtype { get; set; }

[DisplayName("Название")]

public string title { get; set; }

[DisplayName("Производитель")]

public string manufacturer { get; set; }

public MotherBoard()

{

id = Guid.NewGuid().ToString();

}

public MotherBoard(string socket, string gCPUtype, string rAMtype, string title, string manufacturer)

{

id = Guid.NewGuid().ToString();

this.socket = socket;

GCPUtype = gCPUtype;

RAMtype = rAMtype;

this.title = title;

this.manufacturer = manufacturer;

}

public override bool Equals(object obj)

{

return obj is MotherBoard board &&

id == board.id &&

socket == board.socket &&

GCPUtype == board.GCPUtype &&

RAMtype == board.RAMtype &&

title == board.title &&

manufacturer == board.manufacturer;

}

}

}

*Класс PowerBlock:*

using System;

using System.ComponentModel;

namespace db.Classes

{

internal class PowerBlock

{

[DisplayName("id")]

public string id { get; set; }

[DisplayName("Мощность")]

public int power { get; set; }

[DisplayName("Название")]

public string title { get; set; }

[DisplayName("Производитель")]

public string manufacturer { get; set; }

public PowerBlock()

{

id = Guid.NewGuid().ToString();

}

public PowerBlock(int power, string title, string manufacturer)

{

id = Guid.NewGuid().ToString();

this.power = power;

this.title = title;

this.manufacturer = manufacturer;

}

}

}

*Класс RAM:*

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace db.Classes

{

internal class RAM

{

[DisplayName("id")]

public string id { get; set; }

[DisplayName("Тип памяти")]

public string typeMemory { get; set; }

[DisplayName("Частота")]

public int frequency { get; set; }

[DisplayName("Объем")]

public int volune { get; set; }

[DisplayName("Название")]

public string title { get; set; }

[DisplayName("Производитель")]

public string manufacturer { get; set; }

public RAM()

{

id = Guid.NewGuid().ToString();

}

public RAM(string typeMemory, int frequency, int volune, string title, string manufacturer)

{

id = Guid.NewGuid().ToString();

this.typeMemory = typeMemory;

this.frequency = frequency;

this.volune = volune;

this.title = title;

this.manufacturer = manufacturer;

}

}

}

*Класс Storage:*

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace db.Classes

{

internal class Storage

{

[DisplayName("id")]

public string id { get; set; }

[DisplayName("Объем")]

public int volune { get; set; }

[DisplayName("Тип памяти")]

public string type { get; set; }

[DisplayName("Скорость записи")]

public int speedOfWrite { get; set; }

[DisplayName("Скорость чтения")]

public int speedOfRead { get; set; }

[DisplayName("Название")]

public string title { get; set; }

[DisplayName("Производитель")]

public string manufacturer { get; set; }

public Storage()

{

id = Guid.NewGuid().ToString();

}

public Storage(int volune, string type, int speedOfWrite, int speedOfRead, string title, string manufacturer)

{

id = Guid.NewGuid().ToString();

this.volune = volune;

this.type = type;

this.speedOfWrite = speedOfWrite;

this.speedOfRead = speedOfRead;

this.title = title;

this.manufacturer = manufacturer;

}

}

}

*Класс VideoCard:*

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace db.Classes

{

internal class VideoCard

{

[DisplayName("id")]

public string id { get; set; }

[DisplayName("Графический CPU")]

public string GCPU { get; set; }

[DisplayName("Объем памяти")]

public int volumeMemory { get; set; }

[DisplayName("Тип памяти")]

public string typeMemory { get; set; }

[DisplayName("Частота памяти")]

public int frequencyMemory { get; set; }

private int \_power;

[DisplayName("Мощность")]

public int Power

{

get { return \_power; }

set { \_power = value; }

}

[DisplayName("Название")]

public string title { get; set; }

[DisplayName("Производитель")]

public string manufacturer { get; set; }

public VideoCard()

{

id = Guid.NewGuid().ToString();

}

public VideoCard(string gCPU, int volumeMemory, string typeMemory, int frequencyMemory, int power, string title, string manufacturer)

{

id = Guid.NewGuid().ToString();

GCPU = gCPU;

this.volumeMemory = volumeMemory;

this.typeMemory = typeMemory;

this.frequencyMemory = frequencyMemory;

this.Power = power;

this.title = title;

this.manufacturer = manufacturer;

}

}

}

*Класс DataBase:*

using Db4objects.Db4o;

using Db4objects.Db4o.Linq;

using Db4objects.Db4o.Query;

using System;

using System.Collections;

using System.Collections.Generic;

using System.Diagnostics;

using System.Linq;

namespace db.Classes

{

class DataBase

{

const String filename = @"pcBuild.yap";

IObjectContainer db;

public void dbConnect()

{

db = Db4oFactory.OpenFile(filename);

}

public void dbClose()

{

db.Close();

}

#region Работа с материнской платой

public void addMotherBoard(MotherBoard motherBoard)

{

db.Store(motherBoard);

}

public List<MotherBoard> getMotherBoard()

{

return db.Query<MotherBoard>().ToList();

}

internal void deleteMotherBoard(string id)

{

var example = new MotherBoard();

example.id = id;

var found = db.QueryByExample(example).Next() as MotherBoard;

db.Delete(found);

}

internal void updateMotherBoard(MotherBoard mb)

{

var example = new MotherBoard();

example.id = mb.id;

var found = db.QueryByExample(example).Next() as MotherBoard;

found.manufacturer = found.manufacturer;

found.RAMtype = mb.RAMtype;

found.socket = mb.socket;

found.title = mb.title;

found.GCPUtype = mb.GCPUtype;

db.Store(found);

}

#endregion

#region Работа с процессором

public void AddCPU(CPU cpu)

{

db.Store(cpu);

}

public void DeleteCPU(string id)

{

var found = db.Query<CPU>(cpu => cpu.id == id);

db.Delete(found);

}

public List<CPU> GetCPU()

{

return db.Query<CPU>().ToList();

}

#endregion

#region

public void AddVideoCard(VideoCard vc)

{

db.Store(vc);

}

public void DeleteVideoCard(string id)

{

var found = db.Query<VideoCard>(vc => vc.id == id)[0];

db.Delete(found);

}

public List<VideoCard> GetVideoCard() => db.Query<VideoCard>().ToList();

#endregion

#region

public void AddPowerBlock(PowerBlock pb)

{

db.Store(pb);

}

public void DeletePowerBlock(string id)

{

var found = db.Query<PowerBlock>(pb => pb.id == id)[0];

db.Delete(found);

}

public List<PowerBlock> GetPowerBlocks() => db.Query<PowerBlock>().ToList();

#endregion

#region Работа с ПЗУ

public void AddStorage(Storage storage)

{

db.Store(storage);

}

public void DeleteStorage(string id)

{

var proto = new Storage();

proto.id = id;

proto.manufacturer = null;

proto.title = null;

proto.type = null;

proto.speedOfRead = 0;

proto.speedOfWrite = 0;

proto.volune = 0;

var found = db.QueryByExample(proto).Next() as Storage;

db.Delete(found);

}

public List<Storage> GetSrorage() => db.Query<Storage>().ToList();

#endregion

#region Работа с ОЗУ

public void AddRAM(RAM ram)

{

db.Store(ram);

}

public void DeleteRAM(string id)

{

var found = db.Query<RAM>(ram => ram.id == id)[0];

db.Delete(found);

}

public List<RAM> GetRAMs() => db.Query<RAM>().ToList();

#endregion

public List<string> MotherBoardSearch()

{

var asn = (from MotherBoard tt in db.Cast<MotherBoard>()

group tt by tt.manufacturer into t

select new

{

Name = t.Key,

Count = t.Count()

}).OrderByDescending(v => v.Count).ToList();

var lists = new List<string>();

lists.Add(asn[0].Name);

for (int i = 0; i < asn.Count-1; i++)

{

if (asn[i].Count == asn[i + 1].Count)

{

lists.Add(asn[i+1].Name);

}

else

{

break;

}

}

return lists;

}

public List<VideoCard> VideoCardSearch(int power)

{

IQuery query = db.Query();

query.Constrain(typeof(VideoCard));

query.Descend("\_power").Constrain(power).Smaller().Equal();

IObjectSet result = query.Execute();

var soda = new List<VideoCard>();

foreach (var r in result)

{

soda.Add(r as VideoCard);

}

return soda;

}

}

}

Также, были разработаны формы: Form1 для общей работы с данными.

Листинг Form1:

using db.Classes;

using Db4objects.Db4o.IO;

using Db4objects.Db4o.Reflect.Net;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Runtime.InteropServices;

using System.Security.Policy;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace db

{

public partial class Form1 : Form

{

#region Настройка формы

DataBase dataBase;

static Random random = new Random();

List<String> mbMonufacturers = new List<string>();

List<String> socket = new List<string>();

List<String> GCPU = new List<string>();

List<String> RAMtype = new List<string>();

List<String> archetype = new List<string>();

List<String> cpuMonufacturer = new List<string>();

List<float> cpuFrequency = new List<float>();

List<String> vcManufacturer = new List<string>();

List<string> vcGCPU = new List<string>();

List<string> vcMemory = new List<string>();

List<int> vcFrequency = new List<int>();

List<int> vcPower = new List<int>();

List<int> vcVolumeMemory = new List<int>();

List<int> pbPower = new List<int>();

List<String> pbManufacturer = new List<string>();

List<String> storageManufacturer = new List<string>();

List<int> storageVolume = new List<int>();

List<String> storageType = new List<string>();

List<int> storageWrite = new List<int>();

List<int> storageRead = new List<int>();

List<String> ramManufacturer = new List<string>();

List<int> ramVolume = new List<int>();

List<int> ramFrequency = new List<int>();

List<String> ramType = new List<string>();

string chars = "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789" + "ABCDEFGHIJKLMNOPQRSTUVWXYZ".ToLower();

public Form1()

{

InitializeComponent();

ListsSet();

dataBase = new DataBase();

dataBase.dbConnect();

setBindingSource();

setCMB();

rtxtMotherBoardSearch.Text = "Самый востребованный производитель";

//foreach (var ans in dataBase.MotherBoardSearch())

//{

// rtxtMotherBoardSearch.Text += $"\n{ans}";

//}

}

void ListsSet()

{

mbMonufacturers.Add("ASRock");

mbMonufacturers.Add("MSI");

mbMonufacturers.Add("ASUS");

mbMonufacturers.Add("Gigabyte");

socket.Add("Socket 8");

socket.Add("Socket 370");

socket.Add("Socket 423");

socket.Add("Socket 478");

socket.Add("LGA 775");

socket.Add("LGA 771");

socket.Add("LGA 1366");

socket.Add("LGA 1156");

socket.Add("LGA 1567");

socket.Add("LGA 1155");

socket.Add("LGA 2011");

socket.Add("LGA 1356");

socket.Add("LGA 1150");

socket.Add("LGA2011-3");

socket.Add("LGA 1151");

socket.Add("LGA 3647");

socket.Add("LGA 2066");

socket.Add("LGA 1200");

socket.Add("LGA 1700");

socket.Add("Super Socket 7");

socket.Add("Slot A");

socket.Add("Socket A");

socket.Add("Socket 754");

socket.Add("Socket 940");

socket.Add("Socket 939");

socket.Add("Socket S1");

socket.Add("Socket AM2");

socket.Add("Socket F");

socket.Add("Socket AM2+");

socket.Add("Socket AM3");

socket.Add("Socket G34");

socket.Add("Socket C32");

socket.Add("Socket AM3+");

socket.Add("Socket FM1");

socket.Add("Socket FS1");

socket.Add("Socket FM2");

socket.Add("Socket FM2+");

socket.Add("Socket AM1");

socket.Add("АМ4");

socket.Add("Socket SP3");

socket.Add("Socket TR4");

socket.Add("Socket sTRX4");

GCPU.Add("GeForce series");

for (int i = 2; i < 10; i++)

{

GCPU.Add($"GeForce {i} series");

}

for (int i = 100; i <= 900; i += 100)

{

GCPU.Add($"GeForce {i} series");

}

GCPU.Add("GeForce 10 series");

GCPU.Add("GeForce 16 series");

GCPU.Add("GeForce 20 series");

for (int i = 100; i <= 800; i += 100)

{

GCPU.Add($"Radeon R{i}");

}

for (int i = 1; i <= 4; i++)

{

RAMtype.Add($"DDR{i}");

}

archetype.Add("Nehalem");

archetype.Add("Westmere");

archetype.Add("Sandy Bridge");

archetype.Add("Ivy Bridge");

archetype.Add("Haswell");

archetype.Add("Broadwell");

archetype.Add("Skylake");

archetype.Add("Kaby Lake");

archetype.Add("Coffee Lake");

archetype.Add("Coffee Lake Refresh");

archetype.Add("Dali");

archetype.Add("Zen 2");

archetype.Add("Zen 3");

archetype.Add("Raven Ridge");

archetype.Add("Zen");

archetype.Add("Bristol Ridge");

archetype.Add("Seattle");

archetype.Add("Merlin Falcon");

archetype.Add("Kyoto");

archetype.Add("Kabini");

cpuMonufacturer.Add("Intel");

cpuMonufacturer.Add("AMD");

for(float i = 1.0f; i <= 4.0f; i += 0.5f)

{

cpuFrequency.Add(i);

}

vcManufacturer.Add("NVIDIA");

vcManufacturer.Add("AMD");

for (int i = 128; i <= 16384; i \*= 2)

{

vcVolumeMemory.Add(i);

}

vcMemory.Add("DDR");

vcMemory.Add("DDR2");

vcMemory.Add("GDDR3");

vcMemory.Add("GDDR4");

vcMemory.Add("GDDR5");

vcMemory.Add("GDDR6");

for (int i = 40; i <= 600; i += 40)

{

vcPower.Add(i);

}

for (int i = 500; i <= 5000; i += 250)

{

vcFrequency.Add(i);

}

vcGCPU.Add("GeForce series");

for (int i = 2; i < 10; i++)

{

vcGCPU.Add($"GeForce {i} series");

}

for (int i = 100; i <= 900; i += 100)

{

vcGCPU.Add($"GeForce {i} series");

}

vcGCPU.Add("GeForce 10 series");

vcGCPU.Add("GeForce 16 series");

vcGCPU.Add("GeForce 20 series");

vcGCPU.Add("GeForce 30 series");

vcGCPU.Add("GeForce 40 series");

for (int i = 100; i <= 800; i += 100)

{

vcGCPU.Add($"Radeon R{i}");

}

pbManufacturer.Add("Thermaltake");

pbManufacturer.Add("Sea Sonic");

pbManufacturer.Add("Chieftec");

pbManufacturer.Add("Cougar");

pbManufacturer.Add("Zalman");

pbManufacturer.Add("ENERMAX");

pbManufacturer.Add("BE QUIET!");

pbManufacturer.Add("Corsair");

pbManufacturer.Add("DeepCool");

pbManufacturer.Add("FSP");

for (int i = 300; i <= 1500; i += 150)

{

pbPower.Add(i);

}

storageManufacturer.Add("Western Digital");

storageManufacturer.Add("Seagate");

storageManufacturer.Add("Hitachi HGST");

storageManufacturer.Add("Toshiba");

storageManufacturer.Add("Samsung");

storageType.Add("HHD");

storageType.Add("SSD");

storageType.Add("M2");

for (int i = 128; i <= 8192; i \*= 2)

{

storageVolume.Add(i);

}

for (int i = 50; i <= 300; i += 50)

{

storageRead.Add(i);

}

for (int i = 10; i <= 60; i += 10)

{

storageWrite.Add(i);

}

ramManufacturer.Add("Kingston");

ramManufacturer.Add("Hynix");

ramManufacturer.Add("Crucial");

ramManufacturer.Add("Patriot Memory");

ramManufacturer.Add("Samsung");

ramManufacturer.Add("Corsair");

ramManufacturer.Add("G.Skill");

ramManufacturer.Add("Apacer");

ramManufacturer.Add("GoodRAM");

ramManufacturer.Add("Silicon Power");

for (int i = 1; i <= 4; i++)

{

ramType.Add($"DDR{i}");

}

for (int i = 200; i <= 3200; i += 100)

{

ramFrequency.Add(i);

}

for (int i = 512; i <= 16384; i \*= 2)

{

ramVolume.Add(i);

}

}

void setBindingSource()

{

foreach (var m in dataBase.getMotherBoard())

{

motherBoardBindingSource.Add(m);

}

foreach (var c in dataBase.GetCPU())

{

cPUBindingSource.Add(c);

}

foreach (var vc in dataBase.GetVideoCard())

{

videoCardBindingSource.Add(vc);

}

foreach(var pb in dataBase.GetPowerBlocks())

{

powerBlockBindingSource.Add(pb);

}

foreach(var s in dataBase.GetSrorage())

{

storageBindingSource.Add(s);

}

foreach(var ram in dataBase.GetRAMs())

{

rAMBindingSource.Add(ram);

}

}

void setCMB()

{

cmbMBSocket.DataSource = socket;

cmbMBGCPU.DataSource = GCPU;

cmbMBmonufacturer.DataSource = mbMonufacturers;

cmbMBram.DataSource = RAMtype;

cmbMBGCPU.SelectedIndex = 0;

cmbMBmonufacturer.SelectedIndex = 0;

cmbMBram.SelectedIndex = 0;

cmbMBSocket.SelectedIndex = 0;

cmbCpuArchetype.DataSource = archetype;

cmbCpuArchetype.SelectedIndex = 0;

cmbCpuSocket.DataSource = socket;

cmbCpuSocket.SelectedIndex = 0;

cmbCpuMonufacturer.DataSource = cpuMonufacturer;

cmbCpuMonufacturer.SelectedIndex = 0;

cmbCpuFrequency.DataSource = cpuFrequency;

cmbCpuFrequency.SelectedIndex = 0;

cmbVideoCardFrequency.DataSource = vcFrequency;

cmbVideoCardFrequency.SelectedIndex = 0;

cmbVideoCardGCPU.DataSource = vcGCPU;

cmbVideoCardGCPU.SelectedIndex = 0;

cmbVideoCardManufacturer.DataSource = vcManufacturer;

cmbVideoCardManufacturer.SelectedIndex = 0;

cmbVideoCardMemory.DataSource = vcMemory;

cmbVideoCardMemory.SelectedIndex = 0;

cmbVideoCardPower.DataSource = vcPower;

cmbVideoCardPower.SelectedIndex = 0;

cmbVideoCardVolume.DataSource = vcVolumeMemory;

cmbVideoCardVolume.SelectedIndex = 0;

cmbPowerBlockManufacturer.DataSource = pbManufacturer;

cmbPowerBlockManufacturer.SelectedIndex = 0;

cmbPowerBlockPower.DataSource = pbPower;

cmbPowerBlockPower.SelectedIndex = 0;

cmbStorageManufacturer.DataSource = storageManufacturer;

cmbStorageManufacturer.SelectedIndex = 0;

cmbStorageRead.DataSource = storageRead;

cmbStorageRead.SelectedIndex = 0;

cmbStorageType.DataSource = storageType;

cmbStorageType.SelectedIndex = 0;

cmbStorageVolume.DataSource = storageVolume;

cmbStorageVolume.SelectedIndex = 0;

cmbStorageWrite.DataSource = storageWrite;

cmbStorageWrite.SelectedIndex = 0;

cmbRamFrenquency.DataSource = ramFrequency;

cmbRamFrenquency.SelectedIndex = 0;

cmbRamManufacturer.DataSource = ramManufacturer;

cmbRamManufacturer.SelectedIndex = 0;

cmbRamType.DataSource = ramType;

cmbRamType.SelectedIndex = 0;

cmbRamVolume.DataSource = ramVolume;

cmbRamVolume.SelectedIndex = 0;

}

#endregion

#region Утилиты

void messageBoxSuccessAdd()

{

MessageBox.Show("Запись успешно добавлена",

"Успех", MessageBoxButtons.OK,

MessageBoxIcon.Information);

}

void messageBoxError(string message)

{

MessageBox.Show(message,

"Ошибка", MessageBoxButtons.OK,

MessageBoxIcon.Error);

}

DialogResult messageBoxClickResult(string message)

{

var dialogResult = MessageBox.Show(message,

"Предупреждение",

MessageBoxButtons.YesNo,

MessageBoxIcon.Question);

return dialogResult;

}

string generateString(int from, int to)

{

return new string(Enumerable.Repeat(chars, random.Next(from, to))

.Select(s => s[random.Next(s.Length)]).ToArray());

}

//Отменение изменений

private void tableValidating(object sender, DataGridViewCellValidatingEventArgs e)

{

if (string.IsNullOrEmpty(e.FormattedValue.ToString()))

{

((DataGridView)sender).CancelEdit();

((DataGridView)sender).EndEdit();

}

}

#endregion

#region Работа с материнской платой

private void dgvMotherBoard\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

if (e.RowIndex < 0)

{

return;

}

if (dgvMotherBoard.Columns[e.ColumnIndex].Index == dgvMotherBoard.Columns.Count - 1)

{

if (messageBoxClickResult("Удалить эту запись?") == DialogResult.Yes)

{

var id = dgvMotherBoard.Rows[e.RowIndex].Cells[0].Value.ToString();

motherBoardBindingSource.RemoveAt(e.RowIndex);

dataBase.deleteMotherBoard(id);

}

}

}

MotherBoard selectedMB;

private void dgvMotherBoard\_CellBeginEdit(object sender, DataGridViewCellCancelEventArgs e)

{

selectedMB = new MotherBoard();

selectedMB.id = dgvMotherBoard.Rows[e.RowIndex].Cells[0].Value.ToString();

selectedMB.socket = dgvMotherBoard.Rows[e.RowIndex].Cells[1].Value.ToString();

selectedMB.GCPUtype = dgvMotherBoard.Rows[e.RowIndex].Cells[2].Value.ToString();

selectedMB.RAMtype = dgvMotherBoard.Rows[e.RowIndex].Cells[3].Value.ToString();

selectedMB.title = dgvMotherBoard.Rows[e.RowIndex].Cells[4].Value.ToString();

selectedMB.manufacturer = dgvMotherBoard.Rows[e.RowIndex].Cells[5].Value.ToString();

}

private void dgvMotherBoard\_CellEndEdit(object sender, DataGridViewCellEventArgs e)

{

var mb = new MotherBoard();

mb.id = dgvMotherBoard.Rows[e.RowIndex].Cells[0].Value.ToString();

mb.socket = dgvMotherBoard.Rows[e.RowIndex].Cells[1].Value.ToString();

mb.GCPUtype = dgvMotherBoard.Rows[e.RowIndex].Cells[2].Value.ToString();

mb.RAMtype = dgvMotherBoard.Rows[e.RowIndex].Cells[3].Value.ToString();

mb.title = dgvMotherBoard.Rows[e.RowIndex].Cells[4].Value.ToString();

mb.manufacturer = dgvMotherBoard.Rows[e.RowIndex].Cells[5].Value.ToString();

if (selectedMB.Equals(mb))

{

return;

}

var dialogResult = messageBoxClickResult("Изменить эту запись?");

if (dialogResult == DialogResult.No)

{

dgvMotherBoard[1, e.RowIndex].Value = selectedMB.socket;

dgvMotherBoard[2, e.RowIndex].Value = selectedMB.GCPUtype;

dgvMotherBoard[3, e.RowIndex].Value = selectedMB.RAMtype;

dgvMotherBoard[4, e.RowIndex].Value = selectedMB.title;

dgvMotherBoard[5, e.RowIndex].Value = selectedMB.manufacturer;

return;

}

if (dialogResult == DialogResult.Yes)

{

dataBase.updateMotherBoard(dgvMotherBoard.CurrentRow.DataBoundItem as MotherBoard);

}

}

private void cbMotherBoard\_CheckedChanged(object sender, EventArgs e)

{

if (cbMotherBoard.Checked)

{

dgvMotherBoard.Columns[1].ReadOnly = false;

dgvMotherBoard.Columns[2].ReadOnly = false;

dgvMotherBoard.Columns[3].ReadOnly = false;

dgvMotherBoard.Columns[4].ReadOnly = false;

dgvMotherBoard.Columns[5].ReadOnly = false;

dgvMotherBoard.Columns[6].Visible = true;

}

else

{

dgvMotherBoard.Columns[1].ReadOnly = true;

dgvMotherBoard.Columns[2].ReadOnly = true;

dgvMotherBoard.Columns[3].ReadOnly = true;

dgvMotherBoard.Columns[4].ReadOnly = true;

dgvMotherBoard.Columns[5].ReadOnly = true;

dgvMotherBoard.Columns[6].Visible = false;

}

}

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

{

if (string.IsNullOrEmpty(txtMBtitle.Text))

{

messageBoxError("Вы не ввели название");

return;

}

var mb = new MotherBoard();

mb.manufacturer = cmbMBmonufacturer.Text;

mb.RAMtype = cmbMBram.Text;

mb.socket = cmbMBSocket.Text;

mb.title = txtMBtitle.Text;

mb.GCPUtype = cmbMBGCPU.Text;

dataBase.addMotherBoard(mb);

motherBoardBindingSource.Add(mb);

txtMBtitle.Clear();

messageBoxSuccessAdd();

rtxtMotherBoardSearch.Clear();

rtxtMotherBoardSearch.Text = "Самый востребованный производитель";

foreach(var ans in dataBase.MotherBoardSearch())

{

rtxtMotherBoardSearch.Text += $"\n{ans}";

}

}

#endregion

#region Работа с процессором

private void cbCpuIsDelete\_CheckedChanged(object sender, EventArgs e)

{

if (cbCpuIsDelete.Checked)

{

dgvCPU.Columns[7].Visible = true;

}

else

{

dgvCPU.Columns[7].Visible = false;

}

}

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

{

if (string.IsNullOrEmpty(txtCpuTitle.Text))

{

messageBoxError("Вы не ввели название");

return;

}

var cpu = new CPU();

cpu.title = txtCpuTitle.Text;

cpu.power = (int)nudCpuPower.Value;

cpu.manufacturer = cmbCpuMonufacturer.SelectedItem.ToString();

cpu.socket = cmbCpuSocket.SelectedItem.ToString();

cpu.archetype = cmbCpuArchetype.SelectedItem.ToString();

cpu.frequency = (float)cmbCpuFrequency.SelectedItem;

dataBase.AddCPU(cpu);

cPUBindingSource.Add(cpu);

messageBoxSuccessAdd();

}

private void dgvCPU\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

if (e.RowIndex < 0)

{

return;

}

if (dgvCPU.Columns[e.ColumnIndex].Index == dgvCPU.Columns.Count - 1)

{

if (messageBoxClickResult("Удалить эту запись?") == DialogResult.Yes)

{

var id = dgvCPU.Rows[e.RowIndex].Cells[0].Value.ToString();

cPUBindingSource.RemoveAt(e.RowIndex);

dataBase.DeleteCPU(id);

}

}

}

#endregion

#region Работа с видео картой

private void cbVideoCardDelete\_CheckedChanged(object sender, EventArgs e)

{

if (cbVideoCardDelete.Checked)

{

dgvVideoCard.Columns[8].Visible = true;

}

else

{

dgvVideoCard.Columns[8].Visible = false;

}

}

private void dgvVideoCard\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

if (e.RowIndex < 0)

{

return;

}

if (dgvVideoCard.Columns[e.ColumnIndex].Index == dgvVideoCard.Columns.Count - 1)

{

if (messageBoxClickResult("Удалить эту запись?") == DialogResult.Yes)

{

var id = dgvVideoCard.Rows[e.RowIndex].Cells[0].Value.ToString();

videoCardBindingSource.RemoveAt(e.RowIndex);

dataBase.DeleteVideoCard(id);

}

}

}

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

{

if (string.IsNullOrEmpty(txtVideoCardTitle.Text))

{

messageBoxError("Вы не ввели название");

return;

}

var vc = new VideoCard();

vc.title = txtVideoCardTitle.Text;

vc.manufacturer = cmbVideoCardManufacturer.SelectedItem.ToString();

vc.frequencyMemory = (int)cmbVideoCardFrequency.SelectedItem;

vc.volumeMemory = (int)cmbVideoCardVolume.SelectedItem;

vc.typeMemory = cmbVideoCardMemory.SelectedItem.ToString();

vc.GCPU = cmbVideoCardGCPU.SelectedItem.ToString();

vc.Power = (int)cmbVideoCardPower.SelectedItem;

dataBase.AddVideoCard(vc);

videoCardBindingSource.Add(vc);

txtVideoCardTitle.Clear();

messageBoxSuccessAdd();

}

#endregion

#region Работа с блоком питания

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

{

if (string.IsNullOrEmpty(txtPowerBlockTitle.Text))

{

messageBoxError("Вы не ввели название");

return;

}

var pb = new PowerBlock();

pb.manufacturer = cmbPowerBlockManufacturer.SelectedItem.ToString();

pb.power = (int)cmbPowerBlockPower.SelectedItem;

pb.title = txtPowerBlockTitle.Text;

dataBase.AddPowerBlock(pb);

powerBlockBindingSource.Add(pb);

txtPowerBlockTitle.Clear();

messageBoxSuccessAdd();

}

private void cbPowerBlockDelete\_CheckedChanged(object sender, EventArgs e)

{

if (cbPowerBlockDelete.Checked)

{

dgvPowerBlock.Columns[4].Visible = true;

}

else

{

dgvPowerBlock.Columns[4].Visible = false;

}

}

private void dgvPowerBlock\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

if (e.RowIndex < 0)

{

return;

}

if (dgvPowerBlock.Columns[e.ColumnIndex].Index == dgvPowerBlock.Columns.Count - 1)

{

if (messageBoxClickResult("Удалить эту запись?") == DialogResult.Yes)

{

var id = dgvPowerBlock.Rows[e.RowIndex].Cells[0].Value.ToString();

powerBlockBindingSource.RemoveAt(e.RowIndex);

dataBase.DeletePowerBlock(id);

}

}

}

#endregion

#region Работа с ПУЗ

private void cbStorageDelete\_CheckedChanged(object sender, EventArgs e)

{

if (cbStorageDelete.Checked)

{

dgvStorage.Columns[7].Visible = true;

}

else

{

dgvStorage.Columns[7].Visible = false;

}

}

private void dgvStorage\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

if (e.RowIndex < 0)

{

return;

}

if (dgvStorage.Columns[e.ColumnIndex].Index == dgvStorage.Columns.Count - 1)

{

if (messageBoxClickResult("Удалить эту запись?") == DialogResult.Yes)

{

var id = dgvStorage.Rows[e.RowIndex].Cells[0].Value.ToString();

storageBindingSource.RemoveAt(e.RowIndex);

dataBase.DeleteStorage(id);

}

}

}

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

{

if (string.IsNullOrEmpty(txtStorageTitle.Text))

{

messageBoxError("Вы не ввели название");

return;

}

var storage = new Storage();

storage.manufacturer = cmbStorageManufacturer.SelectedItem.ToString();

storage.title = txtStorageTitle.Text;

storage.volune = (int)cmbStorageVolume.SelectedItem;

storage.speedOfRead = (int)cmbStorageRead.SelectedItem;

storage.speedOfWrite = (int)cmbStorageRead.SelectedItem;

storage.type = cmbStorageType.SelectedItem.ToString();

dataBase.AddStorage(storage);

storageBindingSource.Add(storage);

txtRamTitle.Clear();

messageBoxSuccessAdd();

}

#endregion

private void dgvRam\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

if (e.RowIndex < 0)

{

return;

}

if (dgvRam.Columns[e.ColumnIndex].Index == dgvRam.Columns.Count - 1)

{

if (messageBoxClickResult("Удалить эту запись?") == DialogResult.Yes)

{

var id = dgvRam.Rows[e.RowIndex].Cells[0].Value.ToString();

rAMBindingSource.RemoveAt(e.RowIndex);

dataBase.DeleteRAM(id);

}

}

}

private void cbRamDelete\_CheckedChanged(object sender, EventArgs e)

{

if (cbRamDelete.Checked)

{

dgvRam.Columns[6].Visible = true;

}

else

{

dgvRam.Columns[6].Visible = false;

}

}

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

{

if (string.IsNullOrEmpty(txtRamTitle.Text))

{

messageBoxError("Вы не ввели название");

return;

}

var ram = new RAM();

ram.manufacturer = cmbRamManufacturer.SelectedItem.ToString();

ram.title = txtRamTitle.Text;

ram.volune = (int)cmbRamVolume.SelectedItem;

ram.typeMemory = cmbRamVolume.SelectedItem.ToString();

ram.frequency = (int)cmbRamFrenquency.SelectedItem;

dataBase.AddRAM(ram);

rAMBindingSource.Add(ram);

txtRamTitle.Clear();

messageBoxSuccessAdd();

}

private void nudVideoCardSearch\_ValueChanged(object sender, EventArgs e)

{

var ans = dataBase.VideoCardSearch((int)nudVideoCardSearch.Value);

videoCardBindingSource.Clear();

foreach (var a in ans)

{

videoCardBindingSource.Add(a);

}

}

}

}

3 Работа программы

Ниже представлен внешний вид работающей программы (рис. 1 – 8).

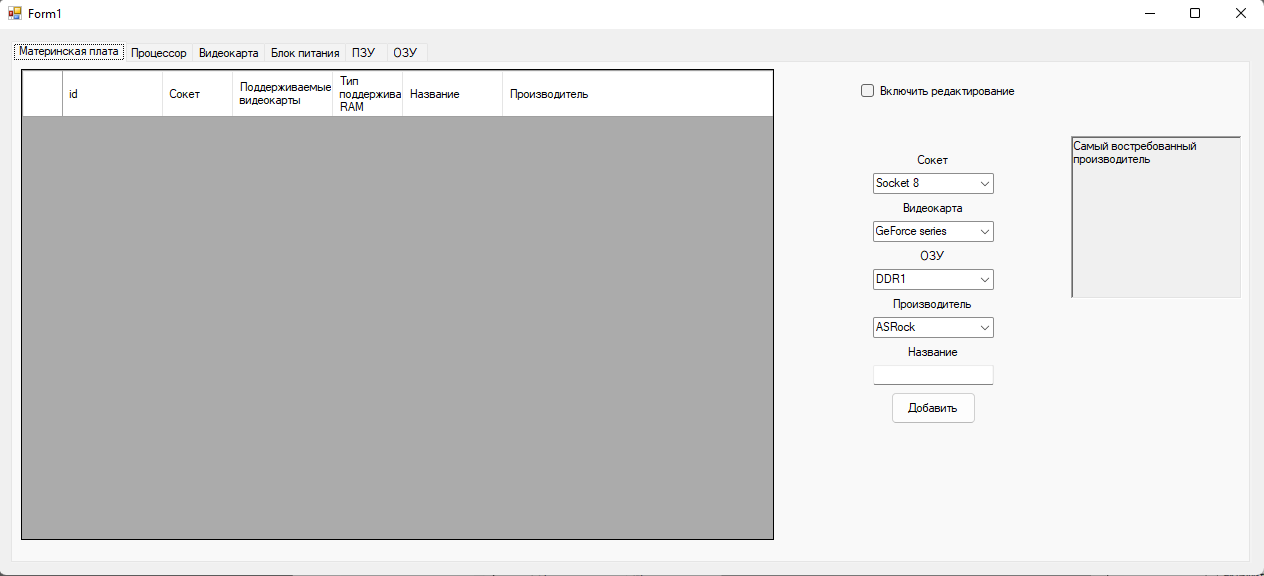


Рисунок 1 – Вкладка «Материнская плата»

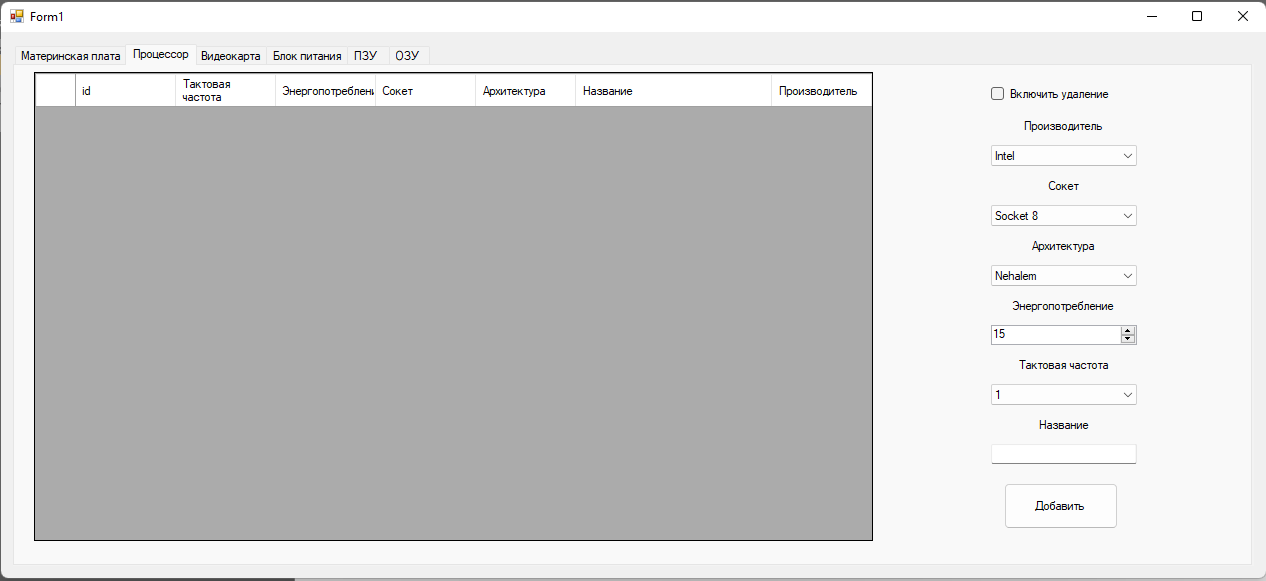


Рисунок 2 – Вкладка «Процессор»

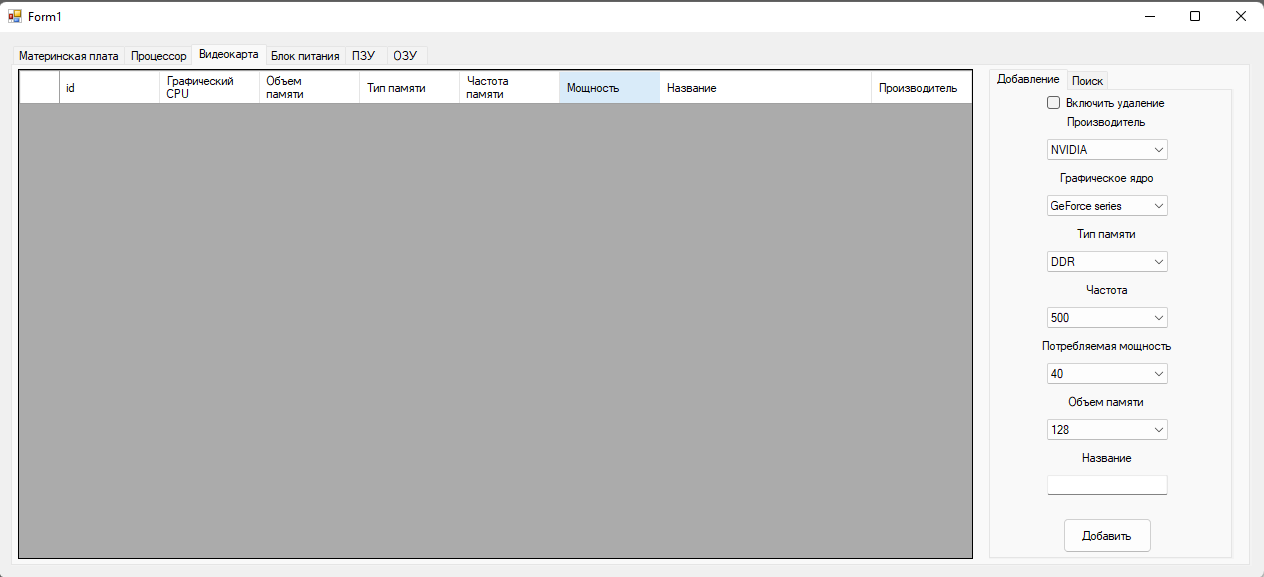


Рисунок 3 – Вкладка «Видеокарта»

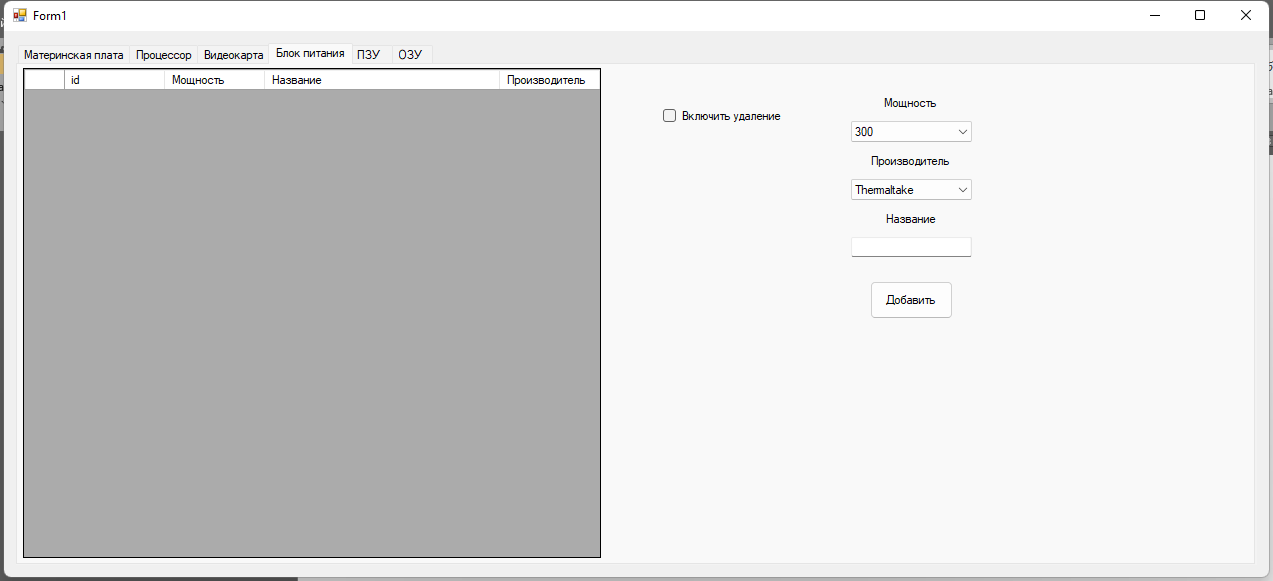


Рисунок 4 – Вкладка «Блок питания»

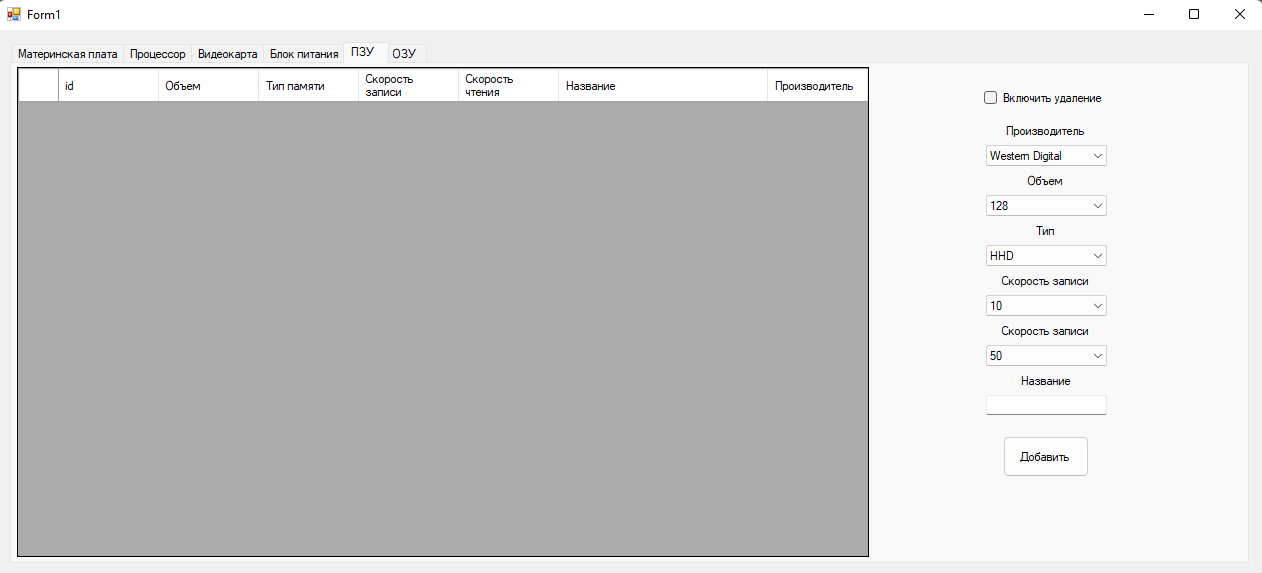


Рисунок 5 – Вкладка «ПЗУ»

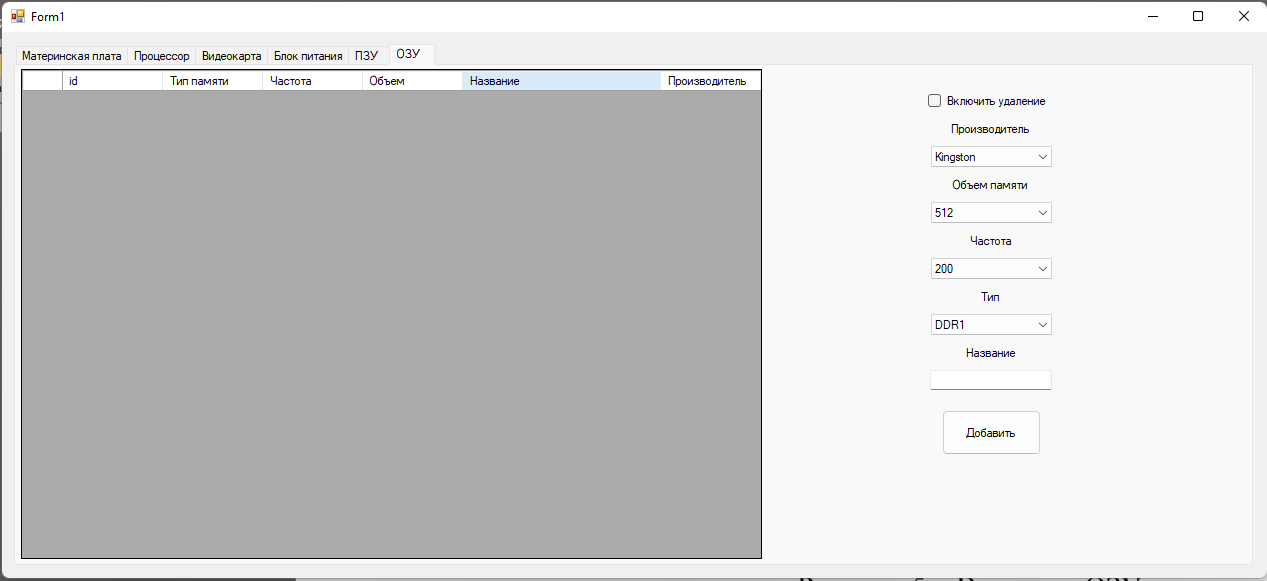


Рисунок 5 – Вкладка «ОЗУ»

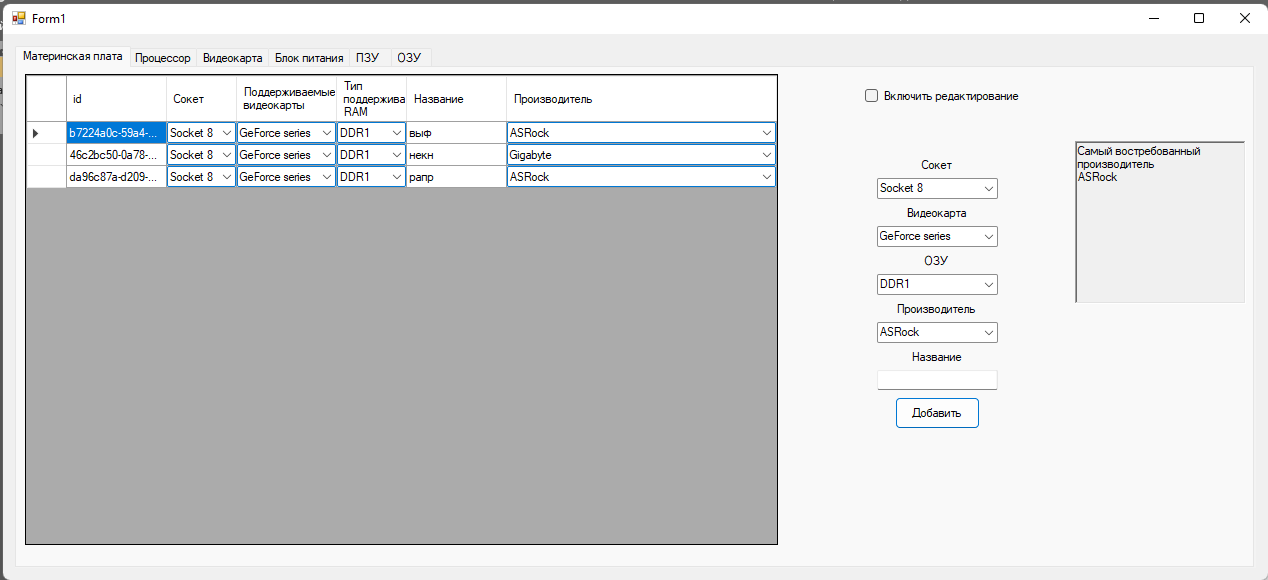


Рисунок 7 – Выдача первого документа

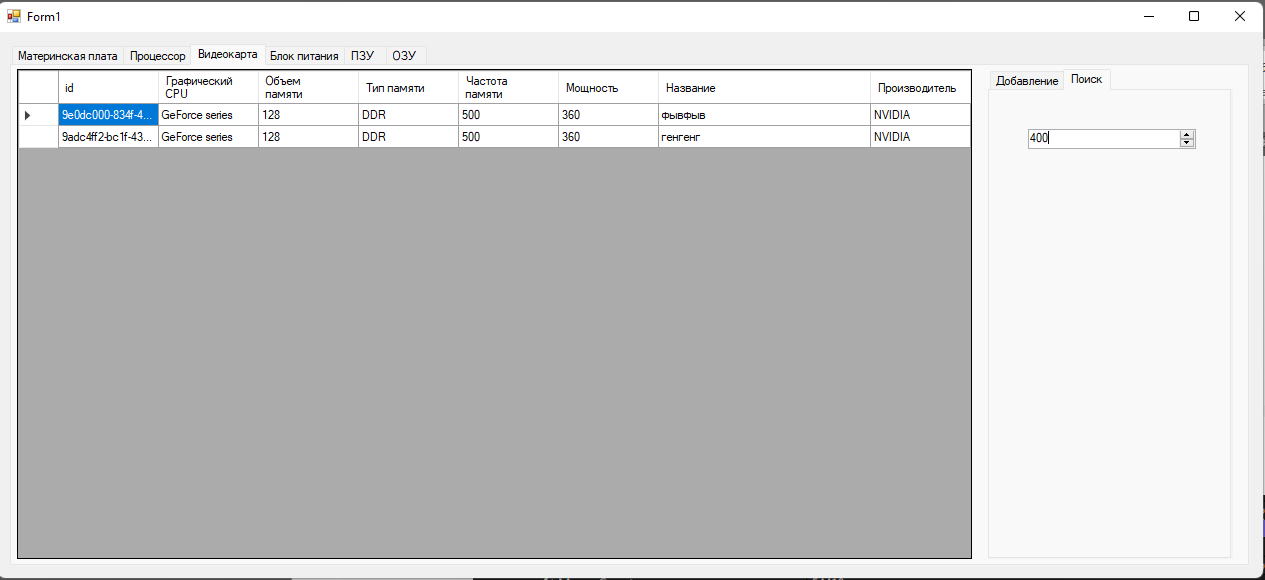


Рисунок 8 – Выдача второго документа