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 | |  |  |  | Михейко Ю.А. |
| 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**

**Выставка животных**

Предметная область включает в себя Котов, Собак, Птиц и Выставку.

Необходимо реализовать CRUD функциональность для данной предметной области.

**Выходные документы:**

* Список котов, чей возраст находится в промежутке.
* Список всех собак конкретной породы, чья кличка содержит в себе заданные символы.

# **2 Классы**

В данной лабораторной работе необходимо было разработать 6 классов: Bird, Cat, DataBase, Dog, Exhibition, IAnimal и RootClass на языке C# c использованием библиотеки Perst.

Код классов представлен ниже.

*Интерфейс IAnimal:*

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace CatPerst.Classes

{

internal interface IAnimal

{

[DisplayName("Id")]

string Id { get; set; }

[DisplayName("Кличка")]

string Name { get; set; }

[DisplayName("Возраст")]

int Age { get; set; }

[DisplayName("Цвет")]

AnimalColor Color { get; set; }

string PrintBreed();

string PrintColor();

}

}

*Класс DataBase:*

using Perst;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.InteropServices;

using System.Text;

using System.Threading.Tasks;

namespace CatPerst.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 котами

public List<string> IsHaveCat(string id)

{

var found = ((db.Root as RootClass).ExhibitionId);

var response = from exhibitions in found.Cast<Exhibition>().ToList()

from cats in exhibitions.Cats.Cast<Cat>().ToList()

where cats.Id == id

group cats by exhibitions.Title into ans

select new

{

Title = ans.Key

}; ;

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

foreach (var res in response)

{

values.Add(res.Title);

}

return values;

}

public void AddCat(Cat cat)

{

(db.Root as RootClass).CatId.Put(cat);

(db.Root as RootClass).CatAgeSearch.Put(cat);

}

public void DeleteCat(string id)

{

var ans = IsHaveCat(id);

if(ans.Count > 0)

{

var message = "Кошка все еще учавствует в выскавках:\n";

foreach(var a in ans)

{

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

}

throw new Exception(message);

}

var found = (db.Root as RootClass).CatId[id] as Cat;

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

}

public void UpdateCat(Cat cat)

{

var found = (db.Root as RootClass).CatId[cat.Id] as Cat;

found.Age = cat.Age;

found.Name = cat.Name;

found.Modify();

}

public List<Cat> GetCats() => (db.Root as RootClass).CatId.Cast<Cat>().ToList();

public List<Cat> GetCats(int from, int to)

{

Key key1 = new Key(from);

Key key2 = new Key(to);

var responce = (db.Root as RootClass).CatAgeSearch.Range(

key1, key2,

IterationOrder.DescentOrder

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

return responce;

}

#endregion

#region Работа с собаками

public void AddDog(Dog dog)

{

(db.Root as RootClass).DogId.Put(dog);

}

public List<string> IsHaveDog(string id)

{

var found = ((db.Root as RootClass).ExhibitionId);

var response = from exhibitions in found.Cast<Exhibition>().ToList()

from dogs in exhibitions.Dogs.Cast<Dog>().ToList()

where dogs.Id == id

group dogs by exhibitions.Title into ans

select new

{

Title = ans.Key

}; ;

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

foreach (var res in response)

{

values.Add(res.Title);

}

return values;

}

public void DeleteDog(string id)

{

var ans = IsHaveDog(id);

if (ans.Count > 0)

{

var message = "Собака все еще учавствует в выскавках:\n";

foreach (var a in ans)

{

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

}

throw new Exception(message);

}

var found = (db.Root as RootClass).DogId[id] as Dog;

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

}

public void UpdateDog(Dog dog)

{

var found = (db.Root as RootClass).DogId[dog.Id] as Dog;

found.Age = dog.Age;

found.Name = dog.Name;

found.Modify();

}

public List<Dog> GetDogs() => (db.Root as RootClass).DogId.Cast<Dog>().ToList();

public List<Dog> GetDogs(DogBreed dogBreed, string name)

{

var dog = new Dog();

dog.Name = name;

dog.Breed = dogBreed;

var response = (db.Root as RootClass).DogId.Select(typeof(Dog),

$"Name like '%{dog.Name}%' and " +

$"PrintBreed like '{dog.PrintBreed()}'").Cast<Dog>().ToList();

return response;

}

#endregion

#region Работа с птицами

public void AddBird(Bird bird)

{

(db.Root as RootClass).BirdId.Put(bird);

}

public List<string> IsHaveBird(string id)

{

var found = ((db.Root as RootClass).ExhibitionId);

var response = from exhibitions in found.Cast<Exhibition>().ToList()

from birds in exhibitions.Birds.Cast<Bird>().ToList()

where birds.Id == id

group birds by exhibitions.Title into ans

select new

{

Title = ans.Key

}; ;

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

foreach (var res in response)

{

values.Add(res.Title);

}

return values;

}

public void DeleteBird(string id)

{

var ans = IsHaveBird(id);

if (ans.Count > 0)

{

var message = "Птица все еще учавствует в выскавках:\n";

foreach (var a in ans)

{

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

}

throw new Exception(message);

}

var found = (db.Root as RootClass).BirdId[id] as Bird;

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

}

public void UpdateBird(Bird bird)

{

var found = (db.Root as RootClass).BirdId[bird.Id] as Bird;

found.Age = bird.Age;

found.Name = bird.Name;

found.Modify();

}

public List<Bird> GetBirds() => (db.Root as RootClass).BirdId.Cast<Bird>().ToList();

#endregion

#region Работа с выставками

public Exhibition AddExhibition(List<Cat> cats, List<Dog> dogs, List<Bird> birds,

string title)

{

var exhibition = new Exhibition();

exhibition.Cats = db.CreateLink();

exhibition.Dogs = db.CreateLink();

exhibition.Birds = db.CreateLink();

foreach(var cat in cats)

{

exhibition.Cats.Add(cat);

exhibition.Modify();

}

foreach(var dog in dogs)

{

exhibition.Dogs.Add(dog);

exhibition.Modify();

}

foreach(var bird in birds)

{

exhibition.Birds.Add(bird);

exhibition.Modify();

}

exhibition.Title = title;

(db.Root as RootClass).ExhibitionId.Put(exhibition);

return exhibition;

}

public List<Exhibition> GetExhibitions() => (db.Root as RootClass).ExhibitionId.Cast<Exhibition>().ToList();

public Exhibition GetExhibitions(string id) => (db.Root as RootClass).ExhibitionId[id] as Exhibition;

public void DeleteExhibition(string id)

{

(db.Root as RootClass).ExhibitionId.Remove(GetExhibitions(id));

}

public Exhibition UpdateExhibition(string id, List<Cat> cats, List<Dog> dogs, List<Bird> birds,

string title)

{

var exhibition = (db.Root as RootClass).ExhibitionId[id] as Exhibition;

exhibition.Cats.Clear();

exhibition.Dogs.Clear();

exhibition.Birds.Clear();

foreach (var cat in cats)

{

exhibition.Cats.Add(cat);

exhibition.Modify();

}

foreach (var dog in dogs)

{

exhibition.Dogs.Add(dog);

exhibition.Modify();

}

foreach (var bird in birds)

{

exhibition.Birds.Add(bird);

exhibition.Modify();

}

exhibition.Title = title;

exhibition.Modify();

return exhibition;

}

#endregion

}

}

*Класс Dog:*

using Perst;

using System;

using System.ComponentModel;

namespace CatPerst.Classes

{

internal class Dog : Persistent, IAnimal

{

[DisplayName("Id")]

public string Id { get; set; }

[DisplayName("Кличка")]

public string Name { get; set; }

[DisplayName("Возраст")]

public int Age { get; set; }

[DisplayName("Цвет")]

public AnimalColor Color { get; set; }

[DisplayName("Порода")]

public DogBreed Breed { get; set; }

public Dog()

{

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

Name = "Name";

Age = 0;

Color = AnimalColor.White;

Breed = DogBreed.Mudi;

}

public Dog(string name, int age, AnimalColor color, DogBreed breed)

{

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

Name = name;

Age = age;

Color = color;

Breed = breed;

}

public override bool Equals(object obj)

{

return obj is Dog dog &&

Name == dog.Name &&

Age == dog.Age &&

Color == dog.Color &&

Breed == dog.Breed;

}

public override string ToString()

{

return $"Id: {Id}\nКличка: {Name}\nВозраст: {Age}\nПорода: {PrintBreed()}\n" +

$"Цвет: {PrintColor()}";

}

public string PrintBreed()

{

var breed = "";

switch ((int)Breed)

{

case 0:

{

breed = "Айди";

}

break;

case 1:

{

breed = "Бардет";

}

break;

case 2:

{

breed = "Чортай";

}

break;

case 3:

{

breed = "Динго";

}

break;

case 4:

{

breed = "Хокайдо";

}

break;

case 5:

{

breed = "Кокони";

}

break;

case 6:

{

breed = "Муди";

}

break;

case 7:

{

breed = "Пули";

}

break;

case 8:

{

breed = "Ротвейлер";

}

break;

case 9:

{

breed = "Тази";

}

break;

}

return breed;

}

public string PrintColor()

{

string color = "";

switch ((int)Color)

{

case 0:

{

color = "Серый";

}

break;

case 1:

{

color = "Черный";

}

break;

case 3:

{

color = "Белый";

}

break;

case 4:

{

color = "Рыжий";

}

break;

}

return color;

}

}

public enum DogBreed: int

{

Aidi = 0,

Bardet = 1,

Chortai = 2,

Dingo = 3,

Hokkaido = 4,

Kokoni = 5,

Mudi = 6,

Puli = 7,

Rottweiler = 8,

Tazy = 9

}

}

*Класс RootClass:*

using Perst;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace CatPerst.Classes

{

internal class RootClass: Persistent

{

public FieldIndex CatId;

public FieldIndex DogId;

public FieldIndex BirdId;

public FieldIndex ExhibitionId;

public FieldIndex CatAgeSearch;

public RootClass(Storage db):base(db)

{

CatId = db.CreateFieldIndex(typeof(Cat), "Id", true);

DogId = db.CreateFieldIndex(typeof(Dog), "Id", true);

BirdId = db.CreateFieldIndex(typeof(Bird), "Id", true);

ExhibitionId = db.CreateFieldIndex(typeof(Exhibition), "Id", true);

CatAgeSearch = db.CreateFieldIndex(typeof(Cat), "Age", false);

}

}

}

*Класс Exhibition:*

using Perst;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace CatPerst.Classes

{

internal class Exhibition: Persistent

{

[DisplayName("Id")]

public string Id { get; set; }

[DisplayName("Название")]

public string Title { get; set; }

[DisplayName("Коты")]

public Link Cats { get; set; }

[DisplayName("Собаки")]

public Link Dogs { get; set; }

[DisplayName("Птицы")]

public Link Birds { get; set; }

public Exhibition()

{

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

}

}

}

*Класс Cat:*

using System;

using System.Collections.Generic;

using System.Drawing.Drawing2D;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

using System.Xml.Linq;

using System.ComponentModel;

using Perst;

namespace CatPerst.Classes

{

internal class Cat : Persistent, IAnimal

{

[DisplayName("Id")]

public string Id { get; set; }

[DisplayName("Кличка")]

public string Name { get; set; }

[DisplayName("Возраст")]

public int Age { get; set; }

[DisplayName("Цвет")]

public AnimalColor Color { get; set; }

[DisplayName("Порода")]

public CatBreed Breed { get; set; }

public Cat()

{

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

Age = 0;

Color = AnimalColor.Black;

Breed = CatBreed.Minuet;

Name = "Name";

}

public Cat(int age, AnimalColor color, CatBreed breed, string name)

{

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

Age = age;

Color = color;

Breed = breed;

Name = name;

}

public string PrintColor()

{

string color = "";

switch ((int)Color){

case 0:

{

color = "Серый";

}

break;

case 1:

{

color = "Черный";

}

break;

case 3:

{

color = "Белый";

}

break;

case 4:

{

color = "Рыжий";

}

break;

}

return color;

}

public string PrintBreed()

{

var breed = "";

switch ((int)Breed)

{

case 0:

{

breed = "Азиатская";

}

break;

case 1:

{

breed = "Бенгальская";

}

break;

case 2:

{

breed = "Чаузи";

}

break;

case 3:

{

breed = "Кипреская";

}

break;

case 4:

{

breed = "Фолдекс";

}

break;

case 5:

{

breed = "Канаани";

}

break;

case 6:

{

breed = "Корат";

}

break;

case 7:

{

breed = "Менуэт";

}

break;

case 8:

{

breed = "Раас";

}

break;

case 9:

{

breed = "Сококе";

}

break;

}

return breed;

}

public override bool Equals(object obj)

{

return obj is Cat cat &&

Age == cat.Age &&

Color == cat.Color &&

Breed == cat.Breed &&

Name == cat.Name;

}

public override string ToString()

{

return $"Id: {Id}\nКличка: {Name}\nВозраст: {Age}\nПорода: {PrintBreed()}\n" +

$"Цвет: {PrintColor()}";

}

}

public enum AnimalColor: int

{

Grey = 0,

Black = 1,

White = 2,

Ginger = 3

}

public enum CatBreed: int

{

Asian = 0,

Bengal = 1,

Chausie = 2,

Cyprus = 3,

Foldex = 4,

Kanaani = 5,

Korat = 6,

Minuet = 7,

Raas = 8,

Sokoke = 9

}

}

*Класс Bird:*

using Perst;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace CatPerst.Classes

{

internal class Bird: Persistent, IAnimal

{

[DisplayName("Id")]

public string Id { get; set; }

[DisplayName("Кличка")]

public string Name { get; set; }

[DisplayName("Возраст")]

public int Age { get; set; }

[DisplayName("Цвет")]

public AnimalColor Color { get; set; }

[DisplayName("Порода")]

public BirdBreed Breed { get; set; }

public string PrintColor()

{

string color = "";

switch ((int)Color)

{

case 0:

{

color = "Серый";

}

break;

case 1:

{

color = "Черный";

}

break;

case 3:

{

color = "Белый";

}

break;

case 4:

{

color = "Рыжий";

}

break;

}

return color;

}

public string PrintBreed()

{

var breed = "";

switch ((int)Breed)

{

case 0:

{

breed = "Волнистый попугайчик";

}

break;

case 1:

{

breed = "Каик";

}

break;

case 2:

{

breed = "Уиллет";

}

break;

}

return breed;

}

public Bird()

{

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

Name = "Name";

Age = 0;

Color = AnimalColor.White;

Breed = BirdBreed.Willet;

}

public Bird(string name, int age, AnimalColor color, BirdBreed breed)

{

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

Name = name;

Age = age;

Color = color;

Breed = breed;

}

public override bool Equals(object obj)

{

return obj is Bird bird &&

Name == bird.Name &&

Age == bird.Age &&

Color == bird.Color &&

Breed == bird.Breed;

}

public override string ToString()

{

return $"Id: {Id}\nКличка: {Name}\nВозраст: {Age}\nПорода: {PrintBreed()}\n" +

$"Цвет: {PrintColor()}";

}

}

public enum BirdBreed: int

{

Budgerigars = 0,

Caique = 1,

Willet = 2

}

}

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

Листинг Form1:

using CatPerst.Classes;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Drawing.Drawing2D;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Xml.Linq;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace CatPerst

{

/\*

\* Идеи для поисков:

\* Индексы: определенные рамки возраста +

\* SQL: фильтр по породе и имя начинается на строку

\*/

public partial class Form1 : Form

{

DataBase dataBase;

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

{

dataBase.Close();

}

public Form1()

{

InitializeComponent();

dataBase = new DataBase();

dataBase.Connect();

SelectSource();

ComboInit();

TableInit();

}

private void SelectSource()

{

foreach(var cat in dataBase.GetCats())

{

catBindingSource.Add(cat);

}

foreach (var dog in dataBase.GetDogs())

{

dogBindingSource.Add(dog);

}

foreach(var bird in dataBase.GetBirds())

{

birdBindingSource.Add(bird);

}

foreach(var e in dataBase.GetExhibitions())

{

exhibitionBindingSource.Add(e);

}

}

private void ComboInit()

{

cmbCatBreed.DataSource = Enum.GetValues(typeof(CatBreed));

cmbCatColor.DataSource = Enum.GetValues(typeof(AnimalColor));

cmbDogBreed.DataSource = Enum.GetValues(typeof(DogBreed));

cmbDogColor.DataSource = Enum.GetValues(typeof(AnimalColor));

cmbBirdBreed.DataSource = Enum.GetValues(typeof(BirdBreed));

cmbBirdColor.DataSource = Enum.GetValues(typeof(AnimalColor));

cmbDogBreedSearch.DataSource = Enum.GetValues(typeof(DogBreed));

rtbExhibitionCat.Text = (cmbExhibitionCat.SelectedItem as Cat)?.ToString();

rtbExhibitionDog.Text = (cmbExhibitionDog.SelectedItem as Dog)?.ToString();

rtbExhibitionBird.Text = (cmbExhibitionBird.SelectedItem as Bird)?.ToString();

}

private void TableInit()

{

if (exhibitionBindingSource.Count > 0)

{

var exhibition = dataBase.GetExhibitions(dgvExhibition.SelectedCells[0].Value.ToString());

var cats = exhibition.Cats.ToArray();

var dogs = exhibition.Dogs.ToArray();

var birds = exhibition.Birds.ToArray();

exhibitionCatBindingSource.Clear();

exhibitionDogBindingSource.Clear();

exhibitionBirdBindingSource.Clear();

foreach (Cat cat in cats)

{

exhibitionCatBindingSource.Add(cat);

}

foreach (Dog dog in dogs)

{

exhibitionDogBindingSource.Add(dog);

}

foreach (Bird bird in birds)

{

exhibitionBirdBindingSource.Add(bird);

}

txtExhibitionTitle.Text = exhibition.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 котами

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

{

if (cbCat.Checked)

{

dgvCat.Columns[1].ReadOnly = false;

dgvCat.Columns[2].ReadOnly = false;

dgvCat.Columns[5].Visible = true;

}

else

{

dgvCat.Columns[1].ReadOnly = true;

dgvCat.Columns[2].ReadOnly = true;

dgvCat.Columns[5].Visible = false;

}

}

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

{

if (e.RowIndex < 0)

{

return;

}

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

{

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

{

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

try

{

dataBase.DeleteCat(id);

catBindingSource.RemoveAt(e.RowIndex);

}

catch(Exception ex)

{

messageBoxError(ex.Message);

}

}

}

}

Cat selectedCat;

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

{

if (e.RowIndex < 0)

{

return;

}

selectedCat = new Cat();

selectedCat.Id = dgvCat.Rows[e.RowIndex].Cells[0].Value.ToString();

selectedCat.Name = dgvCat.Rows[e.RowIndex].Cells[1].Value.ToString();

selectedCat.Age = (int)dgvCat.Rows[e.RowIndex].Cells[2].Value;

selectedCat.Color = (AnimalColor)dgvCat.Rows[e.RowIndex].Cells[3].Value;

selectedCat.Breed = (CatBreed)dgvCat.Rows[e.RowIndex].Cells[4].Value;

}

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

{

Cat cat = new Cat();

cat.Id = dgvCat.Rows[e.RowIndex].Cells[0].Value.ToString();

cat.Name = dgvCat.Rows[e.RowIndex].Cells[1].Value.ToString();

cat.Age = (int)dgvCat.Rows[e.RowIndex].Cells[2].Value;

cat.Color = (AnimalColor)dgvCat.Rows[e.RowIndex].Cells[3].Value;

cat.Breed = (CatBreed)dgvCat.Rows[e.RowIndex].Cells[4].Value;

if(cat.Age < nudCatAge.Minimum || cat.Age > nudCatAge.Maximum)

{

messageBoxError("Введите корретное значение возраста");

dgvCat.Rows[e.RowIndex].Cells[2].Value = selectedCat.Age;

return;

}

if (cat.Equals(selectedCat))

{

return;

}

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

if (dialogResult == DialogResult.No)

{

dgvCat[1, e.RowIndex].Value = cat.Name;

dgvCat[2, e.RowIndex].Value = cat.Age;

dgvCat[3, e.RowIndex].Value = cat.Color;

dgvCat[4, e.RowIndex].Value = cat.Breed;

return;

}

if (dialogResult == DialogResult.Yes)

{

dataBase.UpdateCat(dgvCat.CurrentRow.DataBoundItem as Cat);

}

}

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

{

if (string.IsNullOrEmpty(txtCatName.Text))

{

messageBoxError("Вы не ввели кличку");

return;

}

Cat cat = new Cat();

cat.Age = (int)nudCatAge.Value;

cat.Name = txtCatName.Text;

cat.Color = (AnimalColor)cmbCatColor.SelectedItem;

cat.Breed = (CatBreed)cmbCatBreed.SelectedItem;

dataBase.AddCat(cat);

txtCatName.Clear();

catBindingSource.Add(cat);

messageBoxSuccessAdd();

}

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

{

dgvCat.Rows[e.RowIndex].Cells[2].Value = selectedCat.Age;

messageBoxError("Введите числовое значение возраста");

}

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

{

nudCatSearchTo.Minimum = nudCatSearchFrom.Value + 1;

catBindingSource.Clear();

foreach (var cat in dataBase.GetCats((int)nudCatSearchFrom.Value, (int)nudCatSearchTo.Value))

{

catBindingSource.Add(cat);

}

}

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

{

nudCatSearchFrom.Maximum = nudCatSearchTo.Value - 1;

catBindingSource.Clear();

foreach (var cat in dataBase.GetCats((int)nudCatSearchFrom.Value, (int)nudCatSearchTo.Value))

{

catBindingSource.Add(cat);

}

}

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

{

catBindingSource.Clear();

foreach (var cat in dataBase.GetCats())

{

catBindingSource.Add(cat);

}

}

#endregion

#region Работа с собаками

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

{

if (cbDog.Checked)

{

dgvDog.Columns[1].ReadOnly = false;

dgvDog.Columns[2].ReadOnly = false;

dgvDog.Columns[5].Visible = true;

}

else

{

dgvDog.Columns[1].ReadOnly = true;

dgvDog.Columns[2].ReadOnly = true;

dgvDog.Columns[5].Visible = false;

}

}

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

{

dogBindingSource.Clear();

var name = txtDogSearchName.Text;

var breed = (DogBreed)cmbDogBreedSearch.SelectedItem;

foreach (var dog in dataBase.GetDogs(breed, name))

{

dogBindingSource.Add(dog);

}

}

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

{

dogBindingSource.Clear();

var name = txtDogSearchName.Text;

var breed = (DogBreed)cmbDogBreedSearch.SelectedItem;

foreach (var dog in dataBase.GetDogs(breed, name))

{

dogBindingSource.Add(dog);

}

}

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

{

dogBindingSource.Clear();

foreach (var dog in dataBase.GetDogs())

{

dogBindingSource.Add(dog);

}

}

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

{

if (e.RowIndex < 0)

{

return;

}

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

{

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

{

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

try

{

dataBase.DeleteDog(id);

dogBindingSource.RemoveAt(e.RowIndex);

}

catch(Exception ex)

{

messageBoxError(ex.Message);

}

}

}

}

Dog selectedDog;

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

{

if (e.RowIndex < 0)

{

return;

}

selectedDog = new Dog();

selectedDog.Id = dgvDog.Rows[e.RowIndex].Cells[0].Value.ToString();

selectedDog.Name = dgvDog.Rows[e.RowIndex].Cells[1].Value.ToString();

selectedDog.Age = (int)dgvDog.Rows[e.RowIndex].Cells[2].Value;

selectedDog.Color = (AnimalColor)dgvDog.Rows[e.RowIndex].Cells[3].Value;

selectedDog.Breed = (DogBreed)dgvDog.Rows[e.RowIndex].Cells[4].Value;

}

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

{

int age = (int)dgvDog.Rows[e.RowIndex].Cells[2].Value;

if(age < nudDogAge.Minimum || age > nudDogAge.Maximum)

{

messageBoxError("Введите корретное значение возраста");

dgvDog.Rows[e.RowIndex].Cells[2].Value = selectedDog.Age;

return;

}

Dog dog = new Dog();

dog.Id = dgvDog.Rows[e.RowIndex].Cells[0].Value.ToString();

dog.Name = dgvDog.Rows[e.RowIndex].Cells[1].Value.ToString();

dog.Age = age;

dog.Color = (AnimalColor)dgvDog.Rows[e.RowIndex].Cells[3].Value;

dog.Breed = (DogBreed)dgvDog.Rows[e.RowIndex].Cells[4].Value;

if (dog.Equals(selectedDog))

{

return;

}

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

if (dialogResult == DialogResult.No)

{

dgvDog[1, e.RowIndex].Value = dog.Name;

dgvDog[2, e.RowIndex].Value = dog.Age;

dgvDog[3, e.RowIndex].Value = dog.Color;

dgvDog[4, e.RowIndex].Value = dog.Breed;

return;

}

if (dialogResult == DialogResult.Yes)

{

dataBase.UpdateDog(dgvDog.CurrentRow.DataBoundItem as Dog);

}

}

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

{

dgvDog.Rows[e.RowIndex].Cells[2].Value = selectedDog.Age;

messageBoxError("Введите числовое значение возраста");

}

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

{

if (string.IsNullOrEmpty(txtDogName.Text))

{

messageBoxError("Вы не ввели кличку");

return;

}

Dog dog = new Dog();

dog.Age = (int)nudDogAge.Value;

dog.Name = txtDogName.Text;

dog.Color = (AnimalColor)cmbDogColor.SelectedItem;

dog.Breed = (DogBreed)cmbDogBreed.SelectedItem;

dataBase.AddDog(dog);

txtDogName.Clear();

dogBindingSource.Add(dog);

messageBoxSuccessAdd();

}

#endregion

#region Работа с птицами

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

{

if (cbBird.Checked)

{

dgvBird.Columns[1].ReadOnly = false;

dgvBird.Columns[2].ReadOnly = false;

dgvBird.Columns[5].Visible = true;

}

else

{

dgvBird.Columns[1].ReadOnly = true;

dgvBird.Columns[2].ReadOnly = true;

dgvBird.Columns[5].Visible = false;

}

}

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

{

if (e.RowIndex < 0)

{

return;

}

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

{

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

{

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

try

{

dataBase.DeleteBird(id);

birdBindingSource.RemoveAt(e.RowIndex);

}

catch(Exception ex)

{

messageBoxError(ex.Message);

}

}

}

}

Bird selectedBird;

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

{

if (e.RowIndex < 0)

{

return;

}

selectedBird = new Bird();

selectedBird.Id = dgvBird.Rows[e.RowIndex].Cells[0].Value.ToString();

selectedBird.Name = dgvBird.Rows[e.RowIndex].Cells[1].Value.ToString();

selectedBird.Age = (int)dgvBird.Rows[e.RowIndex].Cells[2].Value;

selectedBird.Color = (AnimalColor)dgvBird.Rows[e.RowIndex].Cells[3].Value;

selectedBird.Breed = (BirdBreed)dgvBird.Rows[e.RowIndex].Cells[4].Value;

}

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

{

int age = (int)dgvBird.Rows[e.RowIndex].Cells[2].Value;

if (age < nudBirdAge.Minimum || age > nudBirdAge.Maximum)

{

messageBoxError("Введите корретное значение возраста");

dgvBird.Rows[e.RowIndex].Cells[2].Value = selectedBird.Age;

return;

}

Bird bird = new Bird();

bird.Id = dgvBird.Rows[e.RowIndex].Cells[0].Value.ToString();

bird.Name = dgvBird.Rows[e.RowIndex].Cells[1].Value.ToString();

bird.Age = age;

bird.Color = (AnimalColor)dgvBird.Rows[e.RowIndex].Cells[3].Value;

bird.Breed = (BirdBreed)dgvBird.Rows[e.RowIndex].Cells[4].Value;

if (bird.Equals(selectedBird))

{

return;

}

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

if (dialogResult == DialogResult.No)

{

dgvBird[1, e.RowIndex].Value = bird.Name;

dgvBird[2, e.RowIndex].Value = bird.Age;

dgvBird[3, e.RowIndex].Value = bird.Color;

dgvBird[4, e.RowIndex].Value = bird.Breed;

return;

}

if (dialogResult == DialogResult.Yes)

{

dataBase.UpdateBird(dgvBird.CurrentRow.DataBoundItem as Bird);

}

}

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

{

dgvBird.Rows[e.RowIndex].Cells[2].Value = selectedBird.Age;

messageBoxError("Введите числовое значение возраста");

}

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

{

if (string.IsNullOrEmpty(txtBirdName.Text))

{

messageBoxError("Вы не ввели кличку");

return;

}

Bird bird = new Bird();

bird.Age = (int)nudBirdAge.Value;

bird.Name = txtBirdName.Text;

bird.Color = (AnimalColor)cmbBirdColor.SelectedItem;

bird.Breed = (BirdBreed)cmbBirdBreed.SelectedItem;

dataBase.AddBird(bird);

txtBirdName.Clear();

birdBindingSource.Add(bird);

messageBoxSuccessAdd();

}

#endregion

#region Работа с выставками

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

{

rtbExhibitionCat.Text = (cmbExhibitionCat.SelectedItem as Cat)?.ToString();

}

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

{

rtbExhibitionDog.Text = (cmbExhibitionDog.SelectedItem as Dog)?.ToString();

}

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

{

rtbExhibitionBird.Text = (cmbExhibitionBird.SelectedItem as Bird)?.ToString();

}

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

{

if (catBindingSource.Count < 1)

{

messageBoxError("Не добавлены коты");

return;

}

var cat = (cmbExhibitionCat.SelectedItem as Cat);

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

{

if ((Cat)exhibitionCatBindingSource[i] == cat)

{

messageBoxError($"Кот: {cat.ToString()}\nуже учавствует в этой выставке");

return;

}

}

exhibitionCatBindingSource.Add(cat);

}

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

{

if (dogBindingSource.Count < 1)

{

messageBoxError("Не добавлены собаки");

return;

}

var dog = (cmbExhibitionDog.SelectedItem as Dog);

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

{

if ((Dog)exhibitionDogBindingSource[i] == dog)

{

messageBoxError($"Собака: {dog.ToString()}\nуже учавствует в этой выставке");

return;

}

}

exhibitionDogBindingSource.Add(dog);

}

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

{

if (birdBindingSource.Count < 1)

{

messageBoxError("Не добавлены птицы");

return;

}

var bird = (cmbExhibitionBird.SelectedItem as Bird);

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

{

if ((Bird)exhibitionBirdBindingSource[i] == bird)

{

messageBoxError($"Птица: {bird.ToString()}\nуже учавствует в этой выставке");

return;

}

}

exhibitionBirdBindingSource.Add(bird);

}

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

{

if (e.RowIndex < 0)

{

return;

}

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

{

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

{

exhibitionCatBindingSource.RemoveAt(e.RowIndex);

}

}

}

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

{

if (e.RowIndex < 0)

{

return;

}

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

{

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

{

exhibitionDogBindingSource.RemoveAt(e.RowIndex);

}

}

}

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

{

if (e.RowIndex < 0)

{

return;

}

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

{

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

{

exhibitionBirdBindingSource.RemoveAt(e.RowIndex);

}

}

}

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

{

if (exhibitionBirdBindingSource.Count == 0

&& exhibitionCatBindingSource.Count == 0

&& exhibitionDogBindingSource.Count == 0)

{

messageBoxError("В высавке должно принять участие хотя бы 1 животное");

return;

}

if (string.IsNullOrEmpty(txtExhibitionTitle.Text))

{

messageBoxError("Вы не ввели название");

return;

}

if (cbExhibition.Checked)

{

if (dgvExhibition.CurrentCell == null)

{

messageBoxError("Вы не выбрали выставку");

return;

}

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

if (result == DialogResult.Yes)

{

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

List<Cat> cats = new List<Cat>();

foreach (Cat cat in exhibitionCatBindingSource)

{

cats.Add(cat);

}

List<Dog> dogs = new List<Dog>();

foreach (Dog dog in exhibitionDogBindingSource)

{

dogs.Add(dog);

}

List<Bird> birds = new List<Bird>();

foreach (Bird bird in exhibitionBirdBindingSource)

{

birds.Add(bird);

}

dataBase.UpdateExhibition(id, cats, dogs, birds, txtExhibitionTitle.Text);

dgvExhibition.CurrentRow.Cells[1].Value = txtExhibitionTitle.Text;

MessageBox.Show("Запись успешно измененена",

"Успех", MessageBoxButtons.OK,

MessageBoxIcon.Information);

txtExhibitionTitle.Clear();

}

}

else

{

List<Cat> cats = new List<Cat>();

foreach (Cat cat in exhibitionCatBindingSource)

{

cats.Add(cat);

}

List<Dog> dogs = new List<Dog>();

foreach (Dog dog in exhibitionDogBindingSource)

{

dogs.Add(dog);

}

List<Bird> birds = new List<Bird>();

foreach (Bird bird in exhibitionBirdBindingSource)

{

birds.Add(bird);

}

exhibitionCatBindingSource.Clear();

exhibitionDogBindingSource.Clear();

exhibitionBirdBindingSource.Clear();

var exhibition = dataBase.AddExhibition(cats, dogs, birds, txtExhibitionTitle.Text);

exhibitionBindingSource.Add(exhibition);

txtExhibitionTitle.Clear();

messageBoxSuccessAdd();

}

}

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

{

if (cbExhibition.Checked)

{

dgvExhibition.Columns[2].Visible = true;

btnExhibitionAdd.Text = "Изменить выставку";

}

else

{

dgvExhibition.Columns[2].Visible = false;

btnExhibitionAdd.Text = "Добавить выставку";

}

}

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

{

if (e.RowIndex < 0)

{

return;

}

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

{

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

{

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

exhibitionBindingSource.RemoveAt(e.RowIndex);

exhibitionCatBindingSource.Clear();

exhibitionDogBindingSource.Clear();

exhibitionBirdBindingSource.Clear();

txtExhibitionTitle.Clear();

dataBase.DeleteExhibition(id);

}

}

else

{

var exhibition = dataBase.GetExhibitions((string)dgvExhibition.Rows[e.RowIndex].Cells[0].Value);

var cats = exhibition.Cats.ToArray();

var dogs = exhibition.Dogs.ToArray();

var birds = exhibition.Birds.ToArray();

exhibitionCatBindingSource.Clear();

exhibitionDogBindingSource.Clear();

exhibitionBirdBindingSource.Clear();

foreach (Cat cat in cats)

{

exhibitionCatBindingSource.Add(cat);

}

foreach (Dog dog in dogs)

{

exhibitionDogBindingSource.Add(dog);

}

foreach (Bird bird in birds)

{

exhibitionBirdBindingSource.Add(bird);

}

txtExhibitionTitle.Text = exhibition.Title;

}

}

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

{

exhibitionCatBindingSource.Clear();

exhibitionDogBindingSource.Clear();

exhibitionBirdBindingSource.Clear();

txtExhibitionTitle.Clear();

dgvExhibition.ClearSelection();

dgvExhibition.CurrentCell = null;

}

#endregion

}

}

# **3 Работа программы**

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

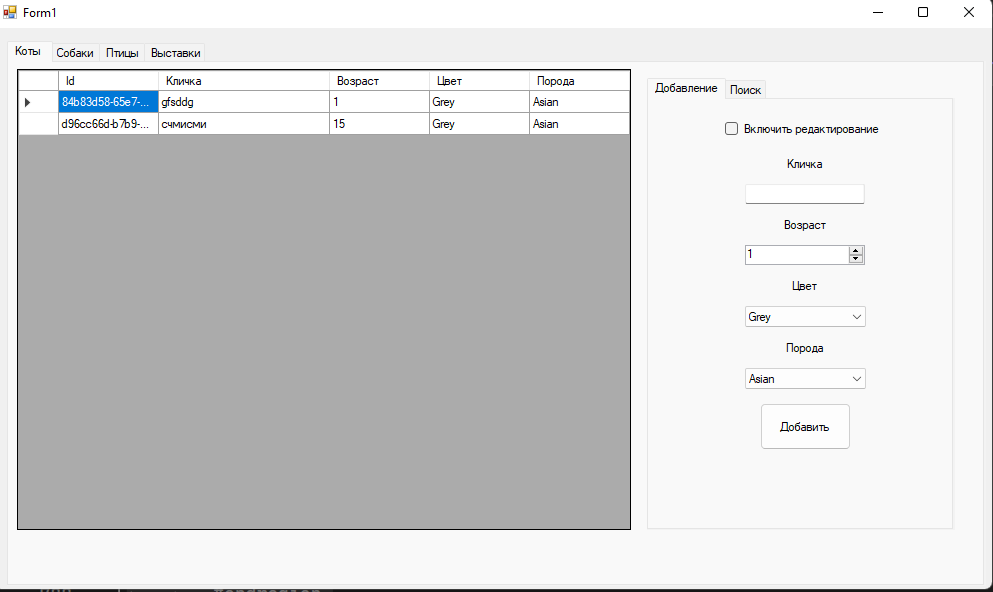


Рисунок 1 – Вкладка «Коты»

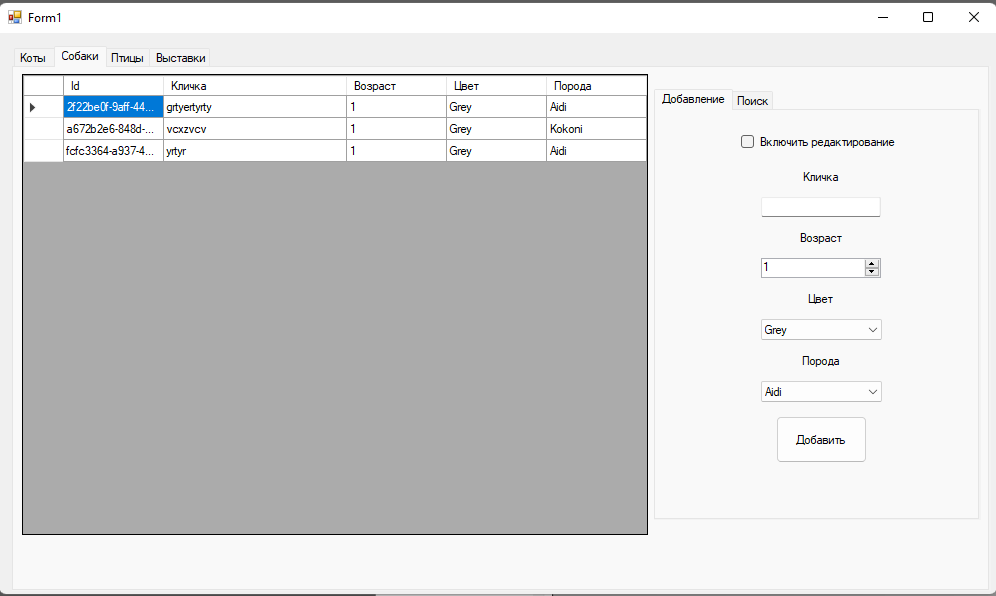


Рисунок 2 – Вкладка «Собаки»

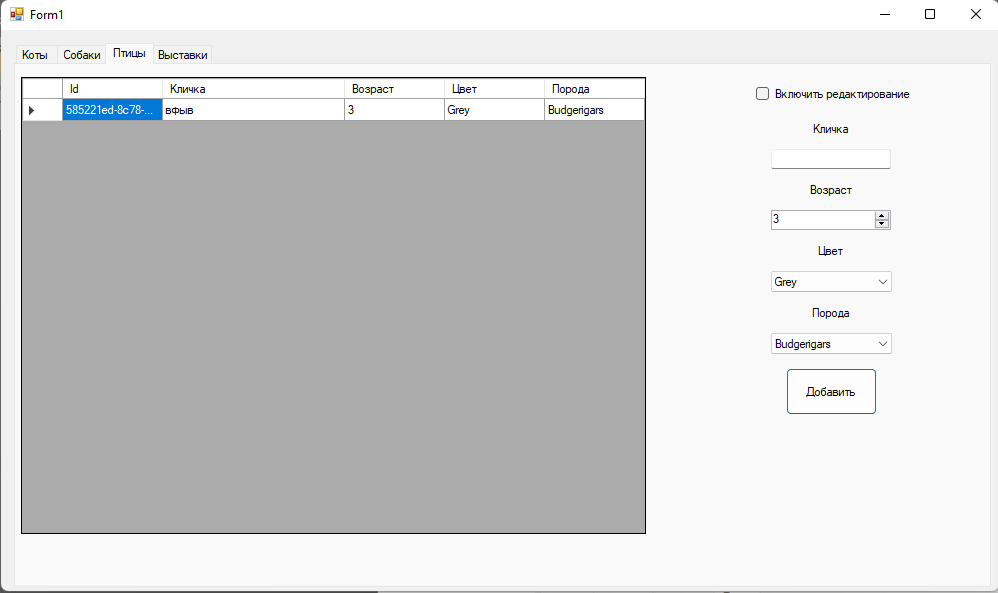


Рисунок 3 – Вкладка «Птицы»

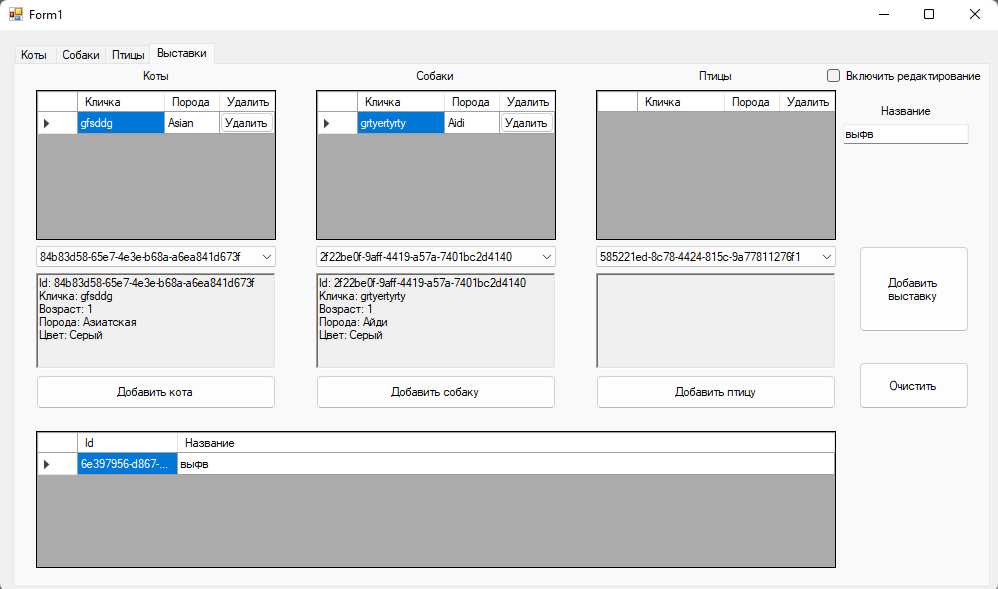


Рисунок 4 – Вкладка «Выставки»

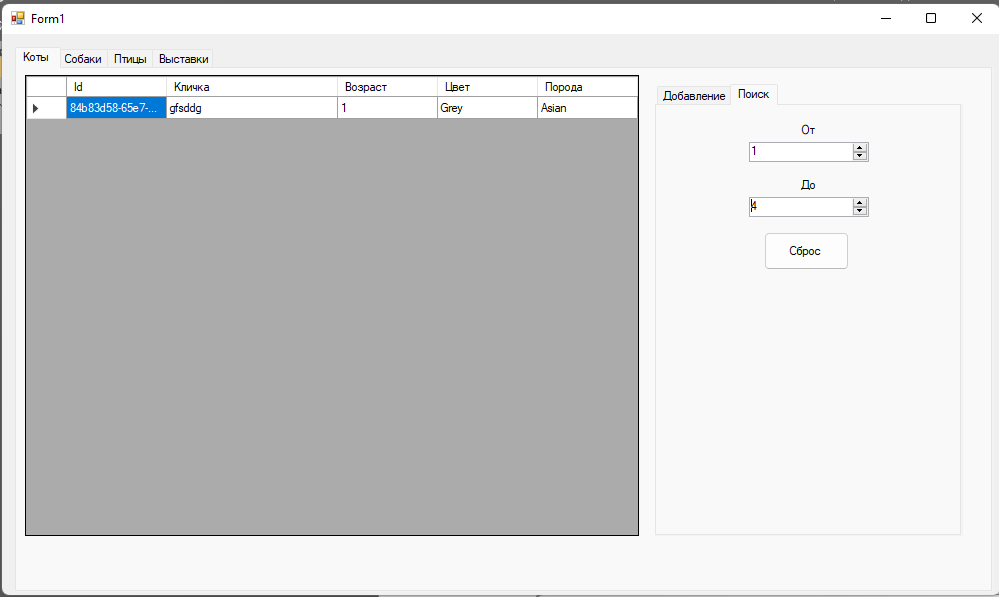


Рисунок 5 – Выдача первого документа

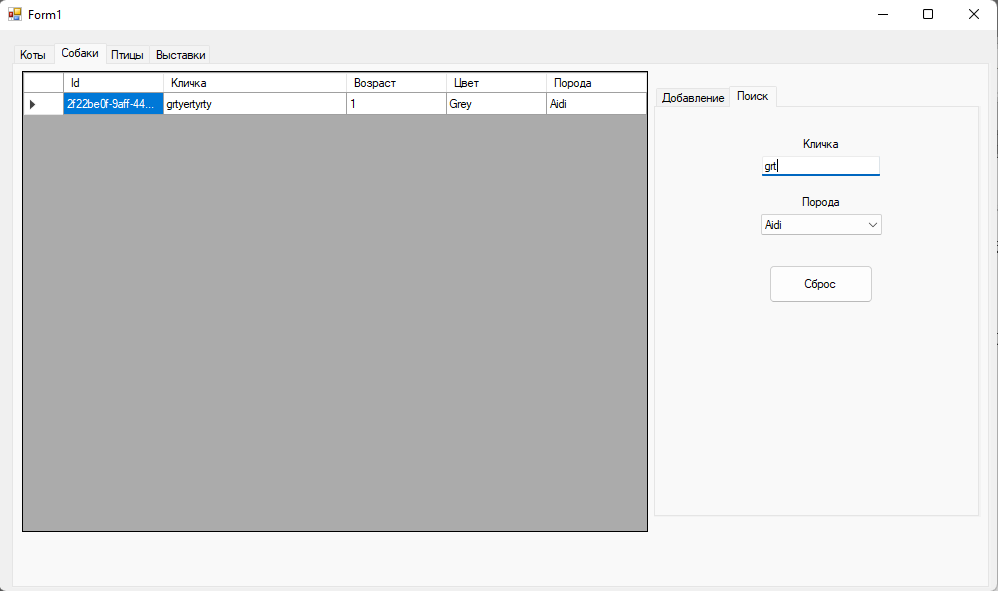


Рисунок 6 – Выдача второго документа