СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 4](#_Toc124858462)

[1. НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ 6](#_Toc124858463)

[2. ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ 7](#_Toc124858464)

[2.1. Постановка задачи 7](#_Toc124858465)

[2.2. Описание организации данных 7](#_Toc124858466)

[2.3. Описание программных средств 8](#_Toc124858467)

[2.4. Описание физической структуры базы данных 10](#_Toc124858468)

[2.5. Проектирование интерфейса 12](#_Toc124858469)

[2.6. Описание логической структуры программы 18](#_Toc124858470)

[3. ТЕСТИРОВАНИЕ И АНАЛИЗ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ 22](#_Toc124858471)

[4. РУКОВОДСТВО ПО ИСПОЛЬЗОВАНИЮ ПРОГРАММНОГО СРЕДСТВА25](#_Toc124858472)

[ЗАКЛЮЧЕНИЕ 34](#_Toc124858473)

[ИСТОЧНИКИ, ИСПОЛЬЗОВАННЫЕ ПРИ РАЗРАБОТКЕ 35](#_Toc124858474)

[ПЕРЕЧЕНЬ ПРИНЯТЫХ СОКРАЩЕНИЙ 36](#_Toc124858475)

[ПРИЛОЖЕНИЕ 1 (ОБЯЗАТЕЛЬНОЕ) – ЛИСТИНГ ПРОГРАММЫ 37](#_Toc124858476)

ВВЕДЕНИЕ

Сегодня любое предприятие использует профессиональное программное обеспечение для учета товаров, расходов, доходов и прочих позиций. Однако, универсальные решения, предлагаемые рынком, не всегда подходят конечному потребителю. Тогда он вынужден делать его на заказ.

Главной целью курсового проектирования является создание приложения «Фермерское хозяйство», которое сможет облегчить учет кормов, животных и их потребностей. А также расширение знаний по предмету «Базы данных и системы управления базами данных».

Задачи выполнения курсового проекта:

* углубление уровня профессионально значимых знаний, умений и навыков;
* овладение современными методами поиска, обработки и использования информации.

Пояснительная записка содержит следующие разделы:

1. «Назначение и область применения» описывается назначение программы, краткая характеристика области применения программы. Проводится анализ предметной области и инфологическое проектирование разрабатываемой базы данных. Приводится обзор существующих подходов к решению задачи, их достоинства и недостатки, отличие разработанной программы от аналогов. Рассматриваются существующие аналоги.
2. «Технические характеристики» – рассматриваются следующие вопросы: формирование требований к системе в целом; требования к функциям (раздел «Постановка задачи»); требования к организации данных в проекте, описание их организации, структуры, построение концептуальной и логической модели данных, способа кодирования (при необходимости) и предварительной подготовки (раздел «Описание организации данных»); обоснование выбора программного обеспечения, используемого для разработки программного средства (раздел «Описание программных средств»); проектирование программного средства. (разделы «Описание физической структуры базы данных», «Проектирование интерфейса», «Описание логической структуры программы»).
3. «Тестирование и анализ полученных результатов». Описывается проверка работы программы на различных тестах и приводится анализ ожидаемых и полученных результатов, выводы по результатам тестирования.
4. «Руководство по использованию программного средства» – описывается последовательность действий пользователя, обеспечивающих загрузку, запуск, выполнение и завершение программы, приведены описание функций, формата и возможных вариантов команд, с помощью которых пользователь управляет выполнением программы, а также ответы программы на эти команды.
5. НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ

В фермерском хозяйстве выращиваются на продажу животные. Производится закупка кормов для этих животных и их кормление. Также животным требуется регулярные медицинские обследования. Для всего этого нужен учет что и будет выполнятся в разрабатываемом приложении «Фермерское хозяйство».

Область применения: фермерское хозяйство.

Разрабатываемая программа имеет хорошую оптимизацию, достаточно малую ресурсоемкость и отсутствие прямых конкурентов с идентичным функционалом, что делает данное приложение на порядок выше многих своих аналогов.

Программное средство разработано для Windows 10.

Минимальные условия и требования к аппаратному обеспечению:

* процессор «Intel(R) Core(TM) i5-1021OU CPU @ 1.60GHz »;
* оперативная память 8гб;
* операционная система Windows 10 Pro;
* видеоадаптеры Intel(R) UHD Graphics;
* свободное место на диске не менее 1000 мб;
* 64 - разрядная ОС.

Фиксированных максимальных требований нет. Подойдут сборки с минимальными требованиями и выше.

1. ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ
   1. Постановка задачи

Спроектировать и разработать базу данных для хранения информации о фермерском хозяйстве. В базе данных необходимо хранить информацию о закупке корма, животных и их медицинских обследованиях, их кормлениях и их продаже. Разработать приложение «Фермерское хозяйство» для работы с базой данных.

В приложении реализовать возможность добавления информации о животных. При вводе данных, предусмотреть обработку исключительных ситуаций.

В приложении реализовать возможность добавления записи о покупке корма. При вводе данных, предусмотреть обработку исключительных ситуаций.

Предусмотреть возможность формирования записи о продаже (добавление информации в таблицу «продажа»).

Предусмотреть возможность формирования записи медицинского обследования (добавление информации в таблицу «медицинское обследование»).

Предусмотреть возможность формирования записи о кормлении (добавление информации в таблицу «кормление»).

Реализовать поиск и возможность просмотра информации о:

* введенном животном;
* введенном корме:
* релевантности животного;
* введенном медицинском обследовании по определенному животному;
  1. Описание организации данных

Входные данные разрабатываемого программного продукта:

* данные, вводимые пользователем, в том числе информация, вводимая в таблицы БД;
* данные, получаемые от пользователя;
* данные, хранящиеся в БД системе;
* значения, используемые по умолчанию: параметры поиска, структура таблиц.

Выходными данными разрабатываемого программного продукта являются:

* информация, сохраняемая в БД;
* отчеты.

В базе данных необходимо хранить следующую информацию:

* информация о животном, содержащая идентификационный номер животного, кличку, пол, дату рождения, породу и продано ли животное;
* информация о продаже, содержащая идентификационный номер продажи, идентификационный номер животного, дату продажи и цену;
* информация о медицинском обследовании, содержащая идентификационный номер медицинского обследования, идентификационный номер животного, дату медицинского обследования, подробная информация о медицинском обследовании и цена;
* информация о закупке корма, содержащая идентификационный номер закупки корма, название корма, количество корма, оставшиеся количество корма, единица измерения корма, цена за единицу товара и дату закупки;
* информация о кормлении, содержащая идентификационный номер кормления, идентификационный номер животного, идентификационный номер закупки корма, дату кормления и количество корма.
  1. Описание программных средств

В качестве выбранного языка программирования используется C#. C# (произносится си шарп) –­ объектно-ориентированный язык программирования. Разработан в 1998 – 2001 годах группой инженеров компании Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумота как язык разработки приложений для платформы Microsoft .NET Framework и .NET Core. Впоследствии был стандартизирован как ECMA-334 и ISO/IEC 23270. C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java.

Для реализации базы данных проекта используется Microsoft SQL Management Studio, язык запросов SQL и СУБД Microsoft SQL Server. Основной используемый язык запросов – Transact-SQL, создан совместно Microsoft и Sybase. Transact-SQL является реализацией стандарта ANSI/ISO по структурированному языку запросов (SQL) с расширениями.

Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, дизайнер классов и дизайнер схемы базы данных. Visual Studio позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода (как, например, Subversion и Visual SourceSafe), добавление новых наборов инструментов или инструментов для прочих аспектов процесса разработки программного обеспечения.

* 1. Описание физической структуры базы данных

Структура таблицы «животное» представлена в таблице 2.1.

Таблица 2.1 – Структура таблицы «животное»

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Описания |
| animalId | int | primary key, identity, идентификатор таблицы |
| nickname | nvarchar(15) | кличка животного |
| sex | bit | пол |
| bornDate | date | дата рождения |
| species | nvarchar(15) | порода |
| Exist | bit | наличие животного |

Структура таблицы «продажа» представлена в таблице 2.2.

Таблица 2.2 – Структура таблицы «продажа»

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Описания |
| saleId | int | primary key, identity, идентификатор таблицы |
| animalId | int | foreing key идентификатор проданного животного |
| saleDate | date | время продажи |
| price | float | цена продажи |

Таблица «продажа» связана с таблицей «животное» по вторичному ключу.

Структура таблицы «медицинское обследование» представлена в таблице 2.3.

Таблица 2.3 – Структура таблицы «медицинское обследование»

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Описания |
| medicalResearchId | int | primary key, identity, идентификатор таблицы |
| animalId | int | foreing key идентификатор животного |
| medicalResearchDate | date | время обследования |
| aboutResearch | nvarchar(15) | Подробности обследования |
| price | float | цена обследования |

Таблица «медицинское обследование» связана с таблицей «животное» по вторичному ключу.

Структура таблицы «закупка корма» представлена в таблице 2.4.

Таблица 2.4 – Структура таблицы «закупка корма»

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Описания |
| purchaseFoodId | int | primary key, identity, идентификатор таблицы |
| foodTitle | nvarchar(15) | название корма |
| foodCount | float | количество корма |
| remainder | float | оставшиеся количество корма |
| si | nvarchar(15) | система исчисления количества корма |
| priceForOne | float | цена за единицу корма |
| realizationDate | date | дата закупки |

Структура таблицы «кормление» представлена в таблице 2.5.

Таблица 2.5 – Структура таблицы «кормленние»

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Описания |
| feedConsumptionId | int | primary key, identity, идентификатор таблицы |
| animalId | int | foreing key идентификатор животного для кормления |
| purchaseFoodId | int | foreing key идентификатор корма |
| realizationDate | date | дата кормления |
| foodCount | float | количество корма |

Таблица «кормление» связана с таблицей «животное» по вторичному ключу.

Таблица «кормление» связана с таблицей «закупка корма» по вторичному ключу.

* 1. Проектирование интерфейса

Данный курсовой проект будет представлено несколькими формами формы.

Форма «menu» из которой происходит переход на иные окна, а также информация о релевантности каждого животного и общее количество животных в наличии и общее количество корма, рис 2.1:

* кнопка «Животные» – кнопка для перехода на форму с информацией о каждом животном и возможностью добавление нового животного;
* кнопка «Продажи» – кнопка для перехода на форму с информацией о каждой продаже;
* кнопка «Закупка корма» – кнопка для перехода на форму с информацией о каждой закупке корма и возможностью закупкой нового корма;
* кнопка «Кормление» – кнопка для перехода на форму с информацией о каждом кормлении и возможностью добавление нового кормления;
* компонент label для отображения количества животных в наличии;
* компонент label для отображения общего количества корма;
* компонент dataGridView для отображения результата запроса.

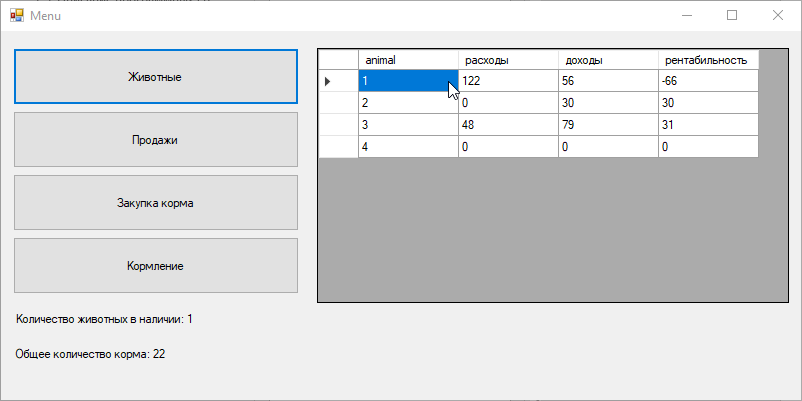


Рис 2.1. Форма «Menu»

Форма «Animal» в которой можно посмотреть информацию о животных. Перейти на форму продажи или медицинских обследований отдельного животного а также добавить в БД нового животного, рис 2.2:

* кнопка «добавить» – кнопка для добавления в таблицу «животное» запись с введенными данными;
* кнопка «медицинские обследования» – кнопка для перехода на форму с медицинскими обследованиями для выделенного животного;
* кнопка «продать» – кнопка для перехода на форму с возможностью продажи выделенного животного;
* кнопка «назад» – кнопка для перехода на предыдущую форму;
* компонент textBox для ввода клички животного;
* компонент radioButton для выбора пола животного;
* компонент dateTimePicker для выбора даты рождения животного;
* компонент richTextBox для ввода породы животного;
* компонент textBox для поиска животного по введенной в данное поле кличке;
* компонент dataGridView для отображения результата запроса.

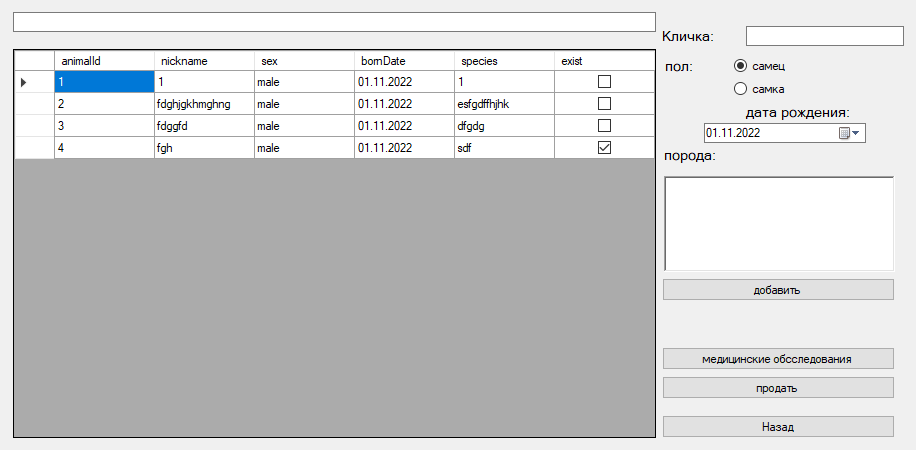


Рис 2.2. Форма «Animal»

Форма «MedicalReserch» в которой можно посмотреть информацию о медицинских обследованиях животного, а также добавить в БД новое медицинское обследование для животного, рис 2.3:

* кнопка «добавить» – кнопка для добавления в таблицу «медицинское обследование» запись с введенными данными;
* кнопка «назад» – кнопка для перехода на предыдущую форму;
* компонент numericUpDown для ввода цены медицинского обследования;
* компонент label для отображения клички выбранного животного;
* компонент dateTimePicker для выбора даты медицинского обследования;
* компонент richTextBox для ввода подробностей обследования;
* компонент textBox для поиска медицинского обследования по введенной в данное поле;
* компонент dataGridView для отображения результата запроса.

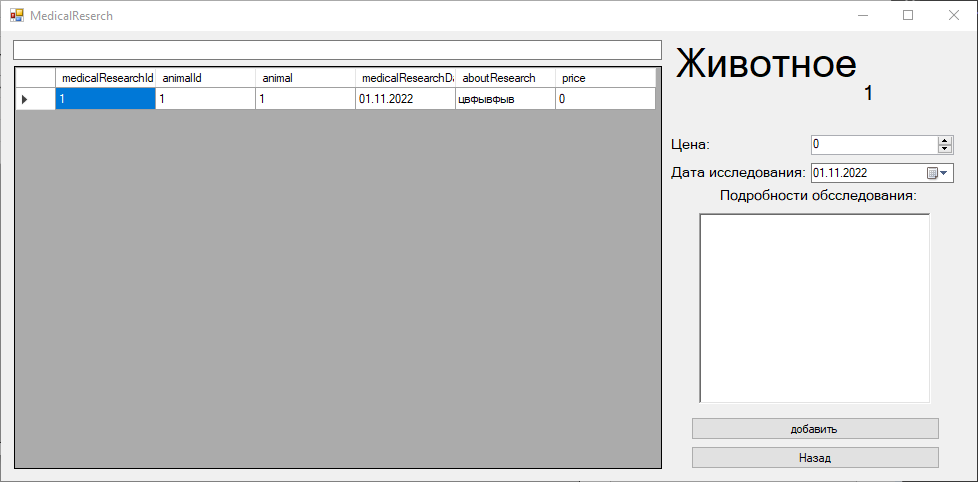


Рис 2.3. Форма «MedicalReserch»

Форма «SaleForm» в которой можно добавить в БД новую продажу животного, рис 2.4:

* компонент numericUpDown для ввода цены продажи;
* компонент dateTimePicker для выбора даты продажи;
* кнопка «Продать» – кнопка создания для добавления в таблицу продажи запись с введенными данными.

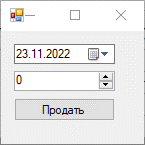


Рис 2.4. Форма «SaleForm»

Форма «AllSales» в которой можно посмотреть информацию о всех продажах, рис 2.5:

* кнопка «вывести» – кнопка вывода всех продаж с заданным временным диапазоном;
* кнопка «в отчет» – кнопка вывода всех продаж с заданным временным диапазоном в текстовый документ;
* кнопка «назад» – кнопка для перехода на предыдущую форму;
* компонент dateTimePicker для выбора начальной даты для диапазона;
* компонент dateTimePicker для выбора конечной даты для диапазона;
* компонент dataGridView для отображения результата запроса.

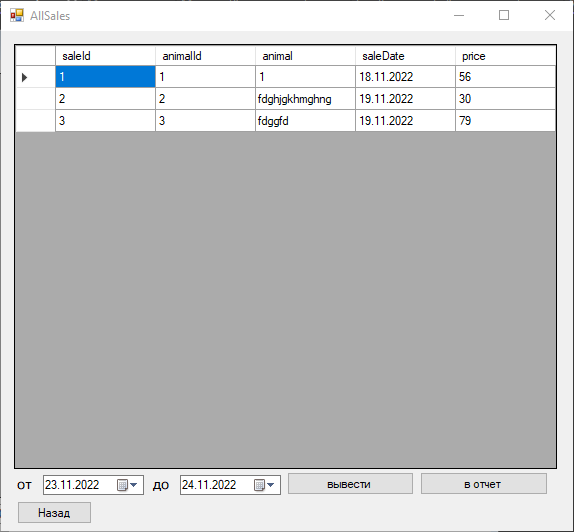


Рис 2.5. Форма «AllSales»

Форма «PurchaseFood» в которой можно посмотреть информацию о всех кормах, а также добавить в БД новый корм, рис 2.6:

* кнопка «вывести» – кнопка вывода всех продаж с заданным временным диапазоном;
* кнопка «в отчет» – кнопка вывода всех продаж с заданным временным диапазоном в текстовый документ;
* кнопка «назад» – кнопка для перехода на предыдущую форму;
* компонент dateTimePicker для выбора начальной даты для диапазона;
* компонент dateTimePicker для выбора конечной даты для диапазона;
* кнопка «добавить» – кнопка для добавления в таблицу «закупка корма» запись с введенными данными;
* компонент numericUpDown для ввода наибольшего оставшегося количества отображаемого корма;
* компонент textBox для поиска закупки корма по введенной в данное поле;
* компонент textBox для ввода названия корма;
* компонент numericUpDown для ввода количество корма;
* компонент textBox для ввода системы исчисления количества корма;
* компонент numericUpDown для ввода цены за единицу корма;
* компонент dateTimePicker для выбора даты закупке корма;
* компонент dataGridView для отображения результата запроса.

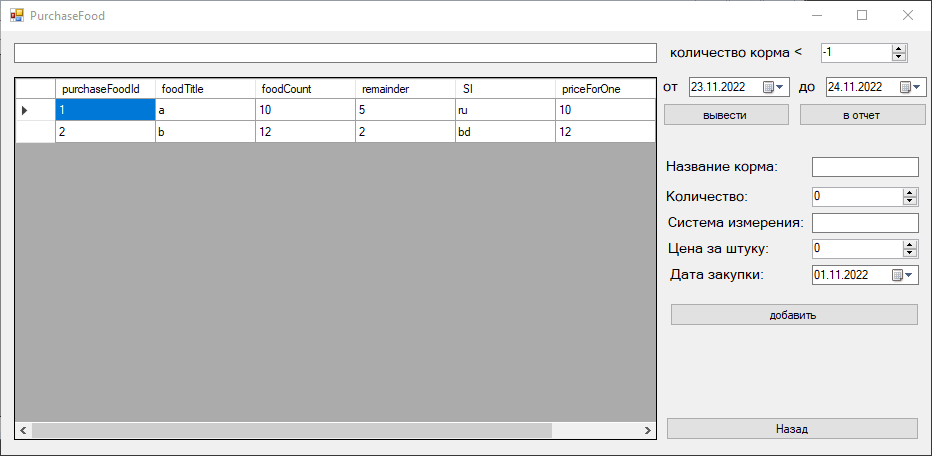


Рис 2.6. Форма «PurchaseFood»

Форма «FeedConsumption» в которой можно посмотреть информацию о всех кормлениях, а также добавить в БД новое кормление, рис 2.7:

* кнопка «назад» – кнопка для перехода на предыдущую форму;
* кнопка «добавить» – кнопка для добавления в таблицу «кормление» запись с введенными данными;
* компонент numericUpDown для ввода количества корма;
* компонент comboBox для выбора корма для кормления;
* компонент comboBox для выбора животного для кормления;
* компонент numericUpDown для ввода количество корма;
* компонент dateTimePicker для выбора даты закупке корма;
* компонент dataGridView для отображения результата запроса.

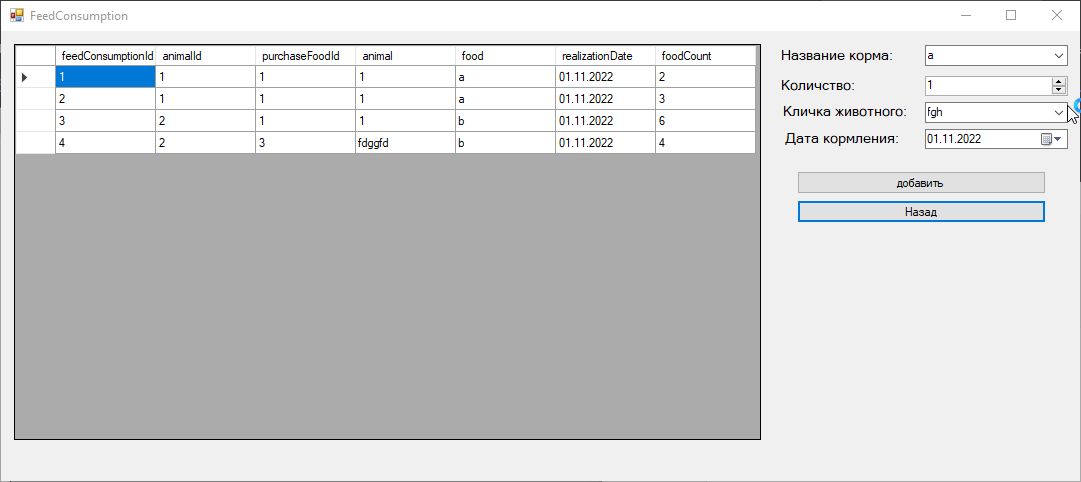


Рис 2.7. Форма «FeedConsumption»

* 1. Описание логической структуры программы

В классе AllSales.cs реализованы следующие функции:

* allSales\_Load(object sender, EventArgs e) ­– вызывается при загрузке формы;
* fillSalesTable() – заполняет таблицу значениями из таблицы «продажи»;
* button1\_Click(object sender, EventArgs e) – вызывается при нажатии кнопки «назад» закрывает текущую форму;
* allSales\_FormClosing(object sender, FormClosingEventArgs e) – вызывается при закрытии формы показывает предыдущую форму;
* dateTimePicker1\_ValueChanged(object sender, EventArgs e) – задает компоненту dateTimePicker2 минимальную дату равную значению dateTimePicker1 плюс 1 день;
* button3\_Click(object sender, EventArgs e) – вызывается при нажатии кнопки «вывести» заполняет таблицу значениями из таблицы «продажи» с заданным временным диапазоном;
* Button2\_Click(object sender, EventArgs e) – вызывается при нажатии кнопки «в отчет» заполняет текстовый документ значениями из таблицы «продажи» с заданным временным диапазоном.

В классе DBConnector.cs реализованы следующие функции:

* open() – открывает подключение к БД;
* close() – закрывает подключение к БД.

В классе FeedConsumption.cs реализованы следующие функции:

* button1\_Click(object sender, EventArgs e) – вызывается при нажатии кнопки «назад» закрывает текущую форму;
* FeedConsumption\_FormClosing(object sender, FormClosingEventArgs e) – вызывается при закрытии формы показывает предыдущую форму;
* FeedConsumption\_Load(object sender, EventArgs e) – вызывается при загрузке формы;
* fillAvailableFood() – сохраняет в свойствах все названия кормов что есть в наличии;
* fillAvailableAnimal() – сохраняет в свойствах все клички не проданных животных;
* fillFeedConsumptionTable() – заполняет таблицу значениями из таблицы «кормление»;
* canAddFeedConsumption() – проверяет валидность введенных в поля данных;
* getAnimalIdWithNickname(string nickname) – получает идентфикатор животного по его кличке;
* getPurchaseFoodIdWithTitle(string foodTitle) – получает идентификатор корма по его названию;
* button2\_Click(object sender, EventArgs e) – вызывается при нажатии кнопки «добавить» и при валидных данных добавляет в таблицу «кормление» новую запись;
* updateFormState() – обновляет состояние всех компонентов формы;
* comboBox1\_SelectedIndexChanged(object sender, EventArgs e) – при изменении выбора корма меняет максимальное возможное количество для выбора на максимальное количество выбранного корма.

В классе Form1.cs реализованы следующие функции:

* Form1\_Load(object sender, EventArgs e) – вызывается при загрузке формы;
* fillAnimalTable(string nickname = "") – заполняет таблицу данными из таблицы «животное» с заданной кличкой или без, при использовании значения по умолчанию;
* medicalButton\_Click(object sender, EventArgs e) – переход на форму « MedicalReserch» для выделенного в таблице животного;
* canAddAnimal() – проверяет валидность введенных в поля данных;
* button1\_Click(object sender, EventArgs e) – вызывается при нажатии кнопки «добавить» и при валидных данных добавляет в таблицу «животное» новую запись;
* button2\_Click(object sender, EventArgs e) – переход на форму «SaleForm» для выделенного в таблице животного;
* Form1\_Activated(object sender, EventArgs e) – вызывается при активации формы;
* button3\_Click(object sender, EventArgs e) – вызывается при нажатии кнопки «назад» закрывает текущую форму;
* Form1\_FormClosing(object sender, FormClosingEventArgs e) – вызывается при закрытии формы показывает предыдущую форму;
* textBox2\_TextChanged(object sender, EventArgs e) – заполняет таблицу значениями из таблицы «животное» с кличкой содержащей символы из компонента textBox2.

В классе MedicalReserch.cs реализованы следующие функции:

* fillDataGrid(string selectedAnimalId,string reserch="") – заполнение таблицы значениями таблицы «медицинское обследование» для определенного животного;
* getAnimalData(string selectedAnimalId) – получение информации о животном по его идентификатором;
* MedicalReserch\_Load(object sender, EventArgs e) – вызывается при загрузке формы;
* MedicalReserch\_FormClosing(object sender, FormClosingEventArgs e) – вызывается при закрытии формы показывает предыдущую форму;
* button2\_Click(object sender, EventArgs e) – вызывается при нажатии кнопки «добавить» и при валидных данных добавляет в таблицу «медицинское обследование» новую запись;
* button1\_Click(object sender, EventArgs e) – вызывается при нажатии кнопки «назад» закрывает текущую форму;
* textBox2\_TextChanged(object sender, EventArgs e) – заполняет таблицу значениями из таблицы «медицинское обследование» подробностями о медицинское обследовании содержащими символы из компонента textBox2.

В классе Menu.cs реализованы следующие функции:

* button1\_Click(object sender, EventArgs e) – при нажатии на кнопку «Животные» переход на форму «Animal»;
* button2\_Click(object sender, EventArgs e) – при нажатии на кнопку «Продажи» переход на форму «AllSale»;
* button3\_Click(object sender, EventArgs e) – при нажатии на кнопку «Закупка корма» переход на форму «PurchaseFood»;

1. ТЕСТИРОВАНИЕ И АНАЛИЗ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ

Существует несколько методов тестирования:

1. Метод «черного ящика». Основан на принципе "вход-выход". Программе подаются некоторые данные на вход и проверяются результаты, в надежде найти несоответствия. При этом как именно работает программа, считается несущественным.
2. Метод «белого ящика». В этом методе тестовые данные получают путем анализа логики программы.
3. Метод «серого ящика». это метод тестирования программного продукта или приложения с частичным знанием его внутреннего устройства.

Этапы процесса тестирования:

1. Проверка в нормальных условиях. Предполагает тестирование на основе данных, которые характерны для реальных условий функционирования системы.
2. Проверка в экстремальных условиях. Тестовые данные включают граничные значения области изменения входных переменных, граничные объемы данных.
3. Проверка в исключительных ситуациях. Проводится с использованием данных, значения которых лежат за пределами допустимой области. Программа должна отвергать любые данные, которые она не в состоянии обработать правильно. Наихудшая ситуация возникает тогда, когда программа принимает данные, выходящие за пределы, и выдает неверный результат.

В проекте использовался метод тестирования «черного ящика». Программа была протестирована на следующих тестах:

1. Тест №1. Добавление нового животного.
2. Тест №2. Продажа животного.
3. Тест №3. Добавление медицинского обследования.
4. Тест №4. Просмотр всех продаж.
5. Тест №5. Добавление новой закупки корма.
6. Тест №6. Добавление нового кормления.
7. Тест №7. Поиск животного по его кличке.
8. Тест №8. Поиск закупки корма по его названию.

Тест №1. Проверка добавления новой записи в таблицу «животное». Переход на форму «Animal». Ввод данных о кличке (тест1), пола (самец), дате рождения (01.11.2022) и породы (тест1). Нажатие кнопки «добавить». В таблице «животное» появится новая запись с введенными данными. Если введены некорректные данные пользователь увидит сообщение. Если данные введены корректно на форме помимо существующих записей в таблице «животное» отобразилась новая запись с данными.

Тест №2. Проверка продажи выбранного животного. Переход на форму «Animal». Нажатие на строку с животным. Нажатие кнопки «продать». Открытие формы «SaleForm». Нажатие кнопки «Продать». Ввод данных о дате (01.11.2022) и цене (1). Если введены некорректные данные пользователь увидит сообщение. Если данные введены корректно в таблице «продажи» появится новая запись с введенными данными.

Тест №3. Проверка добавления новой записи в таблицу «медицинское обследования». Переход на форму «Animal». Нажатие на строку с животным. Нажатие кнопки «медицинское обследование». Открытие формы «MedicalReserch». Ввод данных даты (01.11.2022), цены (1) и подробности обследования (подробности). Нажатие кнопки «добавить». Если введены некорректные данные пользователь увидит сообщение. Если данные введены корректно в таблице «медицинские обследования» появилась новая запись с введенными данными.

Тест №4. Просмотр всех продаж. Переход на форму «AllSales». На форме отобразятся все продажи в случае отсутствия продаж пользователь увидит сообщение.

Тест №5. Проверка добавления новой записи в таблицу «закупка корма». Переход на форму «PurchaseFood». Ввод данных о названии корма (корм), количестве (6), системе измерений (кг), цены за штуку (10) и даты закупки (01.11.2022). Нажатие кнопки «добавить». Если введены некорректные данные пользователь увидит сообщение. Если данные введены корректно в таблице «закупка корма» появилась новая запись с введенными данными.

Тест №6. Проверка добавления новой записи в таблицу «кормление». Переход на форму «FeedConsumption». Ввод данных о названии корма (корм), количестве (6), кличке животного (тест1) и даты кормления (01.11.2022). Нажатие кнопки «добавить». Если введены некорректные данные пользователь увидит сообщение. Если данные введены корректно в таблице «кормление» появилась новая запись с введенными данными.

Тест №7. Проверка поиска из таблицы «животное» на форме «Animal». Переходим на форму «Animal». Ввод данных о кличке (тест1) в верхние поле. Если животных с кличкой, начинающейся с введенных данных, не существует на форме не отобразятся животные, иначе на форме отобразятся записи из таблицы «животное» с кличкой, начинающейся с введенных данных.

Тест №8. Проверка поиска из таблицы «закупка корма» на форме «PurchaseFood». Переходим на форму «PurchaseFood». Ввод данных о названии корма (корм) в верхние поле. Если закупки корма с названием корма, начинающейся с введенных данных, не существует на форме не отобразятся данные о закупке корма, иначе на форме отобразятся записи из таблицы «закупка корма» с названием, начинающейся с введенных данных.

1. РУКОВОДСТВО ПО ИСПОЛЬЗОВАНИЮ ПРОГРАММНОГО СРЕДСТВА

При загрузке приложения загрузится форма «Menu», рис. 4.1. На ней будет отображена релевантность каждого животного, количество животных в наличии и общее количество корма.

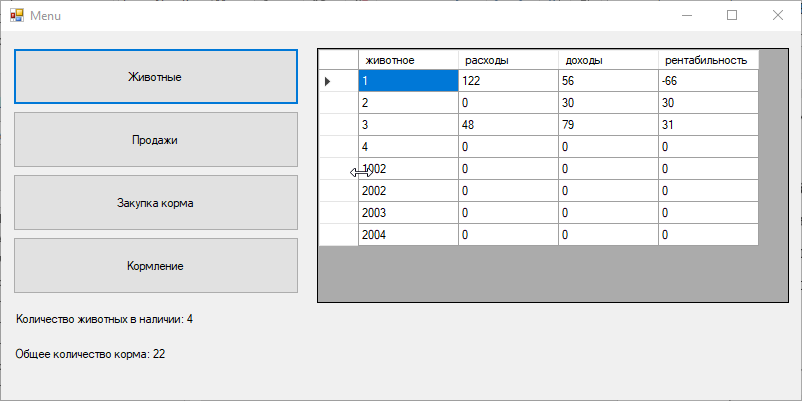


Рис 4.1. Форма «Menu»

Нажать кнопку «Животные», произойдет переход на форму «Animal» рис.4.2.

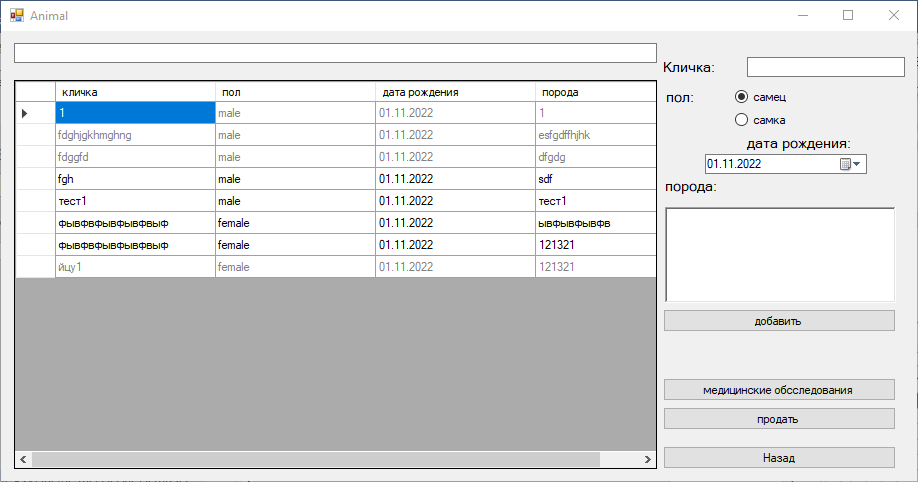


Рис 4.2. Форма «Animal»

В поля для добавления новой записи о животном, рис. 4.3, ввести данные нового животного. Нажать кнопку «добавить».

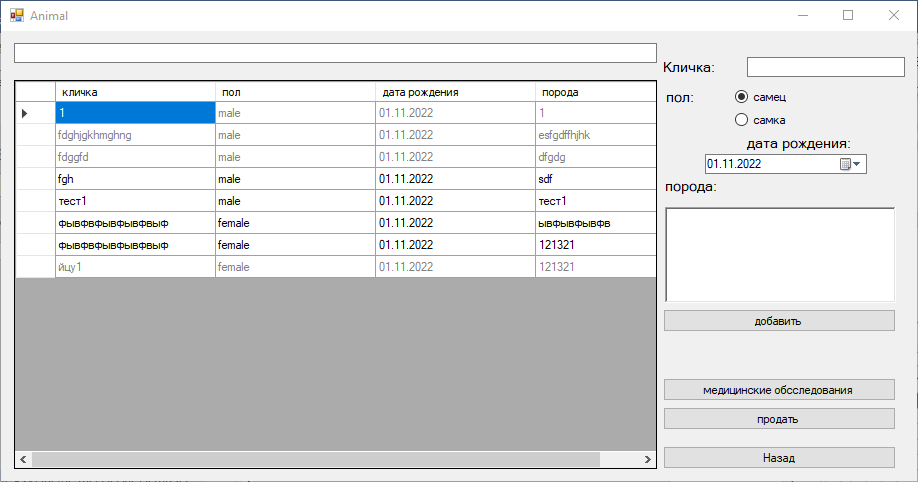


Рис 4.3. Поля для данных нового животного

На форме помимо существующих записей в таблице «животное» отобразится новая запись с введенными данными.

Добавление новой записи о закупке корма.

Нажать кнопку «Закупка корма», произойдет переход на форму «PurchaseFood» рис. 4.4.

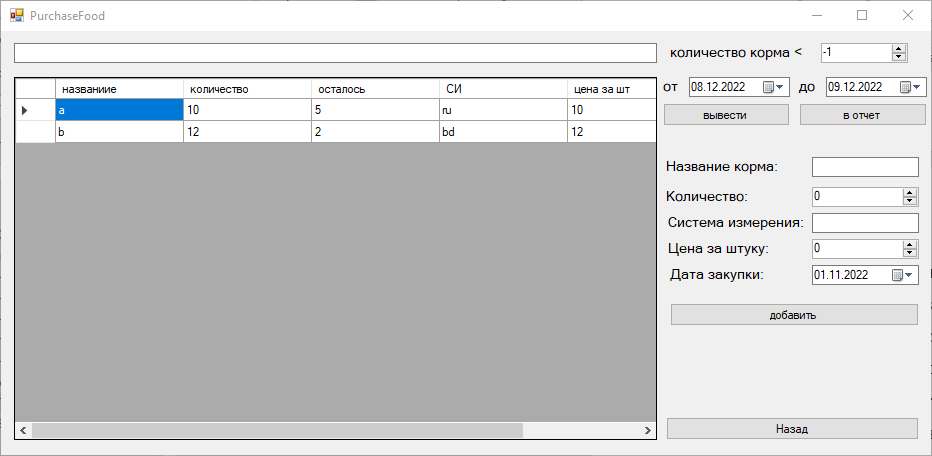


Рис 4.4. Форма «PurchaseFood»

В поля для добавления новой записи о закупке корма, рис. 4.5, ввести данные новой закупки корма. Нажать кнопку «добавить».

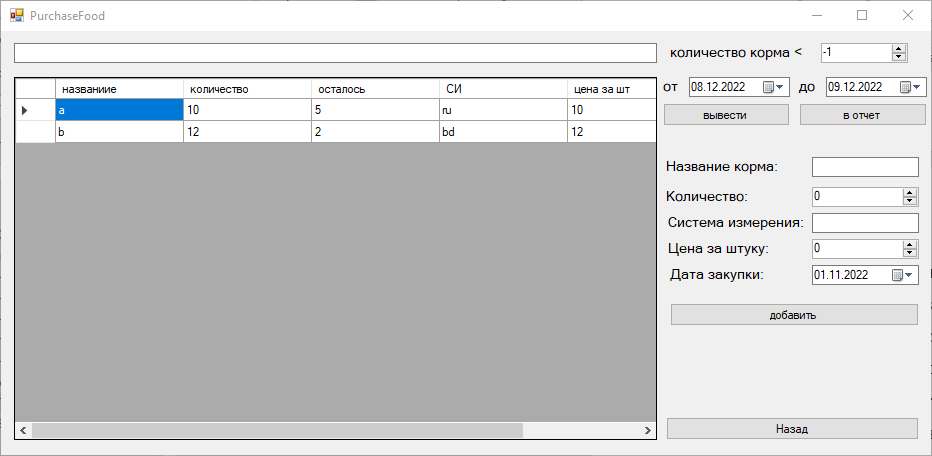


Рис 4.5. Поля для данных новой закупки

На форме помимо существующих записей в таблице «закупка корма» отобразится новая запись с введенными данными.

Формирование записи о продаже.

Выбрать животное для продажи нажав на соответствующую строку с животным в окне «Animal».

Нажать кнопку «продать». Откроется форма «SaleForm» рис. 4.6, где надо указать данные продажи и нажать кнопку «продать».

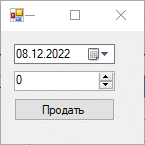


Рис 4.6. Форма «SaleForm»

В таблице «продажа» появится новая запись, которую можно посмотреть на форме «AllSales».

Формирование записи о медицинском обследовании.

Выбрать животное для которого хотите добавить медицинское обследование нажав на соответствующую строку с животным в окне «Animal».

Нажать кнопку «медицинские обследования». Откроется форма «MedicalReserch» рис. 4.7, где можно посмотреть общую информацию о медицинских обследованиях ранее выбранного животного, а также добавить запись о новом медицинском обследовании.

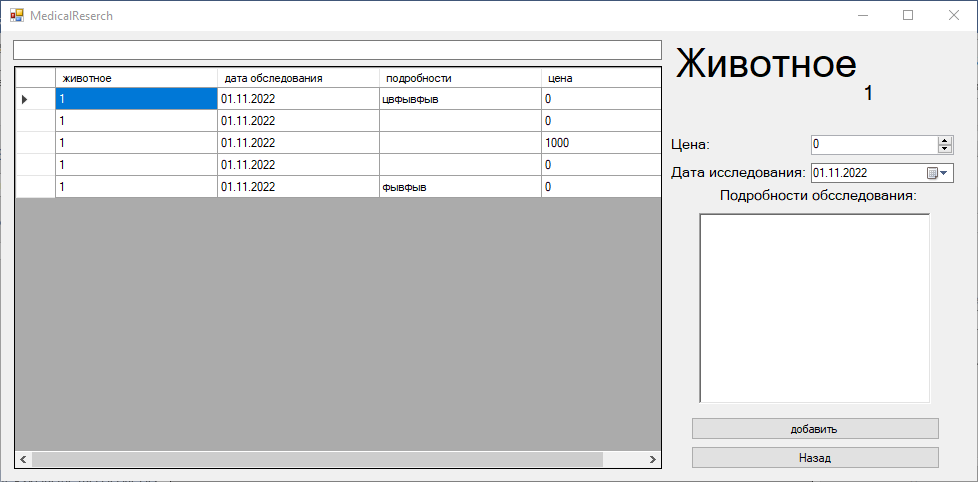


Рис 4.7. Форма «MedicalReserch»

В поля для добавления записи о новом медицинском обследовании, рис.4.8, ввести данные нового медицинского обследования. Нажать кнопку «добавить».

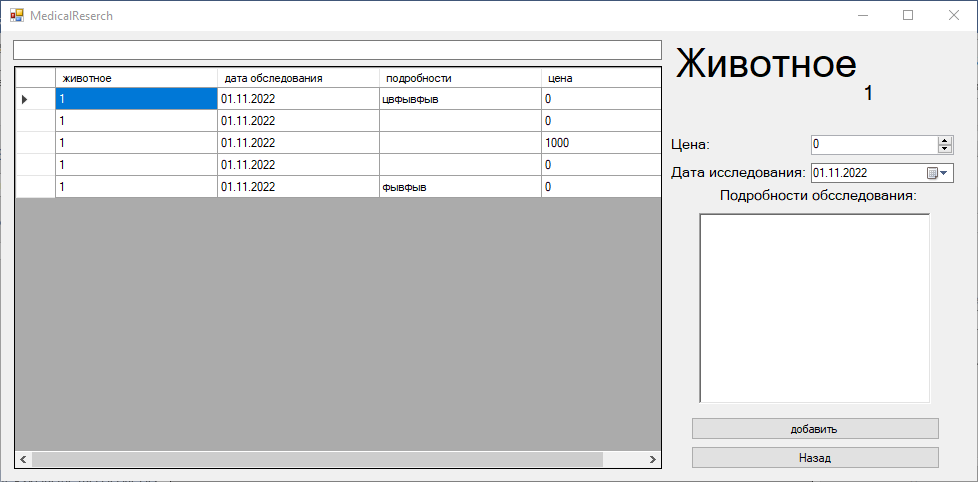


Рис 4.8. Поля для добавления нового медицинского обследования

На форме помимо существующих записей в таблице «медицинское обследование» отобразится новая запись с введенными данными.

Формирование записи о кормлении.

Нажать кнопку «Кормление», произойдет переход на форму «FeedConsumption» рис. 4.9.

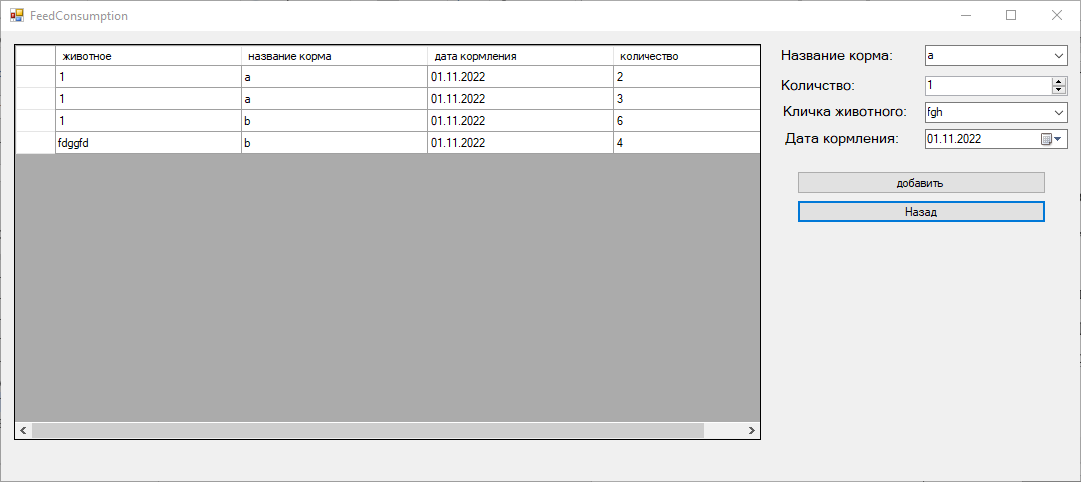


Рис 4.9. Форма «FeedConsumption»

В поля для добавления записи о новом кормлении, рис. 4.10, ввести данные нового кормления. Нажать кнопку «добавить».

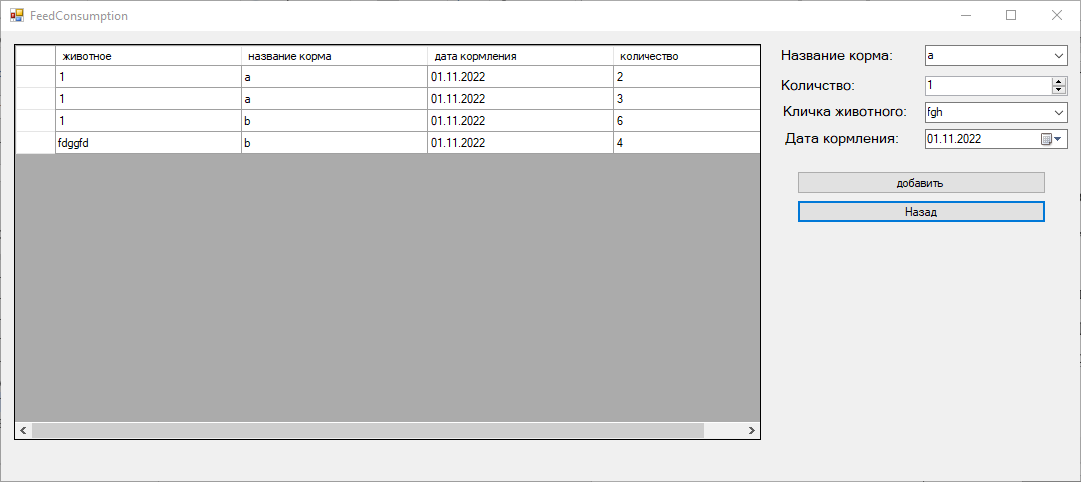


Рис 4.10. Поля для добавления нового кормления

На форме помимо существующих записей в таблице «кормление» отобразится новая запись с введенными данными.

Поиск и просмотр информации о введенном животном.

Выбрать поле в верху окна «Animal» рис.4.11, ввести кличку искомого животного

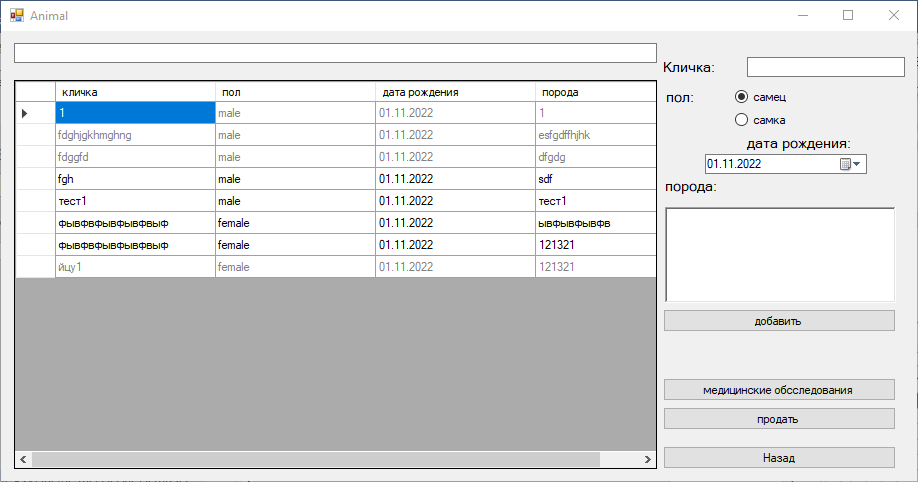


Рис 4.11. Поле в верху окна «Animal»

На форме отобразятся записи из таблицы «животное» у которых кличка начинается с введенных данных.

Поиск и просмотр информации о введенной закупки корма.

Выбрать поле в верху окна «PurchaseFood» рис.4.12, ввести название искомого корма

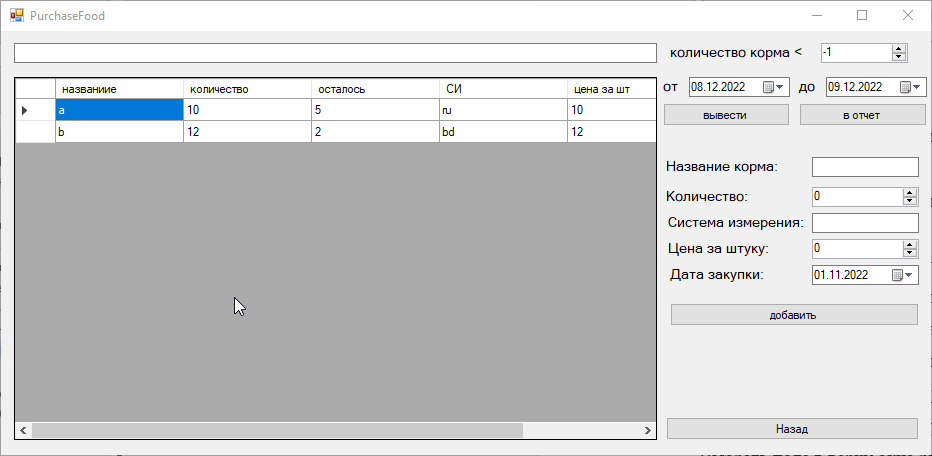


Рис 4.12. Поле в верху окна «PurchaseFood»

На форме отобразятся записи из таблицы «закупка корма» у которых название корма начинается с введенных данных.

Поиск и просмотр информации о введенном медицинском обследовании для животного.

Выбрать поле в верху окна «MedicalReserch» рис.4.13, ввести название искомого корма

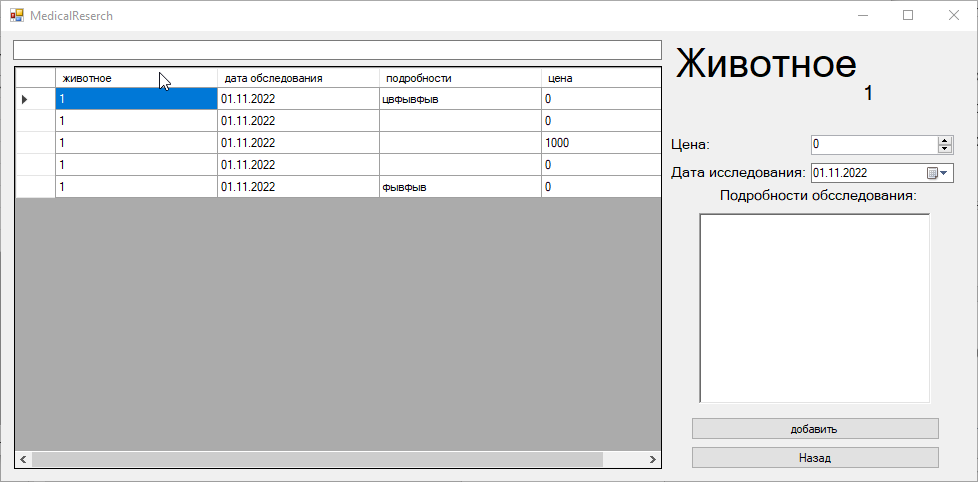


Рис 4.13. Поле в верху окна «MedicalReserch»

На форме отобразятся записи из таблицы «медицинские обследования» у которых подробности обследования начинаются с введенных данных.

Просмотр всех продажах за диапазон дат.

Выбрать начальную и конечную дату в полях для дат рис. 4.14 на форме «AllSales» и нажать кнопку «вывести».

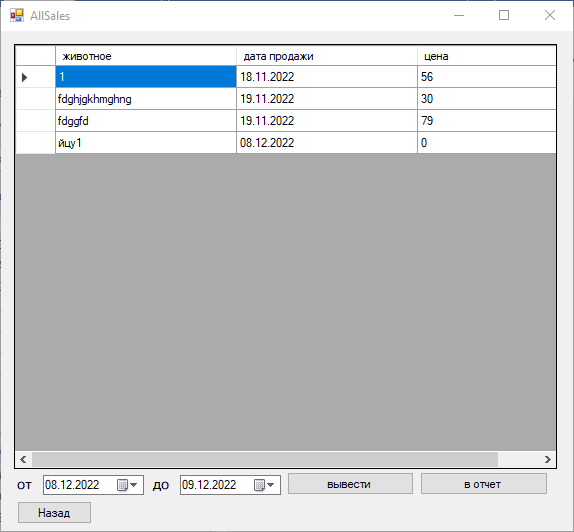


Рис 4.14. Поля диапазона дат на форме «AllSales»

На форме «AllSales» отобразятся записи из таблицы «продажа» у которых дата продажи входит в введенный диапазон.

Просмотр всех финансовых расходах на корм за диапазон дат.

Выбрать начальную и конечную дату в полях для дат рис. 4.15 на форме «PurchaseFood» и нажать кнопку «вывести».

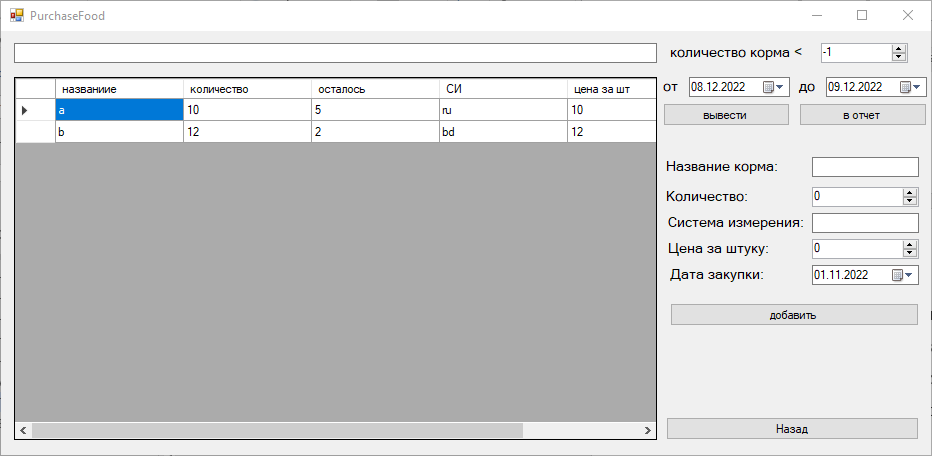


Рис 4.15. Поля диапазона дат на форме «PurchaseFood»

На форме «PurchaseFood» отобразятся записи из таблицы «закупка корма» у которых дата закупки входит в введенный диапазон.

Просмотр кормов меньше введенного.

В поле для количества корма на форме «PurchaseFood» ввести те корма чье количество корма меньше введенного.

На форме «PurchaseFood» отобразятся записи из таблицы «закупка корма» у которых оставшиеся количество корма меньше введенного.

Формирования отчета о продажах за диапазон дат.

Выбрать начальную и конечную дату в полях для дат на форме «AllSales» и нажать кнопку «в отчет».

В директории с проектом будет сформирован текстовый документ с отчетностью за введенный диапазон дат.

Просмотр всех финансовых расходах на корм за диапазон дат.

Выбрать начальную и конечную дату в полях для дат на форме «PurchaseFood» и нажать кнопку «в отчет».

В директории с проектом будет сформирован текстовый документ с отчетностью за введенный диапазон дат.

ЗАКЛЮЧЕНИЕ

В ходе курсового проекта было разработано приложение «Фермерское хозяйство». В программном продукте были реализованы все поставленные задачи, такие как формирование записей в таблицы, вывод данных в отчет и возможность просмотра и поиска информации о:

* введенном животном;
* введенном корме:
* релевантности животного;
* введенном медицинском обследовании по определенному животному;
* всех расходах за диапазон дат;
* всех продажах за диапазон дат;
* корме чье количество меньше заданного;
* количестве животных и корма на данный момент.

Для реализации программного средства был использован язык программирования С# и Microsoft SQL Server.

Приложение было протестировано. Тестирование прошло успешно.

ИСТОЧНИКИ, ИСПОЛЬЗОВАННЫЕ ПРИ РАЗРАБОТКЕ

1. Буч, Г. Введение в UML от создателей языка/ Г. Буч, Д. Рамбо, И.Якобсон. – М.: ДМК Пресс, 2015. – 496 с.
2. Шандриков, А.С. Стандартизация и сертификация программного обеспечения/ А.С. Шандриков. – Минск : РИПО, 2014. – 304 с.
3. Обучение кодированию в Visual Studio [Электронный ресурс] Режим доступа: https://visualstudio.microsoft.com/vs/getting-started/. – Дата доступа: 03.12.2022.
4. What is SQL Server Management Studio [Электронный ресурс] Режим доступа: <https://learn.microsoft.com/en-us/sql/ssms/sql-server-management-studio-ssms?view=sql-server-ver16>. – Дата доступа: 03.12.2022.
5. Введение в MS SQL Server и T-SQL [Электронный ресурс] Режим доступа: https://metanit.com/sql/sqlserver/1.1.php. – Дата доступа: 03.12.2022.

ПЕРЕЧЕНЬ ПРИНЯТЫХ СОКРАЩЕНИЙ

БД – база данных.

СУБД – система управления базами данных.

ПРИЛОЖЕНИЕ 1 (ОБЯЗАТЕЛЬНОЕ) – ЛИСТИНГ ПРОГРАММЫ

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace courseWork

{

public partial class AllSales : Form

{

public AllSales()

{

InitializeComponent();

}

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

{

fillSalesTable();

dateTimePicker2.MinDate = dateTimePicker1.Value.AddDays(1);

if (dataGridView1.RowCount == 0)

{

MessageBox.Show("Продажи отсутствуют");

}

}

private void fillSalesTable()

{

DBConnector.Open();

string query = $"select \* from salesView";

SqlDataAdapter sqlDataAdapter = new SqlDataAdapter(query, DBConnector.connection);

DataSet dataSet = new DataSet();

sqlDataAdapter.Fill(dataSet, "y");

dataGridView1.DataSource = dataSet.Tables["y"];

DBConnector.Close();

var columnWidth = dataGridView1.Width / 3;

dataGridView1.Columns[0].Visible = false;

dataGridView1.Columns[1].Visible = false;

string[] gridHeaders = new string[] { "животное", "дата продажи", "цена" };

for (int i = 2; i < 5; i++)

{

dataGridView1.Columns[i].HeaderText = gridHeaders[i - 2];

dataGridView1.Columns[i].Width = columnWidth;

}

}

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

{

Owner.Enabled = true;

Owner.Visible = true;

this.Close();

}

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

{

Owner.Enabled = true;

Owner.Visible = true;

}

private void dateTimePicker1\_ValueChanged(object sender, EventArgs e)

{

dateTimePicker2.MinDate = dateTimePicker1.Value.AddDays(1);

}

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

{

DBConnector.Open();

string query = $"select \* from salesView" +

$" where saleDate >= '{String.Join(".",dateTimePicker1.Value.ToShortDateString().Split('.').Reverse()).Replace(".","-")}' and " +

$"saleDate < '{String.Join(".", dateTimePicker2.Value.ToShortDateString().Split('.').Reverse()).Replace(".", "-")}'";

SqlDataAdapter sqlDataAdapter = new SqlDataAdapter(query, DBConnector.connection);

DataSet dataSet = new DataSet();

sqlDataAdapter.Fill(dataSet, "y");

dataGridView1.DataSource = dataSet.Tables["y"];

DBConnector.Close();

}

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

{

}

}

}

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace courseWork

{

public static class DBConnector

{

public static string connectionStr = Properties.Settings.Default.courseWorkDBConnectionString;

public static SqlConnection connection = new SqlConnection(connectionStr);

public static void Open()

{

connection.Open();

}

public static void Close()

{

connection.Close();

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace courseWork

{

public partial class FeedConsumption : Form

{

public FeedConsumption()

{

InitializeComponent();

}

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

{

Owner.Enabled = true;

Owner.Visible = true;

this.Close();

}

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

{

Owner.Enabled = true;

Owner.Visible = true;

}

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

{

fillFeedConsumptionTable();

fillAvailableFood();

fillAvailableAnimal();

Array.Sort(availableFoodTitle);

Array.Sort(availableAnimalNickname);

updateFormState();

}

string[] availableFoodTitle;

string[] availableAnimalNickname;

private void fillAvailableFood()

{

DBConnector.Open();

string query = $"select foodTitle from purchaseFood where remainder>0";

SqlDataAdapter sqlDataAdapter = new SqlDataAdapter(query, DBConnector.connection);

DataSet dataSet = new DataSet();

sqlDataAdapter.Fill(dataSet, "y");

DBConnector.Close();

availableFoodTitle = new string[dataSet.Tables["y"].Rows.Count];

for (int i = 0; i < availableFoodTitle.Length; i++)

{

var a = dataSet.Tables["y"].Rows[i][0].ToString();

availableFoodTitle[i] = a;

}

comboBox1.DataSource = availableFoodTitle;

}

private void fillAvailableAnimal()

{

DBConnector.Open();

string query = $"select nickname from animal where exist=1";

SqlDataAdapter sqlDataAdapter = new SqlDataAdapter(query, DBConnector.connection);

DataSet dataSet = new DataSet();

sqlDataAdapter.Fill(dataSet, "y");

DBConnector.Close();

availableAnimalNickname = new string[dataSet.Tables["y"].Rows.Count];

for (int i = 0; i < availableAnimalNickname.Length; i++)

{

var a = dataSet.Tables["y"].Rows[i][0].ToString();

availableAnimalNickname[i] = a;

}

comboBox2.DataSource = availableAnimalNickname;

}

private void fillFeedConsumptionTable()

{

DBConnector.Open();

string query = $"select \* from feedConsumptionView";

SqlDataAdapter sqlDataAdapter = new SqlDataAdapter(query, DBConnector.connection);

DataSet dataSet = new DataSet();

sqlDataAdapter.Fill(dataSet, "y");

dataGridView1.DataSource = dataSet.Tables["y"];

DBConnector.Close();

var columnWidth = dataGridView1.Width / 4;

dataGridView1.Columns[0].Visible = false;

dataGridView1.Columns[1].Visible = false;

dataGridView1.Columns[2].Visible = false;

string[] gridHeaders = new string[] { "животное", "название корма", "дата кормления", "количество" };

for (int i = 3; i < 7; i++)

{

dataGridView1.Columns[i].HeaderText = gridHeaders[i - 3];

dataGridView1.Columns[i].Width = columnWidth;

}

}

private bool canAddFeedConsumption()

{

if (comboBox2.Text == "")

{

MessageBox.Show("animal not exist");

return false;

}

if (comboBox1.Text == "")

{

MessageBox.Show("food not exist");

return false;

}

return true;

}

private string getAnimalIdWithNickname(string nickname)

{

DBConnector.Open();

string query = $"select animalId from animal where nickname=N'{nickname}' and exist=1";

SqlDataAdapter sqlDataAdapter = new SqlDataAdapter(query, DBConnector.connection);

DataSet dataSet = new DataSet();

sqlDataAdapter.Fill(dataSet, "y");

DBConnector.Close();

for (int i = 0; i < dataSet.Tables["y"].Rows.Count; i++)

{

return dataSet.Tables["y"].Rows[i][0].ToString();

}

return "";

}

private string getPurchaseFoodIdWithTitle(string foodTitle)

{

DBConnector.Open();

string query = $"select purchaseFoodId from purchaseFood where foodTitle=N'{foodTitle}' and remainder>0";

SqlDataAdapter sqlDataAdapter = new SqlDataAdapter(query, DBConnector.connection);

DataSet dataSet = new DataSet();

sqlDataAdapter.Fill(dataSet, "y");

DBConnector.Close();

for (int i = 0; i < dataSet.Tables["y"].Rows.Count; i++)

{

return dataSet.Tables["y"].Rows[i][0].ToString();

}

return "";

}

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

{

if (!canAddFeedConsumption()) return;

var command = new SqlCommand($"exec addFeedConsumption @animalId,@purchaseFoodId,@rd,@foodCount", DBConnector.connection);

var animalId=getAnimalIdWithNickname(comboBox2.Text);

var foodId= getPurchaseFoodIdWithTitle(comboBox1.Text);

command.Parameters.Add("@animalId", SqlDbType.Int).Value = int.Parse(animalId);

command.Parameters.Add("@purchaseFoodId", SqlDbType.Int).Value = int.Parse(foodId);

command.Parameters.Add("@rd", SqlDbType.Date).Value = dateTimePicker1.Value.Date;

command.Parameters.Add("@foodCount", SqlDbType.Int).Value = numericUpDown1.Value;

DBConnector.Open();

command.ExecuteNonQuery();

DBConnector.Close();

command = new SqlCommand($"update purchaseFood set remainder=remainder-@countFood where purchaseFoodId=@purchaseFoodId", DBConnector.connection);

command.Parameters.Add("@countFood", SqlDbType.Int).Value = numericUpDown1.Value;

command.Parameters.Add("@purchaseFoodId", SqlDbType.Int).Value = int.Parse(foodId);

DBConnector.Open();

command.ExecuteNonQuery();

DBConnector.Close();

updateFormState();

fillFeedConsumptionTable();

}

private void updateFormState()

{

fillAvailableFood();

fillAvailableAnimal();

Array.Sort(availableFoodTitle);

Array.Sort(availableAnimalNickname);

comboBox1\_SelectedIndexChanged(comboBox1, EventArgs.Empty);

if (availableFoodTitle.Length == 0)

{

MessageBox.Show("No food available");

this.Close();

}

if (availableFoodTitle.Length == 0)

{

MessageBox.Show("No animal available");

this.Close();

}

}

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

DBConnector.Open();

string query = $"select remainder from purchaseFood where foodTitle=N'{comboBox1.Text}'";

SqlDataAdapter sqlDataAdapter = new SqlDataAdapter(query, DBConnector.connection);

DataSet dataSet = new DataSet();

sqlDataAdapter.Fill(dataSet, "y");

DBConnector.Close();

int max = 0;

for (int i = 0; i < dataSet.Tables["y"].Rows.Count; i++)

{

var a = dataSet.Tables["y"].Rows[i][0].ToString();

max = int.Parse(a);

}

numericUpDown1.Maximum = max;

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace courseWork

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

fillAnimalTable();

dateTimePicker1.MaxDate=DateTime.Now;

}

private void fillAnimalTable(string nickname = "")

{

DBConnector.Open();

string query = $"select \* from animalView";

if (nickname != "") { query += $" where nickname like N'%{nickname}%'"; }

SqlDataAdapter sqlDataAdapter = new SqlDataAdapter(query, DBConnector.connection);

DataSet dataSet = new DataSet();

sqlDataAdapter.Fill(dataSet, "y");

dataGridView1.DataSource = dataSet.Tables["y"];

DBConnector.Close();

var columnWidth = dataGridView1.Width / 4;

dataGridView1.Columns[0].Visible = false;

string[] gridHeaders = new string[] { "кличка", "пол", "дата рождения", "порода" };

for (int i = 1; i < 5; i++)

{

dataGridView1.Columns[i].HeaderText = gridHeaders[i-1];

dataGridView1.Columns[i].Width = columnWidth;

}

dataGridView1.Columns[dataGridView1.Columns.Count-1].Visible=false;

}

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

{

MedicalReserch medicalReserch = new MedicalReserch((dataGridView1[0, dataGridView1.SelectedCells[0].RowIndex].Value).ToString());

this.Enabled = false;

medicalReserch.Show(this);

}

private bool canAddAnimal()

{

if (textBox1.Text == "")

{

MessageBox.Show("input nickname");

return false;

}

if (richTextBox1.Text == "")

{

MessageBox.Show("input species");

return false;

}

return true;

}

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

{

if (!canAddAnimal()) return;

DBConnector.Open();

var command = new SqlCommand($"exec addAnimal @title,@sex,@born,@spec", DBConnector.connection);

command.Parameters.Add("@title", SqlDbType.NText).Value = textBox1.Text;

command.Parameters.Add("@sex", SqlDbType.Bit).Value = radioButton1.Checked ? 1 : 0;

command.Parameters.Add("@born", SqlDbType.Date).Value = dateTimePicker1.Value.Date;

command.Parameters.Add("@spec", SqlDbType.NText).Value = richTextBox1.Text;

command.ExecuteNonQuery();

DBConnector.Close();

fillAnimalTable();

}

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

{

SaleForm medicalReserch = new SaleForm((dataGridView1[0, dataGridView1.SelectedCells[0].RowIndex].Value).ToString());

this.Enabled = false;

medicalReserch.Show(this);

}

private void Form1\_Activated(object sender, EventArgs e)

{

fillAnimalTable();

}

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

{

Owner.Enabled = true;

Owner.Visible= true;

this.Close();

}

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

{

Owner.Enabled = true;

Owner.Visible = true;

}

private void textBox2\_TextChanged(object sender, EventArgs e)

{

fillAnimalTable(textBox2.Text);

}

private void dataGridView1\_RowPrePaint(object sender, DataGridViewRowPrePaintEventArgs e)

{

foreach (DataGridViewRow item in dataGridView1.Rows)

{

if (!(bool)item.Cells[item.Cells.Count - 1].Value)

{

item.DefaultCellStyle.ForeColor = Color.Gray;

}

}

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace courseWork

{

public partial class MedicalReserch : Form

{

string selectedAnimalId { get; set; }

public MedicalReserch(string selectedAnimalId)

{

InitializeComponent();

fillDataGrid(selectedAnimalId);

getAnimalData(selectedAnimalId);

this.selectedAnimalId= selectedAnimalId;

}

private void fillDataGrid(string selectedAnimalId,string reserch="")

{

DBConnector.Open();

string query = $"select \* from medicalResearchView where animalId={selectedAnimalId}";

if (reserch != "") { query += $" and aboutResearch like N'%{reserch}%'"; }

SqlDataAdapter sqlDataAdapter = new SqlDataAdapter(query, DBConnector.connection);

DataSet dataSet = new DataSet();

sqlDataAdapter.Fill(dataSet, "y");

dataGridView1.DataSource = dataSet.Tables["y"];

DBConnector.Close();

var columnWidth = dataGridView1.Width / 4;

dataGridView1.Columns[0].Visible = false;

dataGridView1.Columns[1].Visible = false;

string[] gridHeaders = new string[] { "животное", "дата обследования", "подробности","цена" };

for (int i = 2; i < 6; i++)

{

dataGridView1.Columns[i].HeaderText = gridHeaders[i - 2];

dataGridView1.Columns[i].Width = columnWidth;

}

}

private void getAnimalData(string selectedAnimalId)

{

DBConnector.Open();

string query = $"select nickname,sex,DATEDIFF(year,bornDate,GETDATE()),species from animal where animalId=@id";

var command = new SqlCommand(query, DBConnector.connection);

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

using (var dataReader = command.ExecuteReader())

{

while (dataReader.Read())

{

nickNameLabel.Text=dataReader.GetString(0);

}

}

DBConnector.Close();

}

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

{

fillDataGrid(selectedAnimalId);

}

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

{

this.Owner.Enabled = true;

}

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

{

var command = new SqlCommand($"exec addMedicalResearch @animalId,@medDate,@about,@price", DBConnector.connection);

command.Parameters.Add("@animalId", SqlDbType.Int).Value = int.Parse(selectedAnimalId);

command.Parameters.Add("@medDate", SqlDbType.Date).Value = dateTimePicker1.Value.Date;

command.Parameters.Add("@about", SqlDbType.NText).Value = richTextBox2.Text;

command.Parameters.Add("@price", SqlDbType.Int).Value = numericUpDown1.Value;

DBConnector.Open();

command.ExecuteNonQuery();

DBConnector.Close();

fillDataGrid(selectedAnimalId);

}

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

{

Owner.Enabled = true;

Owner.Visible = true;

this.Close();

}

private void textBox2\_TextChanged(object sender, EventArgs e)

{

fillDataGrid(selectedAnimalId, textBox2.Text);

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace courseWork

{

public partial class Menu : Form

{

public Menu()

{

InitializeComponent();

}

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

{

Form1 animalInfo = new Form1();

this.Enabled = false;

this.Visible= false;

animalInfo.Show(this);

}

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

{

AllSales salesInfo = new AllSales();

this.Enabled = false;

this.Visible = false;

salesInfo.Show(this);

}

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

{

PurchaseFoodForm salesInfo = new PurchaseFoodForm();

this.Enabled = false;

this.Visible = false;

salesInfo.Show(this);

}

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

{

FeedConsumption salesInfo = new FeedConsumption();

this.Enabled = false;

this.Visible = false;

salesInfo.Show(this);

}

private int[] animalId;

private int[] allFeed;

private int[] allSale;

private int getAllFeed(int animalId)

{

DBConnector.Open();

string query = $"select sum(foodCount)\*dbo.getPriceWithPurchaseFoodId(purchaseFoodId)as d from feedConsumption where animalId={animalId} group by purchaseFoodId";

SqlDataAdapter sqlDataAdapter = new SqlDataAdapter(query, DBConnector.connection);

DataSet dataSet = new DataSet();

sqlDataAdapter.Fill(dataSet, "y");

DBConnector.Close();

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

for (int i = 0; i < dataSet.Tables["y"].Rows.Count; i++)

{

list.Add(int.Parse(dataSet.Tables["y"].Rows[i][0].ToString()));

}

return list.Sum();

}

private int getAllSale(int animalId)

{

DBConnector.Open();

string query = $"select price from sale where animalId={animalId}";

SqlDataAdapter sqlDataAdapter = new SqlDataAdapter(query, DBConnector.connection);

DataSet dataSet = new DataSet();

sqlDataAdapter.Fill(dataSet, "y");

DBConnector.Close();

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

for (int i = 0; i < dataSet.Tables["y"].Rows.Count; i++)

{

list.Add(int.Parse(dataSet.Tables["y"].Rows[i][0].ToString()));

}

return list.Sum();

}

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

{

fillTable();

fillStatistic();

}

private void fillStatistic()

{

DBConnector.Open();

string query = $"select count(animalId) from animal where exist=1 union select sum(foodCount) from purchaseFood; ";

SqlDataAdapter sqlDataAdapter = new SqlDataAdapter(query, DBConnector.connection);

DataSet dataSet = new DataSet();

sqlDataAdapter.Fill(dataSet, "y");

DBConnector.Close();

int animalCount = int.Parse(dataSet.Tables["y"].Rows[0][0].ToString());

int allFoodCount = int.Parse(dataSet.Tables["y"].Rows[1][0].ToString());

label1.Text = $"Количество животных в наличии: {animalCount}";

label2.Text = $"Общее количество корма: {allFoodCount}";

}

private void fillTable()

{

DBConnector.Open();

string query = $"select animalId from animal";

SqlDataAdapter sqlDataAdapter = new SqlDataAdapter(query, DBConnector.connection);

DataSet dataSet = new DataSet();

sqlDataAdapter.Fill(dataSet, "y");

DBConnector.Close();

animalId = new int[dataSet.Tables["y"].Rows.Count];

allFeed = new int[animalId.Length];

allSale = new int[animalId.Length];

for (int i = 0; i < animalId.Length; i++)

{

animalId[i] = int.Parse(dataSet.Tables["y"].Rows[i][0].ToString());

allFeed[i] = getAllFeed(animalId[i]);

allSale[i] = getAllSale(animalId[i]);

}

DataTable dt = new DataTable();

dt.Columns.Add("животное");

dt.Columns.Add("расходы");

dt.Columns.Add("доходы");

dt.Columns.Add("рентабильность");

dt.TableName = "x";

for (int i = 0; i < animalId.Length; i++)

{

dt.Rows.Add(animalId[i], allFeed[i], allSale[i], allSale[i] - allFeed[i]);

}

dataSet.Tables.Add(dt);

dataGridView1.DataSource = dataSet.Tables["x"];

}

private void dataGridView1\_VisibleChanged(object sender, EventArgs e)

{

fillTable();

fillStatistic();

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace courseWork

{

internal static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Menu());

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace courseWork

{

public partial class PurchaseFoodForm : Form

{

public PurchaseFoodForm()

{

InitializeComponent();

}

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

{

Owner.Enabled = true;

Owner.Visible = true;

this.Close();

}

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

{

Owner.Enabled = true;

Owner.Visible = true;

}

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

{

fillPurchaseFoodTable();

dateTimePicker1.MaxDate = DateTime.Now;

dateTimePicker2.MinDate = dateTimePicker3.Value.AddDays(1);

}

private void fillPurchaseFoodTable(string title="")

{

DBConnector.Open();

string query = $"select \* from purchaseFood";

if (title != "") { query += $" where foodTitle like N'%{title}%'"; }

if (numericUpDown3.Value != -1 && title!="") { query += $" and remainder<{numericUpDown3.Value}"; }

else if (numericUpDown3.Value != -1) { query += $" where remainder<{numericUpDown3.Value}"; }

SqlDataAdapter sqlDataAdapter = new SqlDataAdapter(query, DBConnector.connection);

DataSet dataSet = new DataSet();

sqlDataAdapter.Fill(dataSet, "y");

dataGridView1.DataSource = dataSet.Tables["y"];

DBConnector.Close();

var columnWidth = dataGridView1.Width / 5;

dataGridView1.Columns[0].Visible = false;

string[] gridHeaders = new string[] { "названиие", "количество","осталось","СИ","цена за шт", "цена" };

for (int i = 1; i < 6; i++)

{

dataGridView1.Columns[i].HeaderText = gridHeaders[i - 1];

dataGridView1.Columns[i].Width = columnWidth;

}

}

private bool canAddFood()

{

if (textBox1.Text == "")

{

MessageBox.Show("input title");

return false;

}

if (textBox2.Text == "")

{

MessageBox.Show("input unit system");

return false;

}

return true;

}

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

{

if (!canAddFood()) return;

DBConnector.Open();

var command = new SqlCommand($"exec addPurchaseFood @foodTitle,@foodCount,@SI,@priceForOne,@rd", DBConnector.connection);

command.Parameters.Add("@foodTitle", SqlDbType.NText).Value = textBox1.Text;

command.Parameters.Add("@foodCount", SqlDbType.Int).Value = numericUpDown1.Value;

command.Parameters.Add("@SI", SqlDbType.NText).Value = textBox2.Text;

command.Parameters.Add("@priceForOne", SqlDbType.Int).Value = numericUpDown1.Value;

command.Parameters.Add("@rd", SqlDbType.Date).Value = dateTimePicker1.Value.Date;

command.ExecuteNonQuery();

DBConnector.Close();

fillPurchaseFoodTable();

}

private void textBox3\_TextChanged(object sender, EventArgs e)

{

fillPurchaseFoodTable(textBox3.Text);

}

private void numericUpDown3\_ValueChanged(object sender, EventArgs e)

{

fillPurchaseFoodTable(textBox3.Text);

}

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

{

DBConnector.Open();

string query = $"select \* from purchaseFood" +

$" where realizationDate >= '{String.Join(".", dateTimePicker1.Value.ToShortDateString().Split('.').Reverse()).Replace(".", "-")}' and " +

$"realizationDate < '{String.Join(".", dateTimePicker2.Value.ToShortDateString().Split('.').Reverse()).Replace(".", "-")}'";

SqlDataAdapter sqlDataAdapter = new SqlDataAdapter(query, DBConnector.connection);

DataSet dataSet = new DataSet();

sqlDataAdapter.Fill(dataSet, "y");

dataGridView1.DataSource = dataSet.Tables["y"];

DBConnector.Close();

}

private void dateTimePicker3\_ValueChanged(object sender, EventArgs e)

{

dateTimePicker2.MinDate = dateTimePicker3.Value.AddDays(1);

}

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

{

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace courseWork

{

public partial class SaleForm : Form

{

string selectedAnimalId { get; set; }

public SaleForm(string selectedAnimalId)

{

InitializeComponent();

this.selectedAnimalId = selectedAnimalId;

}

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

{

DBConnector.Open();

string query = $"exec addSale @animalId,@saleDate,@price";

var command = new SqlCommand(query, DBConnector.connection);

command.Parameters.Add("@animalId", SqlDbType.Int).Value = selectedAnimalId;

command.Parameters.Add("@saleDate", SqlDbType.Date).Value = dateTimePicker1.Value.Date;

command.Parameters.Add("@price", SqlDbType.Int).Value = numericUpDown1.Value;

command.ExecuteNonQuery();

query = $"update animal set exist=0 WHERE animalId=@id";

command = new SqlCommand(query, DBConnector.connection);

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

command.ExecuteNonQuery();

DBConnector.Close();

this.Owner.Enabled = true;

this.Close();

}

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

{

this.Owner.Enabled = true;

}

}

}