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 №2  по дисциплине «Объектно-ориентированные базы данных»  «Perst» | | |  |

наименование темы

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

Содержание

[1 Задание 3](#_Toc100928321)

[2 Классы 4](#_Toc100928322)

[3 Работа программы 18](#_Toc100928323)

# **1 Задание**

**Автомат с напитками**

Предметная область включает в себя Соки, Газировку, Алкоголь и Торговый автомат.

Необходимо реализовать CRUD функциональность для данной предметной области.

**Выходные документы:**

* Список соков, с набором задаваемых параметров.
* Список алкоголя с заданной крепостью.

# **2 Классы**

В данной лабораторной работе необходимо было разработать 6 классов: Alcohol, Juice, DataBase, Soda, Vendor, IDrink и RootClass на языке C# c использованием библиотеки Perst.

Код классов представлен ниже.

*Интерфейс IDrink:*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DrinkPerst.Classes

{

internal interface IDrink

{

string Id { get; set; }

string Title { get; set; }

uint Volume { get; set; }

}

}

*Класс DataBase:*

using Perst;

using System;

using System.Collections.Generic;

using System.Drawing.Drawing2D;

using System.Linq;

using System.Runtime.InteropServices;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace DrinkPerst.Classes

{

internal class DataBase : Persistent

{

private Storage db;

public long PagePoolSize { get; }

private const string FilePatch = @"MyDB.pdb";

public DataBase()

{

db = StorageFactory.Instance.CreateStorage();

PagePoolSize = 10;

}

public void Connect()

{

db.Open(FilePatch, PagePoolSize);

var root = (RootClass)db.Root;

if (root == null)

{

root = new RootClass(db);

db.Root = root;

}

Console.WriteLine("Connected!");

}

public void Close()

{

db.Close();

}

#region Работа с соком

public List<string> IsHaveJuice(string id)

{

var found = ((db.Root as RootClass).VendorId).Cast<Vendor>().ToList();

var response = from vendors in found

from juices in vendors.Juices.Cast<Juice>().ToList()

where juices.Id == id

group juices by vendors.Title into r

select new

{

Title = r.Key

}; ;

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

foreach (var res in response)

{

values.Add(res.Title);

}

return values;

}

public void AddJuice(Juice juice)

{

(db.Root as RootClass).JuiceId.Put(juice);

}

public void DeleteJuice(string id)

{

var ans = IsHaveJuice(id);

if(ans.Count > 0)

{

var message = "Сок все еще используется в автоматах:\n";

foreach (var a in ans)

{

message += $"{a}\n";

}

throw new Exception(message);

}

var found = (db.Root as RootClass).JuiceId[id] as Juice;

(db.Root as RootClass).JuiceId.Remove(found);

}

public void UpdateJuice(Juice juice)

{

var found = (db.Root as RootClass).JuiceId[juice.Id] as Juice;

found.Taste = juice.Taste;

found.Title = juice.Title;

found.IsPulp = juice.IsPulp;

found.Volume = juice.Volume;

found.Modify();

}

public List<Juice> GetJuices() => (db.Root as RootClass).JuiceId

.Cast<Juice>().ToList();

public List<Juice> JuiceSeatch(Juice juice)

{

var response = (db.Root as RootClass).JuiceId.Select(typeof(Juice),

$"Title like '%{juice.Title}%' and " +

$"Volume >= {juice.Volume} and " +

$"PrintJuiceFruit = '{juice.PrintJuiceFruit()}' and " +

$"IsPulp = {juice.IsPulp}").Cast<Juice>().ToList(); ;

return response;

}

#endregion

#region Работа с газировкой

public void AddSoda(Soda soda)

{

(db.Root as RootClass).SodaId.Put(soda);

}

public List<string> IsHaveSoda(string id)

{

var found = ((db.Root as RootClass).VendorId).Cast<Vendor>().ToList();

var response = from vendors in found

from sodas in vendors.Sodas.Cast<Soda>().ToList()

where sodas.Id == id

group sodas by vendors.Title into r

select new

{

Title = r.Key

}; ;

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

foreach (var res in response)

{

values.Add(res.Title);

}

return values;

}

public void DeleteSoda(string id)

{

var ans = IsHaveSoda(id);

if(ans.Count > 0)

{

var message = "Газировка все еще используется в автоматах:\n";

foreach (var a in ans)

{

message += $"{a}\n";

}

throw new Exception(message);

}

var found = (db.Root as RootClass).SodaId[id] as Soda;

(db.Root as RootClass).SodaId.Remove(found);

}

public List<Soda> GetSodas() => (db.Root as RootClass).SodaId.Cast<Soda>().ToList();

#endregion

#region Работа с алкоголем

public void AddAlcohol(Alcohol alcohol)

{

(db.Root as RootClass).AlcoholId.Put(alcohol);

(db.Root as RootClass).FortressAlco.Put(alcohol);

}

public void DeleteAlcohol(string id)

{

var ans = IsHaveAlcohol(id);

if (ans.Count > 0)

{

var message = "Алкоголь все еще используется в автоматах:\n";

foreach (var a in ans)

{

message += $"{a}\n";

}

throw new Exception(message);

}

var found = (db.Root as RootClass).AlcoholId[id] as Alcohol;

(db.Root as RootClass).AlcoholId.Remove(found);

}

public List<string> IsHaveAlcohol(string id)

{

var found = ((db.Root as RootClass).VendorId).Cast<Vendor>().ToList();

var response = from vendors in found

from alco in vendors.Alcohols.Cast<Alcohol>().ToList()

where alco.Id == id

group alco by vendors.Title into r

select new

{

Title = r.Key

}; ;

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

foreach (var res in response)

{

values.Add(res.Title);

}

return values;

}

public List<Alcohol> GetAlcohols() => ((db.Root as RootClass).AlcoholId).Cast<Alcohol>().ToList();

public List<Alcohol> GetAlcohols(uint from, uint to)

{

Key key1 = new Key(from);

Key key2 = new Key(to);

var alcohols = (db.Root as RootClass).FortressAlco.Range(

key1, key2,

IterationOrder.DescentOrder).Cast<Alcohol>().ToList();

return alcohols;

}

#endregion

#region Работа с автоматами

public List<Vendor> GetVendors() => (db.Root as RootClass).VendorId.Cast<Vendor>().ToList();

public Vendor GetVendors(string id) => (db.Root as RootClass).VendorId[id] as Vendor;

public void DeleteVendor(string id)

{

(db.Root as RootClass).VendorId.Remove(GetVendors(id));

}

public Vendor AddVendor(string title, List<Juice> juices,

List<Soda> sodas, List<Alcohol> alcohols)

{

Vendor vendor = new Vendor();

vendor.Title = title;

vendor.Juices = db.CreateLink();

foreach(var j in juices)

{

vendor.Juices.Add(j);

vendor.Modify();

}

vendor.Sodas = db.CreateLink();

foreach (var s in sodas)

{

vendor.Sodas.Add(s);

vendor.Modify();

}

vendor.Alcohols = db.CreateLink();

foreach (var a in alcohols)

{

vendor.Alcohols.Add(a);

vendor.Modify();

}

(db.Root as RootClass).VendorId.Put(vendor);

return vendor;

}

public void UpdateVendor(string id, string title, List<Juice> juices,

List<Soda> sodas, List<Alcohol> alcohols)

{

Vendor vendor = (db.Root as RootClass).VendorId[id] as Vendor;

vendor.Juices.Clear();

vendor.Sodas.Clear();

vendor.Alcohols.Clear();

foreach (var j in juices)

{

vendor.Juices.Add(j);

vendor.Modify();

}

foreach (var s in sodas)

{

vendor.Sodas.Add(s);

vendor.Modify();

}

foreach (var a in alcohols)

{

vendor.Alcohols.Add(a);

vendor.Modify();

}

vendor.Title = title;

vendor.Modify();

}

#endregion

}

}

*Класс Juice:*

using Perst;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DrinkPerst.Classes

{

internal class Juice : Persistent, IDrink

{

[DisplayName("Id")]

public string Id { get; set; }

[DisplayName("Название")]

public string Title { get; set; }

[DisplayName("Объем")]

public uint Volume { get; set; }

[DisplayName("Вкус")]

public JuiceFruit Taste { get; set; }

[DisplayName("Мякоть")]

public bool IsPulp { get; set; }

public Juice()

{

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

Title = "title";

Volume = 500;

Taste = JuiceFruit.Apple;

IsPulp = true;

}

public Juice(string title, uint volume, JuiceFruit taste, bool isPulp)

{

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

Title = title;

Volume = volume;

Taste = taste;

IsPulp = isPulp;

}

public override bool Equals(object obj)

{

return obj is Juice juice &&

Title == juice.Title &&

Volume == juice.Volume &&

Taste == juice.Taste &&

IsPulp == juice.IsPulp;

}

public string PrintJuiceFruit()

{

var taste = "";

switch ((int)Taste)

{

case 0:

{

taste = "Яблоко";

}

break;

case 1:

{

taste = "Апельсин";

}

break;

case 2:

{

taste = "Банан";

}

break;

case 3:

{

taste = "Арбуз";

}

break;

case 4:

{

taste = "Мультифрукт";

}

break;

}

return taste;

}

public override string ToString()

{

return $"Id: {Id}\nНазвание: {Title}\nОбъем: {Volume}\n" +

$"Вкус: {PrintJuiceFruit()}\nНаличие мякоти: { (IsPulp? "Есть" : "Нет")}";

}

}

public enum JuiceFruit : int

{

Apple = 0,

Orange = 1,

Banana = 2,

Watermelon = 3,

Multifruit = 4

}

}

*Класс RootClass:*

using Perst;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DrinkPerst.Classes

{

internal class RootClass: Persistent

{

public FieldIndex AlcoholId;

public FieldIndex SodaId;

public FieldIndex JuiceId;

public FieldIndex VendorId;

public FieldIndex FortressAlco;

public RootClass(Storage db) : base(db)

{

AlcoholId = db.CreateFieldIndex(typeof(Alcohol), "Id", true);

SodaId = db.CreateFieldIndex(typeof(Soda), "Id", true);

JuiceId = db.CreateFieldIndex(typeof(Juice), "Id", true);

VendorId = db.CreateFieldIndex(typeof(Vendor), "Id", true);

FortressAlco = db.CreateFieldIndex(typeof(Alcohol), "Fortress", false);

}

}

}

*Класс Soda:*

using Perst;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DrinkPerst.Classes

{

internal class Soda: Persistent, IDrink

{

[DisplayName("Id")]

public string Id { get; set; }

[DisplayName("Название")]

public string Title { get; set; }

[DisplayName("Объем")]

public uint Volume { get; set; }

[DisplayName("Тип")]

public SodaType Type { get; set; }

[DisplayName("Кол-во пузырьков")]

public uint BubbleCount { get; set; }

public Soda()

{

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

Title = "title";

Volume = 500;

Type = SodaType.Low;

BubbleCount = 100;

}

public Soda(string title, uint volume, SodaType type, uint bubbleCount)

{

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

Title = title;

Volume = volume;

Type = type;

BubbleCount = bubbleCount;

}

public override bool Equals(object obj)

{

return obj is Soda soda &&

Title == soda.Title &&

Volume == soda.Volume &&

Type == soda.Type &&

BubbleCount == soda.BubbleCount;

}

public string PrintSodaType()

{

var type = "";

switch ((int)Type)

{

case 0:

{

type = "Слабогазированная";

}

break;

case 1:

{

type = "Среднегазированная";

}

break;

case 2:

{

type = "Сильногазированная";

}

break;

}

return type;

}

public override string ToString()

{

return $"Id: {Id}\nНазвание: {Title}\nОбъем: {Volume}\n" +

$"Тип: {PrintSodaType()}\nКол-во пузырьков: {BubbleCount}";

}

}

public enum SodaType: int

{

Low = 0,

Mid = 1,

High = 2

}

}

*Класс Alcohol:*

using Perst;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using static System.Windows.Forms.AxHost;

namespace DrinkPerst.Classes

{

internal class Alcohol: Persistent, IDrink

{

[DisplayName("Id")]

public string Id { get; set; }

[DisplayName("Название")]

public string Title { get; set; }

[DisplayName("Объем")]

public uint Volume { get; set; }

[DisplayName("Тип")]

public AlcoholType Type { get; set; }

[DisplayName("Крепость")]

public uint Fortress { get; set; }

public Alcohol()

{

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

Title = "title";

Volume = 500;

Type = AlcoholType.Gin;

Fortress = 45;

}

public Alcohol(string title, uint volume, AlcoholType type, uint fortress)

{

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

Title = title;

Volume = volume;

Type = type;

Fortress = fortress;

}

public string PrintAlcoholType()

{

var type = "";

switch ((int)Type)

{

case 0:

{

type = "Виски";

}

break;

case 1:

{

type = "Коньяк";

}

break;

case 2:

{

type = "Пиво";

}

break;

case 3:

{

type = "Вино";

}

break;

case 4:

{

type = "Джин";

}

break;

}

return type;

}

public override bool Equals(object obj)

{

return obj is Alcohol alcohol &&

Title == alcohol.Title &&

Volume == alcohol.Volume &&

Type == alcohol.Type &&

Fortress == alcohol.Fortress;

}

public override string ToString()

{

return $"Id: {Id}\nНазвание: {Title}\nОбъем: {Volume}\n" +

$"Тип: {PrintAlcoholType()}\nКрепость: {Fortress}";

}

}

public enum AlcoholType: int {

Whiskey = 0,

Cognac = 1,

Beer = 2,

Wine = 3,

Gin = 4

}

}

*Класс Vendor:*

using Perst;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DrinkPerst.Classes

{

internal class Vendor : Persistent

{

[DisplayName("Id")]

public string Id { get; set; }

[DisplayName("Название")]

public string Title { get; set; }

public Link Alcohols { get; set; }

public Link Sodas { get; set; }

public Link Juices { get; set; }

public Vendor()

{

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

}

}

}

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

Листинг Form1:

using DrinkPerst.Classes;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Runtime.CompilerServices;

using System.Runtime.InteropServices;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace DrinkPerst

{

/\*

\* Идеи для поисков: Напитки определенной крепости

\*

\*/

public partial class Form1 : Form

{

bool isLoad = true;

private void Form1\_FormClosing(object sender, FormClosingEventArgs e)

{

dataBase.Close();

}

DataBase dataBase;

public Form1()

{

InitializeComponent();

dataBase = new DataBase();

dataBase.Connect();

SourseSelect();

CombosInit();

TableInit();

isLoad = false;

}

private void SourseSelect()

{

foreach(var juice in dataBase.GetJuices())

{

juiceBindingSource.Add(juice);

}

foreach(var soda in dataBase.GetSodas())

{

sodaBindingSource.Add(soda);

}

foreach (var alco in dataBase.GetAlcohols())

{

alcoholBindingSource.Add(alco);

}

foreach(var vendor in dataBase.GetVendors())

{

vendorBindingSource.Add(vendor);

}

}

private void CombosInit()

{

(dgvJuice.Columns[3] as DataGridViewComboBoxColumn).DataSource =

Enum.GetValues(typeof(JuiceFruit));

cmbJuiceTaste.DataSource = Enum.GetValues(typeof(JuiceFruit));

cmbJuiceSearchTaste.DataSource= Enum.GetValues(typeof(JuiceFruit));

cmbSodaType.DataSource = Enum.GetValues(typeof(SodaType));

cmbAlcoType.DataSource = Enum.GetValues(typeof(AlcoholType));

rtbVendorJuice.Text = (cmbVendorJuice.SelectedItem as Juice)?.ToString();

rtbVendorSoda.Text = (cmbVendorSoda.SelectedItem as Soda)?.ToString();

rtbVendorAlcohol.Text = (cmbVendorAlcohol.SelectedItem as Alcohol)?.ToString();

}

private void TableInit()

{

if(vendorBindingSource.Count > 0)

{

var vendor = dataBase.GetVendors(dgvVendor.SelectedCells[0].Value.ToString());

var juices = vendor.Juices.ToArray();

var sodas = vendor.Sodas.ToArray();

var alcos = vendor.Alcohols.ToArray();

foreach(var j in juices)

{

vendorJuiceBindingSource.Add(j);

}

foreach (var s in sodas)

{

vendorSodaBindingSource.Add(s);

}

foreach (var a in alcos)

{

vendorAlcoholBindingSource.Add(a);

}

txtVendorTitle.Text = vendor.Title;

}

}

#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;

}

//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 соком

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

{

if (cbJuice.Checked)

{

dgvJuice.Columns[1].ReadOnly = false;

dgvJuice.Columns[2].ReadOnly = false;

dgvJuice.Columns[3].ReadOnly = false;

dgvJuice.Columns[4].ReadOnly = false;

dgvJuice.Columns[5].Visible = true;

}

else

{

dgvJuice.Columns[1].ReadOnly = true;

dgvJuice.Columns[2].ReadOnly = true;

dgvJuice.Columns[3].ReadOnly = true;

dgvJuice.Columns[4].ReadOnly = true;

dgvJuice.Columns[5].Visible = false;

}

}

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

{

if (e.RowIndex < 0)

{

return;

}

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

{

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

{

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

try

{

dataBase.DeleteJuice(id);

juiceBindingSource.RemoveAt(e.RowIndex);

}

catch (Exception ex)

{

messageBoxError(ex.Message);

}

}

}

}

Juice selectedJuice;

private void dgvJuice\_CellClick(object sender, DataGridViewCellEventArgs e)

{

if (e.RowIndex < 0)

{

return;

}

selectedJuice = new Juice();

selectedJuice.Id = dgvJuice.Rows[e.RowIndex].Cells[0].Value.ToString();

selectedJuice.Title = dgvJuice.Rows[e.RowIndex].Cells[1].Value.ToString();

selectedJuice.Volume = (uint)dgvJuice.Rows[e.RowIndex].Cells[2].Value;

selectedJuice.Taste = (JuiceFruit)dgvJuice.Rows[e.RowIndex].Cells[3].Value;

selectedJuice.IsPulp = (bool)dgvJuice.Rows[e.RowIndex].Cells[4].Value;

}

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

{

uint volume = (uint)dgvJuice.Rows[e.RowIndex].Cells[2].Value;

if(volume < nudJuiceVolume.Minimum || volume > nudJuiceVolume.Maximum)

{

dgvJuice.Rows[e.RowIndex].Cells[2].Value = selectedJuice.Volume;

messageBoxError("Введите корретное значение объема");

return;

}

var juice = new Juice();

juice.Id = dgvJuice.Rows[e.RowIndex].Cells[0].Value.ToString();

juice.Title = dgvJuice.Rows[e.RowIndex].Cells[1].Value.ToString();

juice.Volume = volume;

juice.Taste = (JuiceFruit)dgvJuice.Rows[e.RowIndex].Cells[3].Value;

juice.IsPulp = (bool)dgvJuice.Rows[e.RowIndex].Cells[4].Value;

if (juice.Equals(selectedJuice))

{

return;

}

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

if(dialogResult == DialogResult.No)

{

dgvJuice[1, e.RowIndex].Value = juice.Title;

dgvJuice[2, e.RowIndex].Value = juice.Volume;

dgvJuice[3, e.RowIndex].Value = juice.Taste;

dgvJuice[4, e.RowIndex].Value = juice.IsPulp;

}

else

{

dataBase.UpdateJuice(dgvJuice.CurrentRow.DataBoundItem as Juice);

}

}

private void dgvJuice\_DataError(object sender, DataGridViewDataErrorEventArgs e)

{

dgvJuice.Rows[e.RowIndex].Cells[2].Value = selectedJuice.Volume;

messageBoxError("Введите корретное значение объема");

}

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

{

if (string.IsNullOrEmpty(txtJuiceTitle.Text))

{

messageBoxError("Вы не ввели название");

return;

}

var juice = new Juice();

juice.Title = txtJuiceTitle.Text;

juice.Taste = (JuiceFruit)cmbJuiceTaste.SelectedItem;

juice.IsPulp = cbJuicePulp.Checked;

juice.Volume = (uint)nudJuiceVolume.Value;

dataBase.AddJuice(juice);

txtJuiceTitle.Clear();

juiceBindingSource.Add(juice);

messageBoxSuccessAdd();

}

#endregion

#region Работа с газировкой

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

{

dgvSoda.Columns[5].Visible = cbSoda.Checked ? true : false;

}

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

{

if (e.RowIndex < 0)

{

return;

}

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

{

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

{

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

try

{

dataBase.DeleteSoda(id);

sodaBindingSource.RemoveAt(e.RowIndex);

}

catch (Exception ex)

{

messageBoxError(ex.Message);

}

}

}

}

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

{

if (string.IsNullOrEmpty(txtSodaTitle.Text))

{

messageBoxError("Вы не ввели название");

return;

}

var soda = new Soda();

soda.Title = txtSodaTitle.Text;

soda.Volume = (uint)nudSodaVolume.Value;

soda.BubbleCount = (uint)nudSodaBubble.Value;

soda.Type = (SodaType)cmbSodaType.SelectedItem;

dataBase.AddSoda(soda);

txtSodaTitle.Clear();

sodaBindingSource.Add(soda);

messageBoxSuccessAdd();

}

#endregion

#region Работа с алкоголем

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

{

dgvAlcohol.Columns[5].Visible = cbAlco.Checked ? true : false;

}

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

{

if (e.RowIndex < 0)

{

return;

}

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

{

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

{

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

try

{

dataBase.DeleteAlcohol(id);

alcoholBindingSource.RemoveAt(e.RowIndex);

}

catch (Exception ex)

{

messageBoxError(ex.Message);

}

}

}

}

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

{

if (string.IsNullOrEmpty(txtAlcoTitle.Text))

{

messageBoxError("Вы не ввели название");

return;

}

var alco = new Alcohol();

alco.Title = txtAlcoTitle.Text;

alco.Volume = (uint)nudAlcoVolume.Value;

alco.Fortress = (uint)nudAlcoFortress.Value;

alco.Type = (AlcoholType)cmbAlcoType.SelectedItem;

dataBase.AddAlcohol(alco);

txtAlcoTitle.Clear();

alcoholBindingSource.Add(alco);

messageBoxSuccessAdd();

}

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

{

nudAlcoSearchTo.Minimum = nudAlcoSearchFrom.Value + 1;

alcoholBindingSource.Clear();

foreach (var alco in dataBase.GetAlcohols((uint)nudAlcoSearchFrom.Value, (uint)nudAlcoSearchTo.Value))

{

alcoholBindingSource.Add(alco);

}

}

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

{

nudAlcoSearchFrom.Maximum = nudAlcoSearchTo.Value - 1;

alcoholBindingSource.Clear();

foreach (var alco in dataBase.GetAlcohols((uint)nudAlcoSearchFrom.Value, (uint)nudAlcoSearchTo.Value))

{

alcoholBindingSource.Add(alco);

}

}

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

{

alcoholBindingSource.Clear();

foreach (var alco in dataBase.GetAlcohols())

{

alcoholBindingSource.Add(alco);

}

}

#endregion

#region Работа с автоматом

private void cmbVendorJuice\_SelectedIndexChanged(object sender, EventArgs e)

{

rtbVendorJuice.Text = (cmbVendorJuice.SelectedItem as Juice)?.ToString();

}

private void cmbVendorSoda\_SelectedIndexChanged(object sender, EventArgs e)

{

rtbVendorSoda.Text = (cmbVendorSoda.SelectedItem as Soda)?.ToString();

}

private void cmbVendorAlcohol\_SelectedIndexChanged(object sender, EventArgs e)

{

rtbVendorAlcohol.Text = (cmbVendorAlcohol.SelectedItem as Alcohol)?.ToString();

}

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

{

if(juiceBindingSource.Count < 1)

{

messageBoxError("Не добавлены соки");

return;

}

var juice = (cmbVendorJuice.SelectedItem as Juice);

vendorJuiceBindingSource.Add(juice);

}

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

{

if (sodaBindingSource.Count < 1)

{

messageBoxError("Не добавлены газировки");

return;

}

var soda = (cmbVendorSoda.SelectedItem as Soda);

vendorSodaBindingSource.Add(soda);

}

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

{

if (alcoholBindingSource.Count < 1)

{

messageBoxError("Не добавлен алкоголь");

return;

}

var alco = (cmbVendorAlcohol.SelectedItem as Alcohol);

vendorAlcoholBindingSource.Add(alco);

}

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

{

if (e.RowIndex < 0)

{

return;

}

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

{

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

{

vendorJuiceBindingSource.RemoveAt(e.RowIndex);

}

}

}

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

{

if (e.RowIndex < 0)

{

return;

}

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

{

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

{

vendorSodaBindingSource.RemoveAt(e.RowIndex);

}

}

}

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

{

if (e.RowIndex < 0)

{

return;

}

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

{

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

{

vendorAlcoholBindingSource.RemoveAt(e.RowIndex);

}

}

}

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

{

dgvVendor.Columns[2].Visible = cbVendor.Checked;

btnVendorAdd.Text = cbVendor.Checked ? "Изменить автомат" : "Добавить автомат";

}

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

{

vendorAlcoholBindingSource.Clear();

vendorJuiceBindingSource.Clear();

vendorSodaBindingSource.Clear();

txtVendorTitle.Clear();

dgvVendor.ClearSelection();

dgvVendor.CurrentCell = null;

}

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

{

if(vendorSodaBindingSource.Count == 0

&& vendorJuiceBindingSource.Count == 0

&& vendorAlcoholBindingSource.Count == 0)

{

messageBoxError("В автомате должен быть хотя бы 1 напиток");

return;

}

if (string.IsNullOrEmpty(txtVendorTitle.Text))

{

messageBoxError("Вы не ввели название");

return;

}

if (cbVendor.Checked)

{

if (dgvVendor.CurrentCell == null)

{

messageBoxError("Вы не выбрали Автомат");

return;

}

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

if (result == DialogResult.Yes)

{

var id = dgvVendor.CurrentRow.Cells[0].Value.ToString();

List<Juice> juices = vendorJuiceBindingSource.Cast<Juice>().ToList();

List<Soda> sodas = vendorSodaBindingSource.Cast<Soda>().ToList();

List<Alcohol> alcos = vendorAlcoholBindingSource.Cast<Alcohol>().ToList();

dataBase.UpdateVendor(id, txtVendorTitle.Text, juices, sodas, alcos);

dgvVendor.CurrentRow.Cells[1].Value = txtVendorTitle.Text;

MessageBox.Show("Запись успешно измененена",

"Успех", MessageBoxButtons.OK,

MessageBoxIcon.Information);

}

}

else

{

List<Juice> juices = vendorJuiceBindingSource.Cast<Juice>().ToList();

List<Soda> sodas = vendorSodaBindingSource.Cast<Soda>().ToList();

List<Alcohol> alcos = vendorAlcoholBindingSource.Cast<Alcohol>().ToList();

var vendor = dataBase.AddVendor(txtVendorTitle.Text,

juices, sodas, alcos);

vendorBindingSource.Add(vendor);

txtVendorTitle.Clear();

vendorAlcoholBindingSource.Clear();

vendorJuiceBindingSource.Clear();

vendorSodaBindingSource.Clear();

messageBoxSuccessAdd();

}

}

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

{

if (e.RowIndex < 0)

{

return;

}

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

{

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

{

var id = (string)dgvVendor.Rows[e.RowIndex].Cells[0].Value;

vendorBindingSource.RemoveAt(e.RowIndex);

vendorAlcoholBindingSource.Clear();

vendorJuiceBindingSource.Clear();

vendorSodaBindingSource.Clear();

txtVendorTitle.Clear();

dataBase.DeleteVendor(id);

}

}

else

{

vendorAlcoholBindingSource.Clear();

vendorJuiceBindingSource.Clear();

vendorSodaBindingSource.Clear();

TableInit();

}

}

#endregion

private void JuiceSearch()

{

if (isLoad)

{

return;

}

var juice = new Juice();

juice.Title = txtJuiceSearchTitle.Text;

juice.Volume = (uint)nudJuiceSearchVolume.Value;

juice.IsPulp = cbJuiceSearchPulp.Checked;

juice.Taste = (JuiceFruit)cmbJuiceSearchTaste.SelectedItem;

juiceBindingSource.Clear();

foreach (var j in dataBase.JuiceSeatch(juice))

{

juiceBindingSource.Add(j);

}

}

private void txtJuiceSearchTitle\_TextChanged(object sender, EventArgs e)

{

JuiceSearch();

}

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

{

JuiceSearch();

}

private void cmbJuiceSearchTaste\_SelectedIndexChanged(object sender, EventArgs e)

{

JuiceSearch();

}

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

{

JuiceSearch();

}

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

{

txtJuiceSearchTitle.Clear();

juiceBindingSource.Clear();

foreach (var juice in dataBase.GetJuices())

{

juiceBindingSource.Add(juice);

}

}

}

}

# **3 Работа программы**

Ниже представлен внешний вид работающей программы (рис. 1 – 6).

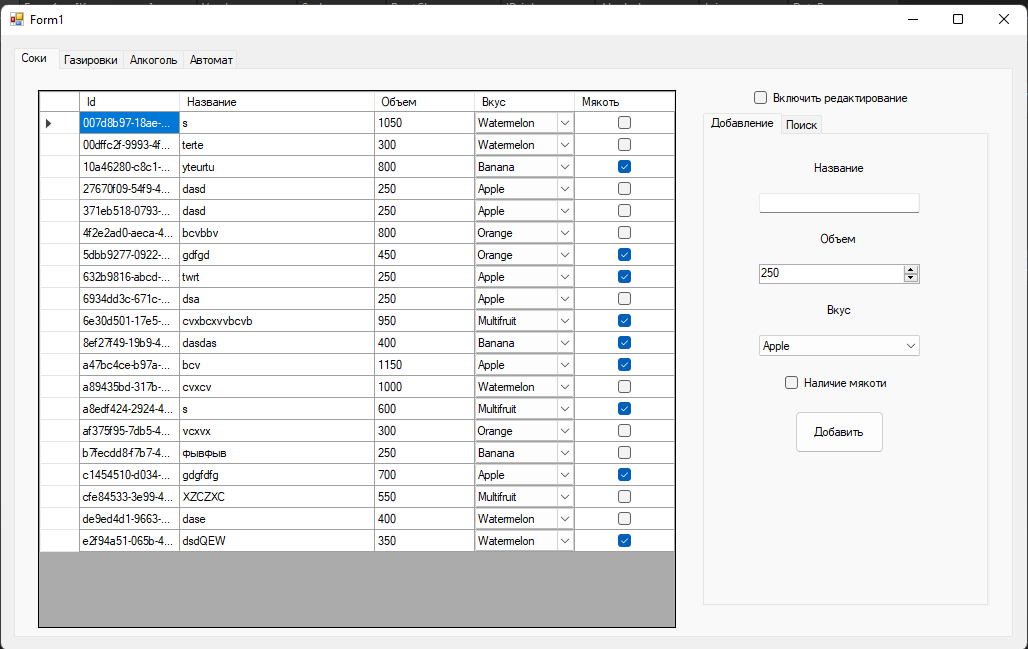


Рисунок 1 – Вкладка «Соки»

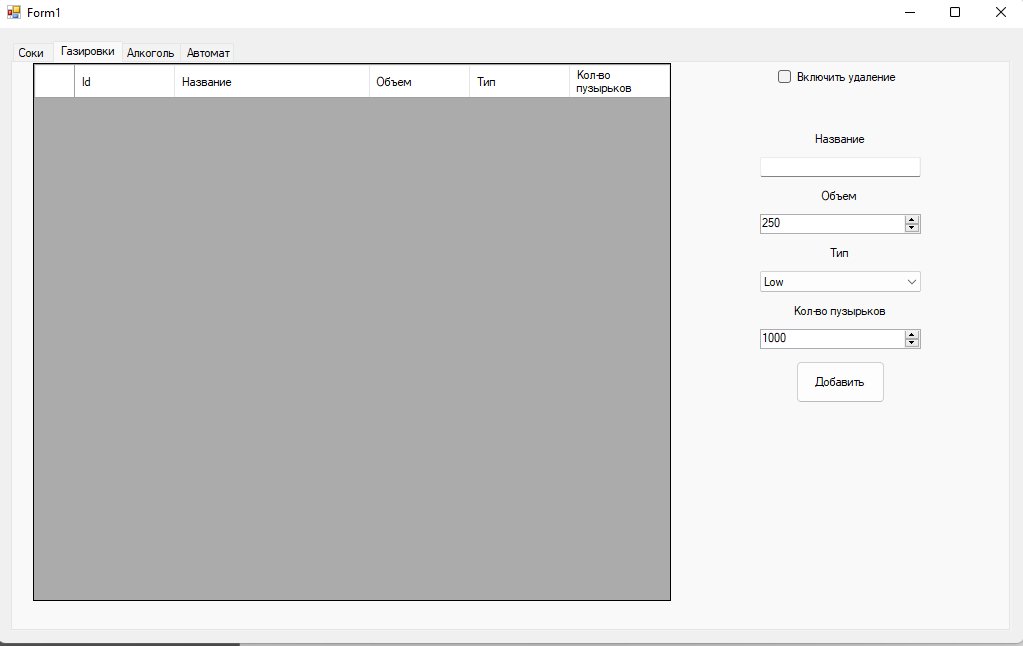


Рисунок 2 – Вкладка «Газировки»

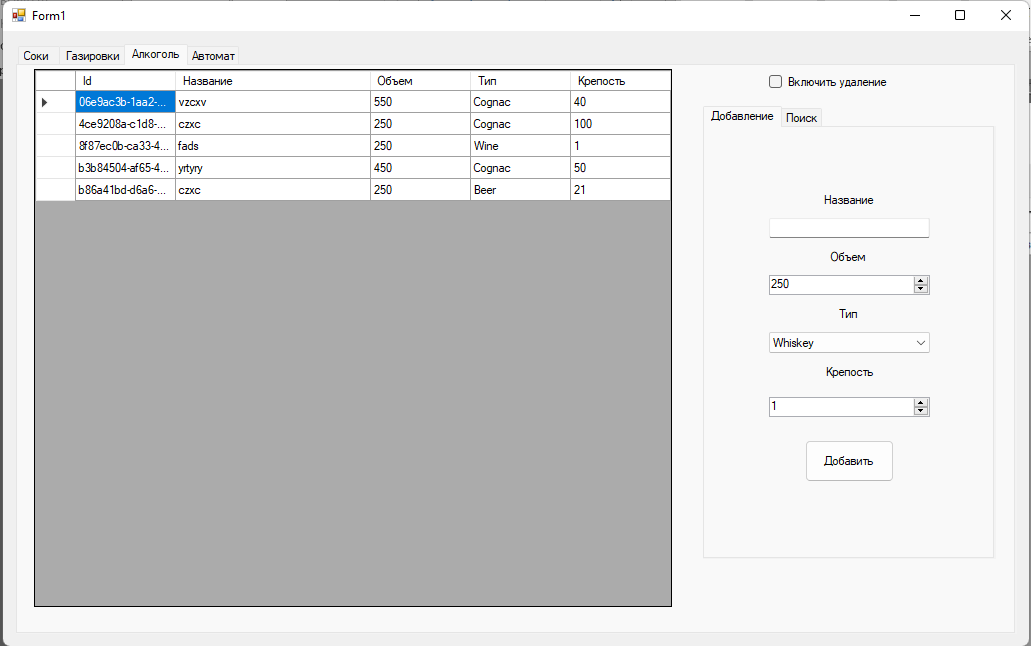


Рисунок 3 – Вкладка «Алкоголь»

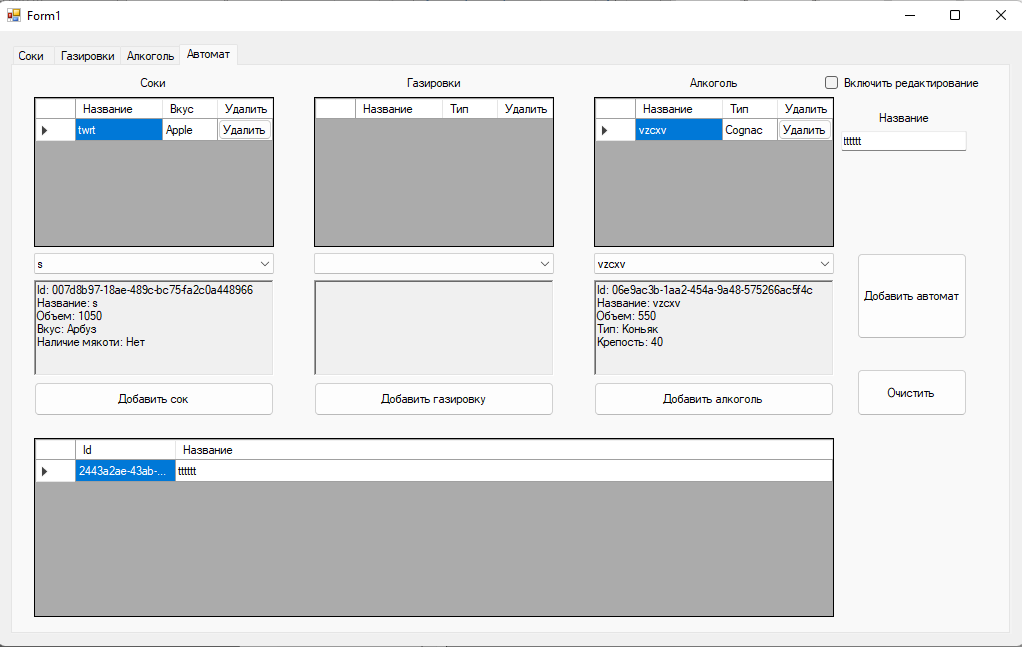


Рисунок 4 – Вкладка «Автомат»

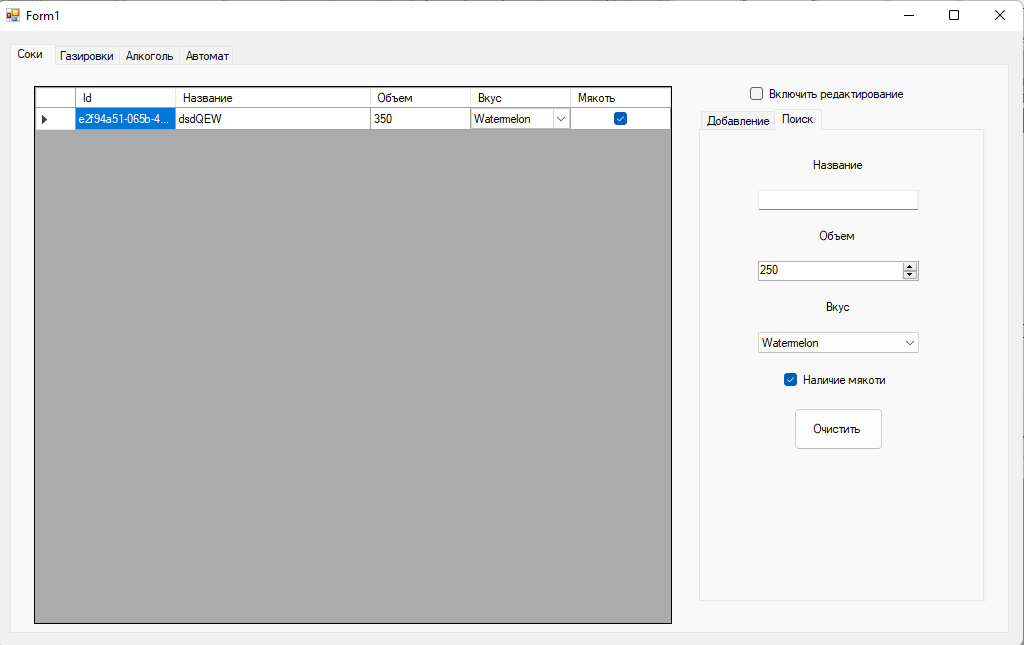


Рисунок 5 – Выдача первого документа

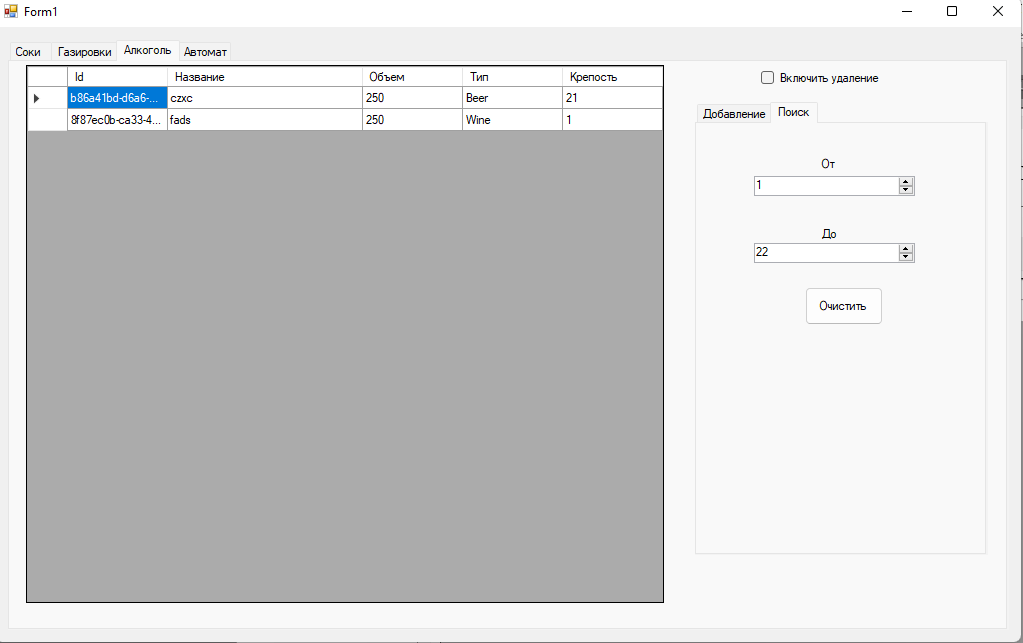


Рисунок 6 – Выдача второго документа