государственное бюджетное профессиональное образовательное учреждение

**«Пермский химико-технологический техникум»**

Специальность 09.02.07 «Информационные системы и программирование»

**КУРСОВОЙ ПРОЕКТ**

**Междисциплинарный курс:** «МДК.11.01Технология разработки и защиты баз данных»

**Тема:** «Подсистема учета кормления животных»

Выполнил студент гр. ИС-20-11

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (И.А. Большаков)

Руководитель проекта:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Л.Ф. Абдалова)

2022

**СОДЕРЖАНИЕ**

[1. Общая часть 4](#_Toc116279102)

[1.1. Описание предметной области 4](#_Toc116279103)

[1.2. Требования к функциональным характеристикам 4](#_Toc116279104)

[1.3 Требования к составу и параметрам технических средств 6](#_Toc116279105)

[2. Описательная часть 8](#_Toc116279106)

[2.1. Выбор технологии проектирования. Выбор СУБД 8](#_Toc116279107)

[2.2 Построение модели системы 10](#_Toc116279108)

[2.3 Проектирование физической структуры базы данных 12](#_Toc116279109)

[2.4 Алгоритм функционирования информационной системы 13](#_Toc116279110)

[2.5 Описание пользовательского интерфейса 14](#_Toc116279111)

[2.6 Тестирование и отладка 20](#_Toc116279112)

[ЗАКЛЮЧЕНИЕ 28](#_Toc116279113)

[СПИСОК ЛИТЕРАТУРЫ 29](#_Toc116279114)

[ТЕКСТ ПРОГРАММЫ 30](#_Toc116279115)

[ПРИЛОЖЕНИЕ 2 39](#_Toc116279116)

[РУКОВОДСТВО ОПЕРАТОРА 39](#_Toc116279117)

**ВВЕДЕНИЕ**

Зоопарки в наше время расположены в каждом городе. И в каждом городе есть свои животные, которых нужно кормить и заботиться о их благополучии. Чем больше животных, тем сложнее закупка качественного корма. Необходимо отслеживать, чтобы корма хватило на всех животных. Так же необходимо отслеживать кормление животных сотрудниками, чтобы понимать, покормил ли кто-то того или иного животного. Данная проблема актуальна и будет затронута в курсовом проекте.

Цель курсового проекта заключается в разработке подсистемы для зоопарка с целью осуществления заказа, отслеживания корма на складе и для отслеживания времени кормления животных.

1. Анализ предметной области
2. Разработка концептуальной модели базы данных
3. Создание и реализация логической модели в конкретной СУБД
4. Создание приложения по работе с базой данных
5. **Общая часть**
   1. **Описание предметной области**

Зоопарк – это учреждение, в котором животные размещаются в вольерах, за ними ухаживают, выставляют на всеобщее обозрение, а в некоторых случаях разводят в целях сохранения. В зоопарках бывают очень много разных видов животных, так же им дают клички и живут они в определенных вольерах. Вольеры в свою очередь бывают разных размеров и обладают своими характеристиками. В них могут проживать разное количество животных.

О корме нам известно название и какое количество хранится на складе. Если корм заканчивается, то необходимо заказать новый. Каждый вновь созданный заказ оформляется только на один вид корма. Информация о расходе корма так же хранится в системе. Сумма расхода корма по каждому факту расхода рассчитывается автоматически и сохраняется в системе. Определенный сотрудник может заказать корм. Для этого необходимо указать название и количества кг корма. В конце заказа мы узнаем сумму.

Разрабатываемое приложение будет создано для зоопарка с целью учёта кормления животных. Директор может просматривать сотрудников и добавлять новых сотрудников, просматривать историю заказов и количество корма на складах. Директор выдает логин и пароль новым сотрудникам для авторизации в приложении. Так же есть старший менеджер и младший менеджер. Старший менеджер осуществляет добавление новых кормов и видов животных в базу данных, а также размещением животных по вольерам. Младший менеджер занимается заказом корма и просмотром последней кормежки животных сотрудниками. Обычный сотрудник кормит животных и отмечается в приложении.

**1.2. Требования к функциональным характеристикам**

База данных должна осуществлять следующие функции:

* хранение и защита данных;

Приложение по работе с базой данных должна осуществлять следующие функции:

* осуществлять добавление, изменение и удаление данных, хранящихся в базе данных;
* просмотр существующих данных о заказе корма, сотрудников и времени кормления животных;

Основные функции информационной системы представлены на рис. 1, а более подробно описываются на рис. 2.

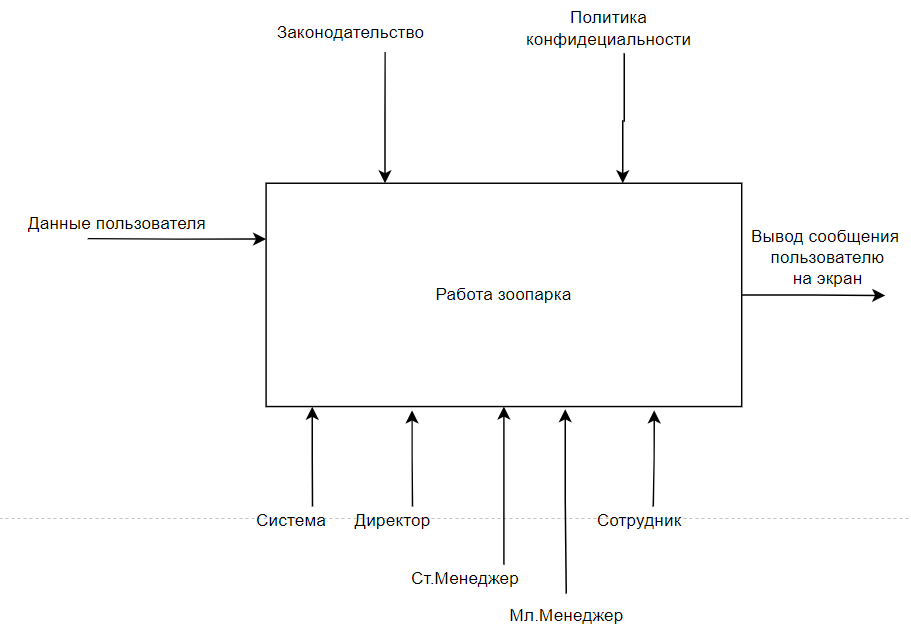


Рис. 1 Функциональная диаграмма А0

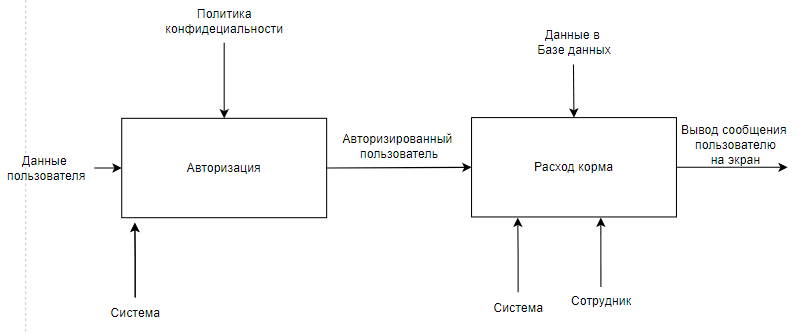


Рис. 2 Функциональная диаграмма А1

Доступные функции в системе и подсистеме для отдельных ролей отображены на рис. 3.

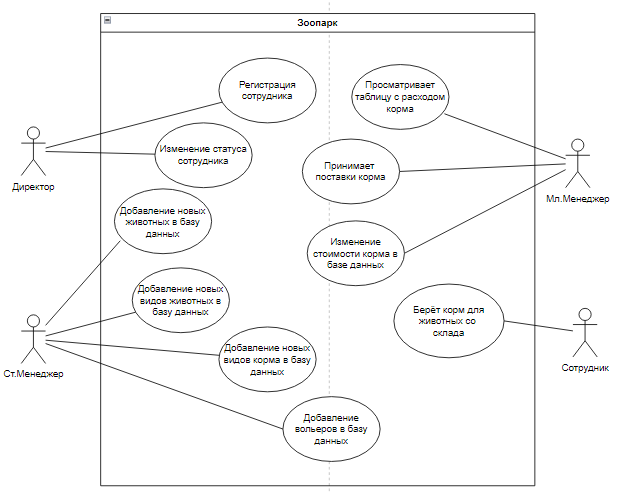


Рис. 3 Диаграмма прецендентов

**1.3 Требования к составу и параметрам технических средств**

Для выполнения программы необходим персональный компьютер. Который состоит из системного блока, монитора, клавиатуры, мыши или ноутбук.

Данная программа может быть откомпилирована и отлажена, если компьютер имеет следующие минимальные технические характеристик:

* 1 ГБ свободного места на жестком диске;
* разрешение экрана не меньше 1024х768 точек;
* операционная система windows 10;
* не менее 512 мб оперативной памяти.

1. **Описательная часть**
   1. **Выбор технологии проектирования. Выбор СУБД**

Проектирование баз данных – процесс создания схемы базы данных и определения необходимых ограничений целостности.

Основные задачи базы данных:

* Обеспечение хранения в БД всей необходимой информации;
* Обеспечение возможности получения данных по всем необходимым запросам;
* Сокращение избыточности и дублирования данных;
* Обеспечение целостности базы данных.

Семантическое проектирование — это подход к реализации приложения, целью которого является создание логической иерархии базы данных, формирование структуры приложения.

Концептуальное проектирование — начальный этап проектирования, при котором создается черновой вариант продукта, показывающий принцип закладываемой логики для первоначального согласования и проверки ограничений средств разработки. Концептуальный прототип создается без направления на конкретную СУБД и модель данных.

Логическое проектирование – это процесс создания модели используемой на предприятии информации на основе выбранной модели организации данных, но без учета типа целевой СУБД и других физических аспектов реализации. Для реляционной модели данных логическая модель — набор схем отношений, обычно с указанием первичных ключей, а также «связей» между отношениями, представляющих собой внешние ключи.

Физическое проектирование — создание схемы базы данных для конкретной СУБД. Специфика конкретной СУБД может включать в себя ограничения на именование объектов базы данных, ограничения на поддерживаемые типы данных и другие параметры.

Выполнив все вышеперечисленные этапы, поможет в полной мере выполнить все задачи и разработать базу данных, которая удовлетворит всем обязательным требованиям для ее администрирования и внедрения в работу.

Для разработки базы данных нужно выбрать конкретную СУБД.

Можно выделить ряд критериев, по которым следует проводить выбор СУБД:

* модель данных;
* дополнительные возможности;
* особенности архитектуры и функциональные возможности;
* особенности разработки приложений;
* производительность;
* надежность;

Существуют множество различных баз данных, между которыми нужно выбрать одну. База данных должна удовлетворять все критерия и быть удобной. Рассмотрим 3 основные базы данных:

SQLite — это быстрая и легкая встраиваемая одно файловая СУБД на языке C, которая не имеет сервера и позволяет хранить всю базу локально на одном устройстве. Для работы SQLite не нужны сторонние библиотеки или службы.

Microsoft SQL Server — система управления реляционными базами данных, разработанная корпорацией Microsoft. Основной используемый язык запросов — Transact-SQL. Используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия.

PostgreSQL — это свободно распространяемая объектно-реляционная система управления базами данных (ORDBMS), наиболее развитая из открытых СУБД в мире и являющаяся реальной альтернативой коммерческим базам данных.

Преимущества SQLite заключаются в файловой системе, то есть все данные хранятся в одном файле. SQLite использует язык запросов SQL, но не все функции доступны. Так же в этой СУБД есть свои недостатки, такие как отсутствие возможности управлять связями в таблицах и невозможность дополнительной настройки.

PostgreSQL имеет полную SQL-совместимость, расширяемость, объектно-ориентированность. Но у PostgreSQL есть проблемы с производительностью и данное СУБД не так распространено, как, например, SQL servrer.

Microsoft SQL Server самая популярная СУБД и достаточна проста в использовании. Имеет много функций, хорошую безопасность, производительность.

Сравнив все СУБД по критериям, можно понять, что Microsoft SQL Server больше походит для решения задач курсового проекта в силу своей доступности и удобства в использовании.

* 1. **Построение модели системы**

**2.2.1 Построение концептуальной модели. ER-диаграмма**

Концептуальная модель представляет собой высокоуровневый взгляд на предметную область. На данном этапе не учитывается модель данных и физические аспекты представления и хранения данных, проектирование одинаково для любой базы данных. ER-диаграмма показана на рис. 4.

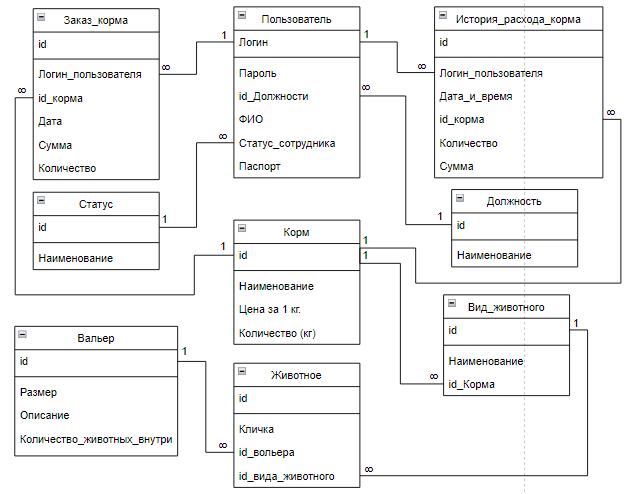


Рис. 4 ER-Диаграмма

* + 1. **Описание логической структуры**

Логическая модель описывает понятия предметной области, их взаимосвязь, а также ограничения на данные, налагаемые предметной областью. Графическое представление логической структуры представлено на рис. 5.

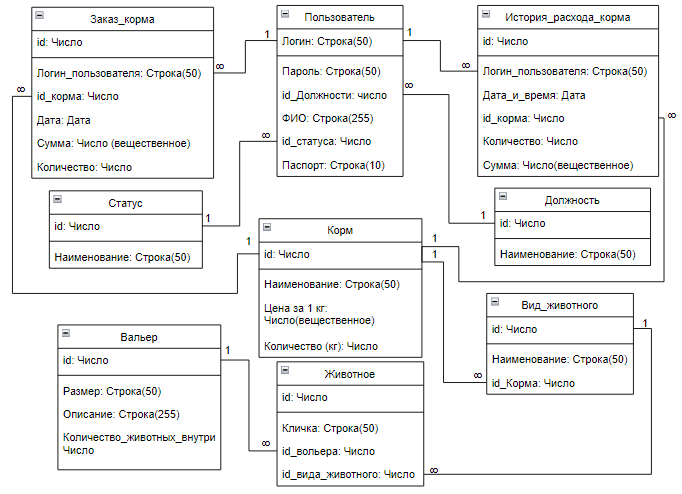
****

Рис. 5 Логическая модель

* 1. **Проектирование физической структуры базы данных**

Физическая модель базы данных декларирует и описывает возможность размещения и связи данных в текущей среде хранения, а также способы доступа. Физическая модель представлена на рис. 6.

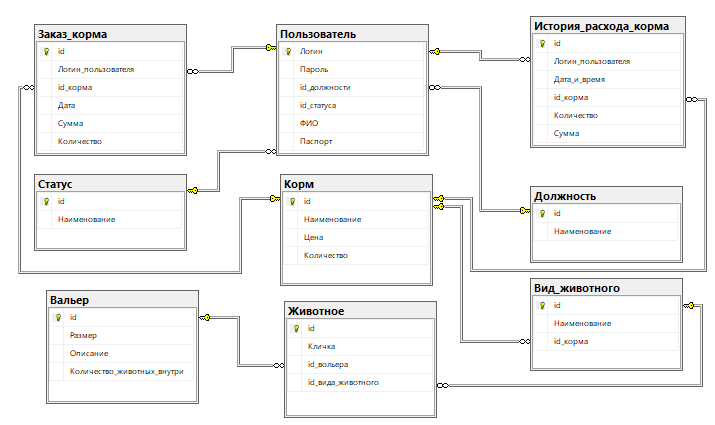


Рис. 6 Физическая модель

* 1. **Алгоритм функционирования информационной системы**

Алгоритм функционирования информационной системы представления в виде блок-схемы, которая представлены на рис. 7.

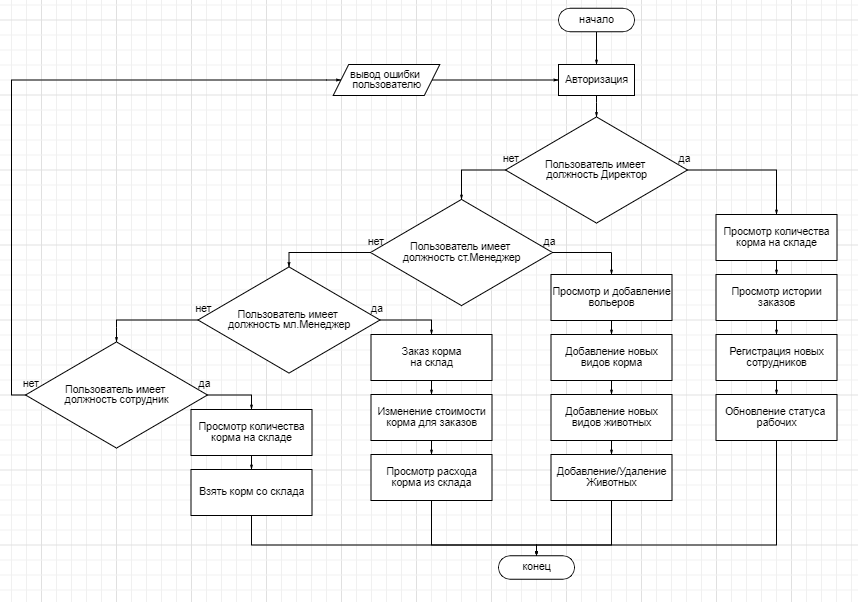


Рис. 7 Блок-схема

* 1. **Описание пользовательского интерфейса**

Пользовательский интерфейс программы – это связующее звено между пользователем и компьютером, выполняющим эту программу.

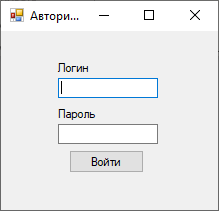


Рис. 8 - Форма авторизации

При входе в систему под ролью Директор можно изменить статус сотрудника и зарегистрировать нового пользователя(рис.12)

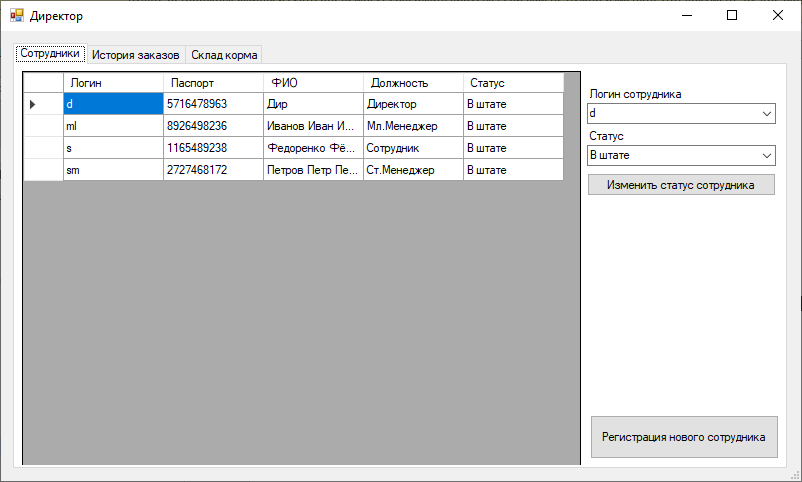


Рис. 9 Форма директора (сотрудники)

На следующей вкладке можно посмотреть историю заказов корма

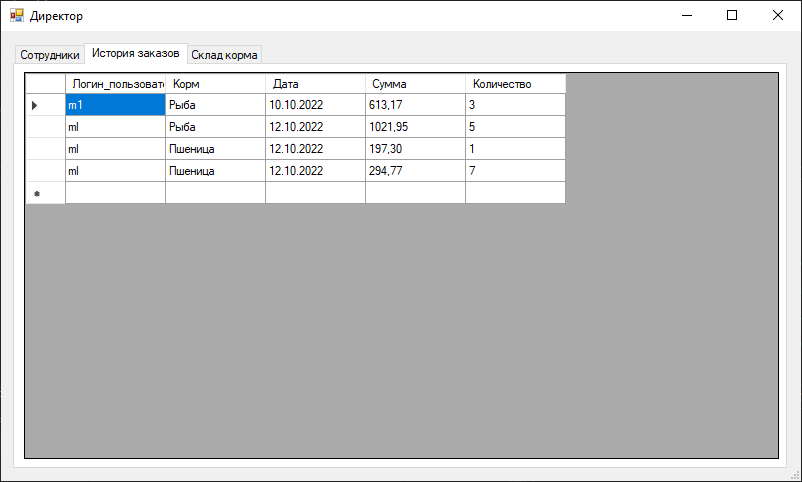


Рис. 10 Форма директора (История заказов)

На вкладке склад корма, пользователь увидит количество и общую стоимость корма, находящегося в информационной базе данных

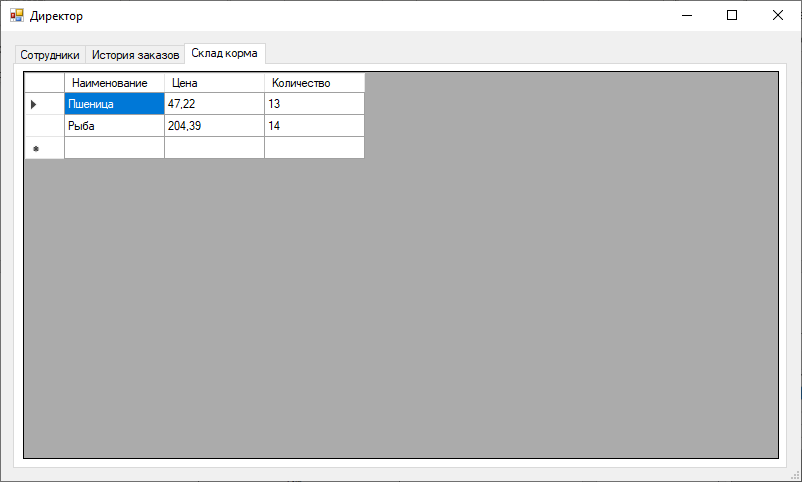


Рис. 11 Форма директора (Склад с кормом)

По нажатию кнопки «зарегистрировать нового сотрудника» откроется форма с регистрацией (рис. 12), на которой нужно заполнить все поля и нажать на кнопку «Регистрация» для добавления нового пользователя в информационную базу данных

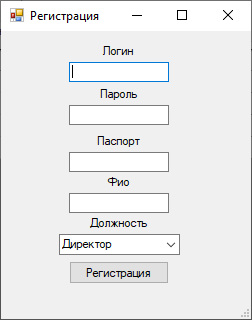


Рис. 12 Форма регистрации нового сотрудника

При входе в систему под ролью «Младший менеджер» Пользователь сможет выбрать нужный корм и изменить его цену за кг. либо же заказать выбранный корм на склад указав количество заказываемого корма.

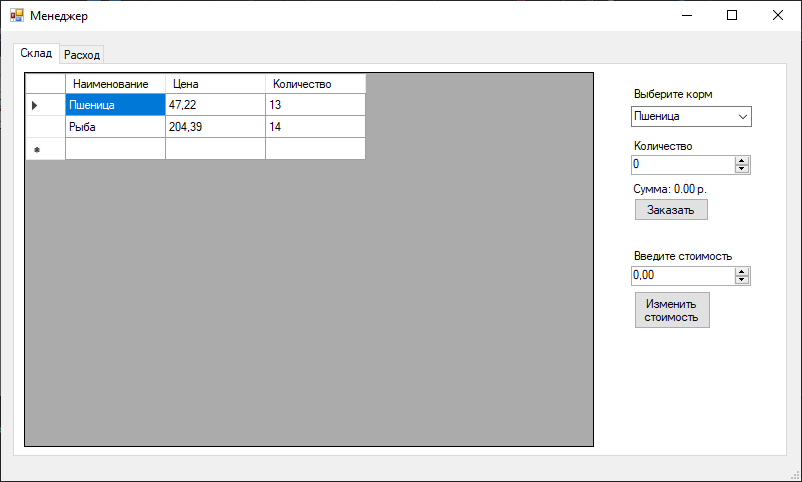


Рис. 13 Форма склада с кормом для младшего менеджера

Так же он может просматривать историю кормлении животных.

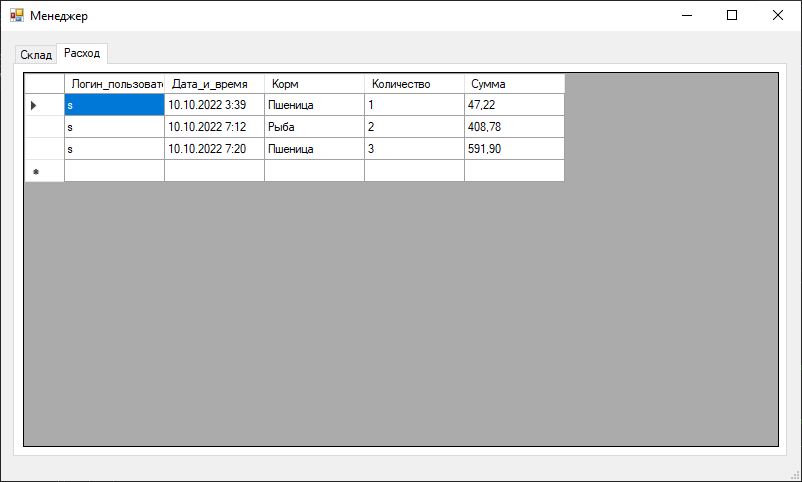


Рис. 14 Форма истории расхода корма со склада для младшего менеджера

При входе в систему под ролью «Старший менеджер». Пользователь сможет добавить новый вольер в базу данных указав его размер и описание после чего нажать на кнопку «добавить»

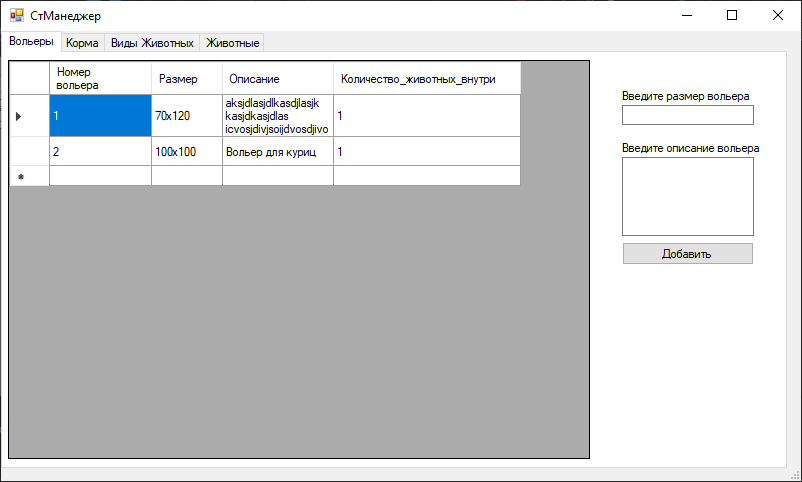


Рис. 15 Форма для старшего менеджера (добавление/просмотр вольеров)

На вкладке «Корма» пользователь может просмотреть доступные корма на складе и добавить новый вид корма.

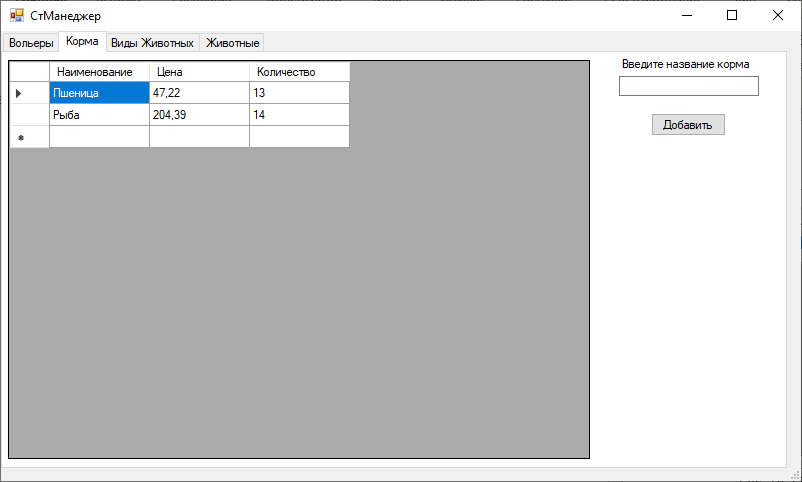


Рис. 16 Форма для старшего менеджера (добавление/просмотр нового корма)

На вкладе «Виды животных» пользователь может видеть все существующие виды животных из базы данных и добавлять новые виды животных в нее

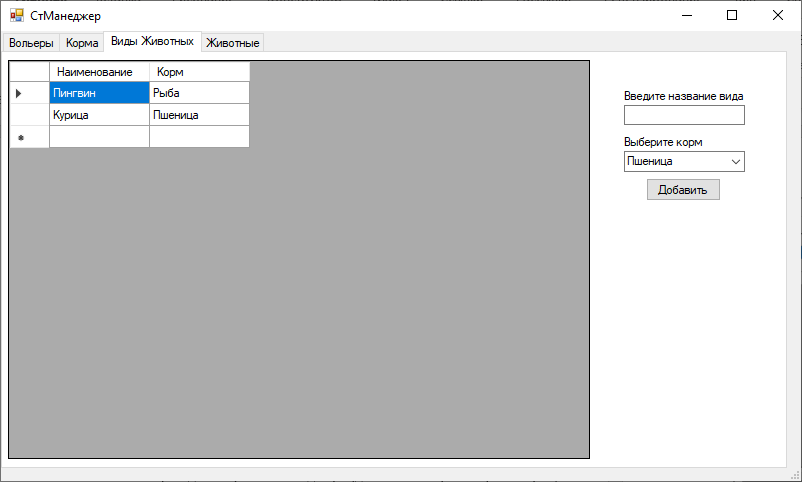


Рис. 17

Форма для старшего менеджера

(добавление/просмотр новых видов животных)

На вкладке «Животные» пользователь сможет добавить новое животное в базу данных указав его кличку, номер вольера и его вид, а также удалить животное из базы данных, выбрав его по кличке

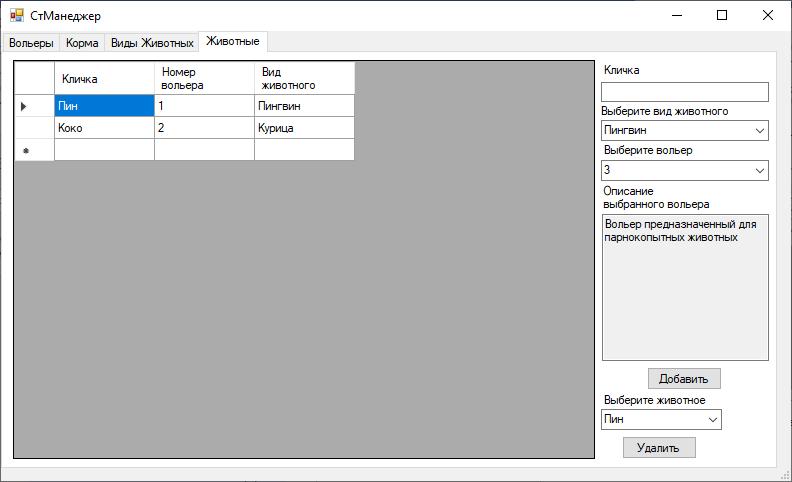


Рис. 18

Форма для старшего менеджера (добавление/удаление животных)

При входе в систему под ролью «Сотрудник» пользователь сможет видеть информацию со склада корма и брать из него введенное количество корма

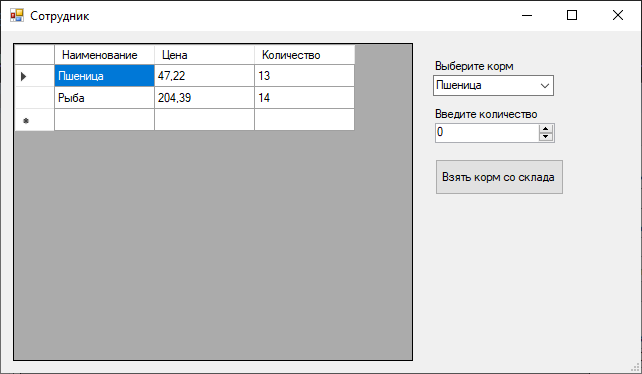


Рис. 18 Форма для обычного сотрудника (удаление корма со склада)

**2.6** **Тестирование и отладка**

Тестирование в исключительных условиях:

|  |  |
| --- | --- |
| **Тест Case #** | 1 |
| **Приоритет тестирования** | Средний |
| **Заголовок/название теста** | Авторизация в системе неизвестного пользователя |
| **Краткое изложение теста** | Ввод неправильных данных в поля ввода |
| **Этапы теста** | 1. Запуск программы  2. Ввод логина и пароля пользователя в текстовое поле  3. Нажатие на кнопку «Вход» |
| **Тестовые данные** | Логин – test  Пароль – test |
| **Ожидаемый результат** | Вывод сообщения о неправильно введенных данных. |
| **Фактический результат** | Рис. 19 |
| **Статус** | Pass |
| **Предварительное условие** | Запуск программы, авторизация |
| **Постусловие** | Вывод сообщения об ошибке. |
| **Примечания/комментарии** |  |

|  |  |
| --- | --- |
| **Тест Case #** | 2 |
| **Приоритет тестирования** | Средний |
| **Заголовок/название теста** | Ввод уже существующего логина при регистрации |
| **Краткое изложение теста** | Ввод некорректных данных |
| **Этапы теста** | 1. Запуск программы  2. Вход под ролю Директор  3. Переход на форму «Регистрация»  4. Ввод в поле «Логин» уже существующего логина в системе  5. Нажатие на кнопку «регистарция» |
| **Тестовые данные** | Логин: d |
| **Ожидаемый результат** | Вывод сообщения о существовании введенного логина в системе |
| **Фактический результат** | Рис. 20 |
| **Статус** | Pass |
| **Предварительное условие** | Запуск программы, авторизация, переход на форму «Регистрация», заполнение остальных полей для регистрации |
| **Постусловие** | Вывод сообщения об ошибке. |
| **Примечания/комментарии** |  |

Тестирование в экстремальных условиях:

|  |  |
| --- | --- |
| **Тест Case #** | 3 |
| **Приоритет тестирования** | Средний |
| **Заголовок/название теста** | Тестирование процедуры незаполненных полей в текстовых полях на форме «Авторизациии» |
| **Краткое изложение теста** | Незаполненные поля логина и пароля в окне «авторизации» |
| **Этапы теста** | 1. Запуск программы  2. Нажатие на кнопку «Вход» |
| **Тестовые данные** | Логин – «»  Пароль – «» |
| **Ожидаемый результат** | Вывод сообщения о неправильно введенных данных. |
| **Фактический результат** | Рис. 21 |
| **Статус** | Pass |
| **Предварительное условие** | Запуск программы, авторизация |
| **Постусловие** | Вывод сообщения об ошибке. |
| **Примечания/комментарии** |  |

|  |  |
| --- | --- |
| **Тест Case #** | 4 |
| **Приоритет тестирования** | Средний |
| **Заголовок/название теста** | Тестирование процедуры незаполненных текстовых полей на форме «регистрация» |
| **Краткое изложение теста** | Незаполненные поля логина и пароля в окне «регистрация нового клиента» |
| **Этапы теста** | 1. Запуск программы  2. Вход под ролю Директор  3. Переход на форму «регистрация»  4. Нажатие на кнопку «регистрация» |
| **Тестовые данные** | Логин – «»  Пароль – «»  Паспорт – «»  ФИО – «» |
| **Ожидаемый результат** | Вывод сообщения о необходимости заполнить все поля |
| **Фактический результат** | Рис. 22 |
| **Статус** | Pass |
| **Предварительное условие** | Запуск программы, авторизация, переход на форму «регистрация» |
| **Постусловие** | Вывод сообщения об ошибке. |
| **Примечания/комментарии** |  |

Тестирование в нормальных условиях:

|  |  |
| --- | --- |
| **Тест Case #** | 5 |
| **Приоритет тестирования** | Средний |
| **Заголовок/название теста** | Тестирование процедуры добавление нового сотрудника в систему |
| **Краткое изложение теста** | Ввод корректных данных в текстовые поля на форме «Регистрация нового сотрудника» |
| **Этапы теста** | 1. Запуск программы  2. Вход под ролью Директор  3. Переход на форму «регистрация нового сотрудника»  4. Заполнить поля данными  5. Нажатие на кнопку «зарегистрировать» |
| **Тестовые данные** | Логин-«pd»  Пароль-«pd»  Паспорт-«1234567890»  ФИО-«Прокофьев Денис»  Должность-«Сотрудник» |
| **Ожидаемый результат** | Вывод сообщения об успешной регистрации сотрудника |
| **Фактический результат** | Рис. 23 |
| **Статус** | Pass |
| **Предварительное условие** | Запуск программы, авторизация, переход на форму «регистрация нового сотрудника» |
| **Постусловие** | Вывод сообщения об успешной регистрации сотрудника. |
| **Примечания/комментарии** |  |

|  |  |
| --- | --- |
| **Тест Case #** | 6 |
| **Приоритет тестирования** | Средний |
| **Заголовок/название теста** | Тестирование процедуры добавление нового вольера в систему |
| **Краткое изложение теста** | Ввод корректных данных в текстовые поля на форме «Старшего менеджера» |
| **Этапы теста** | 1. Запуск программы  2. Вход сотрудника с правами Старшего менеджера  3. Переход на форму «Вольеры»  4. Заполнить поля данными  5. Нажатие на кнопку «Добавить» |
| **Тестовые данные** | Размер вольера – «17x17 m»  Описание вольера– «Вольер предназначенный для парнокопытных животных» |
| **Ожидаемый результат** | Вывод сообщения об успешном добавлении товара в систему |
| **Фактический результат** | Рис. 24 |
| **Статус** | Pass |
| **Предварительное условие** | Запуск программы, авторизация, переход на форму «Вольеры» |
| **Постусловие** | Вывод сообщения об успешном добавлении товара |
| **Примечания/комментарии** |  |

# **ЗАКЛЮЧЕНИЕ**

В ходе выполнения курсовой работы была разработана база данных для хранения и работы с информацией о животных, сотрудниках и продуктах.

Разработанная система удовлетворяет всем требованиям, поставленным на этапе постановки задачи. Данная система ориентирована на сотрудников зоопарка.

Возможностями данной системы являются:

- Осуществление администрирования системы

- Учет заказов корма

- Учет расхода корма

- Учет животных в вольерах

Данная система может использоваться для автоматизации процессов учета зоопарка.

**СПИСОК ЛИТЕРАТУРЫ**

1. Metanit. Полное руководство по C# 10 и .NET 6 [электронный ресурс] / адрес: https://metanit.com/sharp/tutorial/
2. Metanit. Полное руководство по MS SQL Server 2019 /

адрес: https://metanit.com/sql/sqlserver/

**ПРИЛОЖЕНИЕ 1**

## **ТЕКСТ ПРОГРАММЫ**

**Код класса «Авторизация»**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Diagnostics.Eventing.Reader;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WindowsFormsApp2

{

public partial class Авторизация : Form

{

static string dbname = @"KOMPUTER\SQLEXPRESS";

public static SqlConnection sqlcon = new SqlConnection(@"Data Source=" + dbname + ";Initial Catalog=Zoo1;Integrated Security=True");

public static string la ="";

public Авторизация()

{

InitializeComponent();

}

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

{

Авторизация.la = getLogin();

var password = getPassowrd();

var userData = new User { };

try

{

userData = getUserData(Авторизация.la);

}

catch (Exception)

{

MessageBox.Show("Данные введенны неверно!");

return;

}

if (userData.password != password)

{

MessageBox.Show("Данные введенны неверно!");

return;

}

Авторизация.sqlcon.Close();

MessageBox.Show($"Добро Пожаловать {userData.name}!");

ActiveForm.Hide();

switch (userData.role)

{

case 1:

new Сотрудник().ShowDialog();

break;

case 2:

new МлМенеджер().ShowDialog();

break;

case 3:

new СтМенеджер().ShowDialog();

break;

case 4:

new Директор().ShowDialog();

break;

}

Close();

}

private string getLogin()

{

return textBox1.Text;

}

private string getPassowrd()

{

// TODO: err if pass is nil

return maskedTextBox1.Text;

}

private User getUserData(string login)

{

Авторизация.sqlcon.Open();

SqlDataReader dataReader = new SqlCommand($"SELECT Пароль, id\_Должности, ФИО FROM Пользователь WHERE Пользователь.Логин = '{login}'", sqlcon).ExecuteReader();

if (!dataReader.Read())

{

dataReader.Close();

Авторизация.sqlcon.Close();

throw new Exception("no rows");

}

var user = new User

{

password = dataReader.GetString(0),

role = (int)dataReader.GetValue(1),

name = dataReader.GetString(2)

};

dataReader.Close();

Авторизация.sqlcon.Close();

return user;

}

}

struct User

{

public string password;

public int role;

public string name;

}

}

**Код класса «Директор»**

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 WindowsFormsApp2

{

public partial class Директор : Form

{

public Директор()

{

InitializeComponent();

}

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

{

updDB();

}

public void updDB()

{

// TODO: данная строка кода позволяет загрузить данные в таблицу "zooDataSet.Статус". При необходимости она может быть перемещена или удалена.

this.статусTableAdapter.Fill(this.zooDataSet.Статус);

// TODO: данная строка кода позволяет загрузить данные в таблицу "zooDataSet.Пользователь". При необходимости она может быть перемещена или удалена.

this.пользовательTableAdapter.Fill(this.zooDataSet.Пользователь);

DataSet ds = new DataSet();

Авторизация.sqlcon.Open();

new SqlDataAdapter(

$@"SELECT Логин, Паспорт, ФИО, Должность.Наименование, Статус.Наименование FROM Пользователь

join Должность on id\_должности = Должность.id

join Статус on id\_Статуса = Статус.id",

Авторизация.sqlcon).

Fill(ds);

dataGridView1.DataSource = ds.Tables[0];

dataGridView1.Columns[3].HeaderText = "Должность";

dataGridView1.Columns[4].HeaderText = "Статус";

ds = new DataSet();

new SqlDataAdapter(

$@"SELECT Заказ\_корма.Логин\_пользователя, Корм.Наименование, Дата, Сумма, Заказ\_корма.Количество

from Заказ\_корма

join Корм on Корм.id = id\_корма",

Авторизация.sqlcon).

Fill(ds);

dataGridView2.DataSource = ds.Tables[0];

dataGridView2.Columns[1].HeaderText = "Корм";

ds = new DataSet();

new SqlDataAdapter(

$@"SELECT Наименование, Цена, Количество FROM Корм",

Авторизация.sqlcon).

Fill(ds);

dataGridView3.DataSource = ds.Tables[0];

Авторизация.sqlcon.Close();

}

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

{

Регистрация f = new Регистрация();

f.ShowDialog();

updDB();

}

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

{

Авторизация.sqlcon.Open();

try

{

new SqlDataAdapter(

$@"UPDATE Пользователь SET

id\_Статуса = {statusInput.SelectedValue}

WHERE Логин = '{loginIput.SelectedValue}'",

Авторизация.sqlcon).

Fill(new DataSet());

MessageBox.Show("Статус работника успешно изменен!");

}

catch (Exception)

{

MessageBox.Show("Выберите сотрудника и статус");

return;

}

finally

{

Авторизация.sqlcon.Close();

updDB();

}

}

}

}

**Код класса «МлМенеджер»**

using Syncfusion.Windows.Forms.Tools;

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;

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

namespace WindowsFormsApp2

{

public partial class МлМенеджер : Form

{

public МлМенеджер()

{

InitializeComponent();

}

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

{

updDB();

}

public static DataSet ds = new DataSet();

public void updDB()

{

// TODO: данная строка кода позволяет загрузить данные в таблицу "zooDataSet.Корм". При необходимости она может быть перемещена или удалена.

this.кормTableAdapter.Fill(this.zooDataSet.Корм);

Авторизация.sqlcon.Open();

ds = new DataSet();

// Склад

new SqlDataAdapter("select Наименование, Цена, Количество from Корм", Авторизация.sqlcon).Fill(ds);

dataGridView1.DataSource = ds.Tables[0];

ds = new DataSet();

// История Расхода

new SqlDataAdapter(

@"SELECT Логин\_пользователя, Дата\_и\_время, Корм.Наименование, История\_расхода\_корма.Количество, Сумма

FROM История\_расхода\_корма

join Корм on Корм.id = id\_корма",

Авторизация.sqlcon).

Fill(ds);

dataGridView3.DataSource = ds.Tables[0];

dataGridView3.Columns[2].HeaderText = "Корм";

Авторизация.sqlcon.Close();

}

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

{

var num = costInputForUpdateCost.Value.ToString().Replace(",",".");

Авторизация.sqlcon.Open();

ds = new DataSet();

new SqlDataAdapter(

$@"update Корм set Цена = {num}

where Корм.id = {feedInput.SelectedValue}",

Авторизация.sqlcon).

Fill(ds);

Авторизация.sqlcon.Close();

MessageBox.Show("Цена Успешно изменена!");

updDB();

}

public string sum = "0";

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

{

Авторизация.sqlcon.Open();

ds = new DataSet();

new SqlDataAdapter(

$@"update Корм set Количество = Количество + {feedCount.Value}

where id = {feedInput.SelectedValue}",

Авторизация.sqlcon).

Fill(ds);

ds = new DataSet();

new SqlDataAdapter(

$@"INSERT INTO Заказ\_корма(Логин\_пользователя, id\_Корма, Дата, Сумма, Количество)

VALUES('{Авторизация.la}',{feedInput.SelectedValue}, '{DateTime.Today.ToString()}', {sum}, {feedCount.Value})",

Авторизация.sqlcon).

Fill(ds);

Авторизация.sqlcon.Close();

MessageBox.Show("Корм успешно заказан!");

updDB();

}

public double cost = 0;

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

{

sum = (Convert.ToDouble(feedCount.Value) \* cost).ToString().Replace(",",".");

label5.Text = $"Сумма: {sum} р.";

}

private void feedInput\_SelectedValueChanged(object sender, EventArgs e)

{

try

{

Авторизация.sqlcon.Open();

SqlDataReader r = new SqlCommand($"SELECT Корм.Цена FROM Корм WHERE Корм.id = {feedInput.SelectedValue}", Авторизация.sqlcon).ExecuteReader();

r.Read();

cost = Convert.ToDouble(r.GetValue(0));

r.Close();

}

catch (Exception)

{

}

finally

{

Авторизация.sqlcon.Close();

}

}

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

{

costInputForUpdateCost.Value = Math.Round(costInputForUpdateCost.Value, 2);

}

}

}

**Код класса «Регистрация»**

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;

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

namespace WindowsFormsApp2

{

public partial class Регистрация : Form

{

public Регистрация()

{

InitializeComponent();

}

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

{

if (loginInput.Text == "" || passwordInput.Text == "" || passInput.Text == "" || nameInput.Text == "" || roleInput.Text == "")

{

MessageBox.Show("Заполните все поля");

return;

}

Авторизация.sqlcon.Open();

try

{

new SqlDataAdapter(

$@"INSERT INTO Пользователь(Логин, Пароль, id\_Должности, id\_Статуса, ФИО, Паспорт)

VALUES('{loginInput.Text}', '{passwordInput.Text}', {roleInput.SelectedValue}, 1, '{nameInput.Text}', '{passInput.Text}')",

Авторизация.sqlcon).

Fill(new DataSet());

Авторизация.sqlcon.Close();

MessageBox.Show("Запись прошла успешно!");

ActiveForm.Close();

}

catch (Exception)

{

MessageBox.Show("Такой Логин уже существует");

Авторизация.sqlcon.Close();

}

}

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

{

// TODO: данная строка кода позволяет загрузить данные в таблицу "zooDataSet.Должность". При необходимости она может быть перемещена или удалена.

this.должностьTableAdapter.Fill(this.zooDataSet.Должность);

}

}

}

**Код класса «Сотрудник»**

using Syncfusion.Windows.Forms.Tools;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.Common;

using System.Data.SqlClient;

using System.Drawing;

using System.Globalization;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

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

namespace WindowsFormsApp2

{

public partial class Сотрудник : Form

{

public Сотрудник()

{

InitializeComponent();

}

public void updDB()

{

// TODO: данная строка кода позволяет загрузить данные в таблицу "zooDataSet.Корм". При необходимости она может быть перемещена или удалена.

this.кормTableAdapter.Fill(this.zooDataSet.Корм);

Авторизация.sqlcon.Open();

DataSet ds = new DataSet();

new SqlDataAdapter($"SELECT Наименование, Цена, Количество FROM Корм", Авторизация.sqlcon).Fill(ds);

dataGridView1.DataSource = ds.Tables[0];

Авторизация.sqlcon.Close();

}

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

{

updDB();

}

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

{

if (numericUpDown1.Value < 1)

{

MessageBox.Show("Введите количество корма");

return;

}

DataSet ds = new DataSet();

Авторизация.sqlcon.Open();

SqlDataReader r = new SqlCommand($"SELECT Количество, Цена FROM Корм where id = {comboBox1.SelectedValue}", Авторизация.sqlcon).ExecuteReader();

r.Read();

int count = (int)r.GetValue(0);

double cost = Convert.ToDouble(r.GetValue(1));

r.Close();

Авторизация.sqlcon.Close();

if(count < (int)numericUpDown1.Value)

{

MessageBox.Show("На складе не хватает корма!");

return;

}

Авторизация.sqlcon.Open();

new SqlDataAdapter(

$@"update Корм set Количество = Количество - {numericUpDown1.Value}

where id = {comboBox1.SelectedValue}",

Авторизация.sqlcon).

Fill(ds);

string sum = (Convert.ToDouble(numericUpDown1.Value) \* cost).ToString().Replace(",",".");

ds = new DataSet();

new SqlDataAdapter(

$@"insert into История\_расхода\_корма(Логин\_пользователя, Дата\_и\_время, id\_корма, Количество, Сумма)

VALUES('{Авторизация.la}', '{DateTime.Now.ToString("yyyy/MM/dd HH:mm:ss")}', {comboBox1.SelectedValue},

{numericUpDown1.Value.ToString().Replace(",",".")},{sum})",

Авторизация.sqlcon).

Fill(ds);

Авторизация.sqlcon.Close();

MessageBox.Show($"Было успешно взято {numericUpDown1.Value} кг корма \nНа сумму в {sum} руб.");

updDB();

}

}

}

**Код класса «СтМенеджер»**

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;

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

namespace WindowsFormsApp2

{

public partial class СтМенеджер : Form

{

public СтМенеджер()

{

InitializeComponent();

}

public static DataSet ds = new DataSet();

public void updDB()

{

// TODO: данная строка кода позволяет загрузить данные в таблицу "zooDataSet.Животное". При необходимости она может быть перемещена или удалена.

this.животноеTableAdapter.Fill(this.zooDataSet.Животное);

// TODO: данная строка кода позволяет загрузить данные в таблицу "zooDataSet.Вид\_животного". При необходимости она может быть перемещена или удалена.

this.вид\_животногоTableAdapter.Fill(this.zooDataSet.Вид\_животного);

// TODO: данная строка кода позволяет загрузить данные в таблицу "zooDataSet.Вальер". При необходимости она может быть перемещена или удалена.

this.вальерTableAdapter.Fill(this.zooDataSet.Вальер);

// TODO: данная строка кода позволяет загрузить данные в таблицу "zooDataSet.Корм". При необходимости она может быть перемещена или удалена.

this.кормTableAdapter.Fill(this.zooDataSet.Корм);

Авторизация.sqlcon.Open();

ds = new DataSet();

new SqlDataAdapter(

$@"SELECT id, Размер, Описание, Количество\_животных\_внутри FROM Вальер",

Авторизация.sqlcon).

Fill(ds);

dataGridView1.DataSource = ds.Tables[0];

dataGridView1.Columns[0].HeaderText = "Номер вольера";

ds = new DataSet();

new SqlDataAdapter("SELECT Наименование, Цена, Количество FROM Корм", Авторизация.sqlcon).Fill(ds);

dataGridView2.DataSource = ds.Tables[0];

ds = new DataSet();

new SqlDataAdapter(

$@"SELECT Вид\_животного.Наименование, Корм.Наименование FROM Вид\_животного

JOIN Корм on Корм.id = Вид\_животного.id\_корма",

Авторизация.sqlcon).

Fill(ds);

dataGridView3.DataSource = ds.Tables[0];

dataGridView3.Columns[1].HeaderText = "Корм";

ds = new DataSet();

new SqlDataAdapter(

@"SELECT Кличка, id\_вольера, Вид\_животного.Наименование FROM Животное

JOIN Вид\_животного ON Вид\_животного.id = id\_вида\_животного",

Авторизация.sqlcon).

Fill(ds);

dataGridView4.DataSource = ds.Tables[0];

dataGridView4.Columns[2].HeaderText = "Вид животного";

Авторизация.sqlcon.Close();

}

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

{

updDB();

}

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

{

if (feedInputForCreateFeed.Text.Length < 1)

{

MessageBox.Show("Заполните название корма!");

return;

}

Авторизация.sqlcon.Open();

ds = new DataSet();

try

{

new SqlDataAdapter(

$@"INSERT INTO Корм(Наименование, Количество, Цена)

VALUES('{feedInputForCreateFeed.Text}', 0, 0)",

Авторизация.sqlcon).

Fill(ds);

MessageBox.Show("Корм был успешно добавлен");

}

catch (Exception)

{

MessageBox.Show("Произошла ошибка при добавлении");

}

Авторизация.sqlcon.Close();

updDB();

}

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

{

if (!(sizeForCreateValier.Text.Length > 0))

{

MessageBox.Show("Заполните размер вольера!");

return;

}

Авторизация.sqlcon.Open();

ds = new DataSet();

try

{

new SqlDataAdapter(

$@"INSERT INTO Вальер(Размер, Описание, [Количество\_животных\_внутри])

VALUES ('{sizeForCreateValier.Text}','{DescriptionForCreateValier.Text}',0)",

Авторизация.sqlcon).

Fill(ds);

MessageBox.Show("Новый Вольер был успешно добавлен");

}

catch (Exception)

{

MessageBox.Show("Произошла ошибка при добавлении");

}

Авторизация.sqlcon.Close();

updDB();

}

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

{

if (!(animalNameInputForCreate.Text.Length > 0) || !(feedInputForAnimalCreate.Text.Length > 0))

{

MessageBox.Show("Заполните все поля для добавления!");

return;

}

Авторизация.sqlcon.Open();

ds = new DataSet();

try

{

new SqlDataAdapter(

$@"INSERT INTO Вид\_животного(Наименование, id\_Корма)

VALUES('{animalNameInputForCreate.Text}', {feedInputForAnimalCreate.SelectedValue})",

Авторизация.sqlcon).

Fill(ds);

MessageBox.Show("Новый вид животного был успешно добавлен");

}

catch (Exception)

{

MessageBox.Show("Произошла ошибка при добавлении");

}

Авторизация.sqlcon.Close();

updDB();

}

private void button5\_Click(object sender, EventArgs e)

{

if (!(textBox4.Text.Length > 0) || !(comboBox3.Text.Length > 0) || !(comboBox4.Text.Length > 0))

{

MessageBox.Show("Заполните все поля для добавления!");

return;

}

Авторизация.sqlcon.Open();

ds = new DataSet();

try

{

new SqlDataAdapter(

$@"INSERT INTO Животное(Кличка, id\_вольера, [id\_вида\_животного])

VALUES('{textBox4.Text}', {comboBox3.SelectedValue}, {comboBox4.SelectedValue})",

Авторизация.sqlcon).

Fill(ds);

MessageBox.Show("Новое животное было успешно добавлено в Базу Данных!");

}

catch (Exception)

{

MessageBox.Show("Данная Кличка уже занята!");

}

ds = new DataSet();

new SqlDataAdapter(

$@"UPDATE Вальер SET [Количество\_животных\_внутри] = [Количество\_животных\_внутри] + 1

WHERE id = {comboBox3.SelectedValue}",

Авторизация.sqlcon).Fill(ds);

Авторизация.sqlcon.Close();

updDB();

}

private void button6\_Click(object sender, EventArgs e)

{

if (!(comboBox5.Text.Length > 0))

{

MessageBox.Show("Выберите животное для удаления!");

return;

}

Авторизация.sqlcon.Open();

ds = new DataSet();

new SqlDataAdapter(

$@"UPDATE Вальер SET [Количество\_животных\_внутри] = [Количество\_животных\_внутри] - 1

WHERE id = {getValierNum()}",

Авторизация.sqlcon).Fill(ds);

ds = new DataSet();

new SqlDataAdapter($"DELETE FROM Животное where id = {comboBox5.SelectedValue}", Авторизация.sqlcon).Fill(ds);

Авторизация.sqlcon.Close();

MessageBox.Show("Животное было успешно удалено из Базы Данных!");

updDB();

}

private int getValierNum()

{

SqlDataReader r = new SqlCommand($"SELECT id\_вольера FROM Животное where id = {comboBox5.SelectedValue}", Авторизация.sqlcon).ExecuteReader();

r.Read();

int id = (int)r.GetValue(0);

r.Close();

return id;

}

}

}

**ПРИЛОЖЕНИЕ 2**

## **РУКОВОДСТВО ОПЕРАТОРА**

1. **Назначение программы**
   1. **Функциональное назначение программы**

Программа предназначена для автоматизации внутренней работы зоопарка. Программа позволяет просматривать историю расхода/прихода корма на складе, вести учет по количеству животных в вольерах, добавление новых видов корма.

* 1. **Эксплуатационное назначение программы**

Данное приложение будет эксплуатировано сотрудниками зоопарка. Приложение и база данных предназначена строго для использования зоопарком и не предназначено для домашнего использования.

* 1. **Состав функций**

Приложение для сотрудников зоопарка выполняет такие функции как:

* Получение необходимой информации из базы данных;
* Редактирование, удаление и добавление данных, хранимых в базе данных, посредством интерфейса приложения;
* Авторизация в системе и выдача прав в соответствии с должностью сотрудника на предприятии.

1. **Условия выполнения программы**
   1. **Минимальный состав аппаратных средств**

Для приложения требуется компьютер с установленным на нём ОС Windows XP и выше. Требуется манипулятор типа мышь, экран и клавиатура. Технические требования совпадают с требованиями к эксплуатации Windows-приложений общего назначения. Для обеспечения необходимой производительности минимальными характеристиками являются:

 процессор Intel/AMD с частотой не менее 1ГГц;

 оперативная память – 1000 Мб;

 свободное дисковое пространство – 1000 Мб.

* 1. **Минимальный состав программных средств**

Программа должна работать на устройстве с операционной системой Windows XP и выше. Кроме того, дополнительно должны быть установлены программные компоненты для .NET Framework. Для хранения данных нужно использовать СУБД SQL Server 2019 года выпуска.

* 1. **Требования к персоналу**

Конечный пользователь должен иметь автоматизированную систему управления.

Пользователь программы должен обладать практическими навыки работы с пользовательским интерфейсом операционной системы.

1. **Выполнение программы**
   1. **Загрузка и запуск программы**

После того как пользователь запустил приложение открывается главное окно «Авторизация» (рис. 1). Оно включает в себя: текстовые поля для ввода логина и пароля, а также кнопку авторизации.

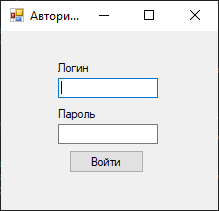


Рис. 1

* 1. **Вход в приложение с правами доступа «Директор»**
     1. Для того, чтобы авторизоваться с правами доступа директор, сотруднику необходимо ввести личный логин и пароль и нажать на кнопку «Войти».

Рис. 2

* + 1. После того, как авторизация пройдена, открывается форма «Панель Директора», в которой директор может увидеть всех зарегистрированных сотрудников в информационной системе (рис. 2).

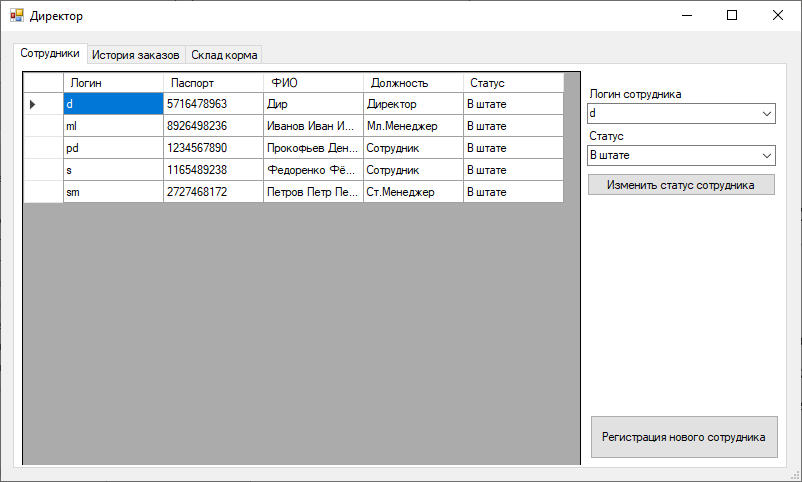


Рис. 2

Информацию об истории заказов корма на склад (рис. 3).

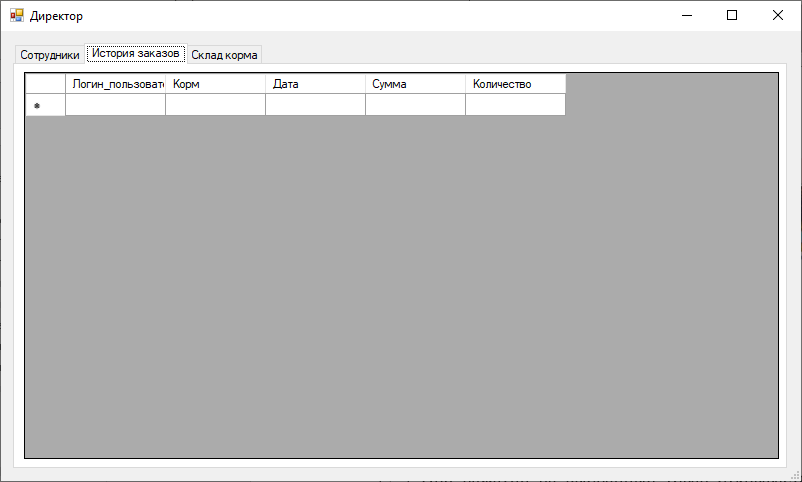


Рис. 3

А также саму информацию о корме который находится на складе (рис 4.)

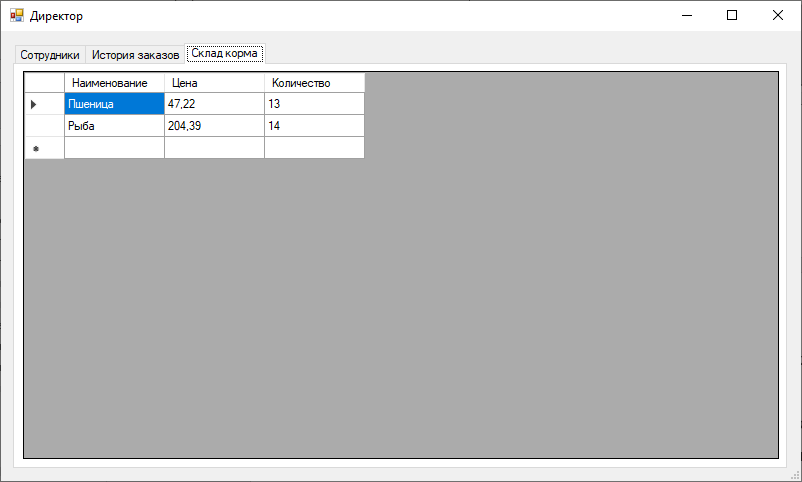


Рис. 4

* + 1. При нажатии на кнопку с регистрацией нового сотрудника открывается форма «Регистрация» (рис. 5). На данной форме директор заполняет информацию о сотруднике и проводит регистрацию по нажатию на кнопку «регистрация». После успешной регистрации форма закрывается автоматически.

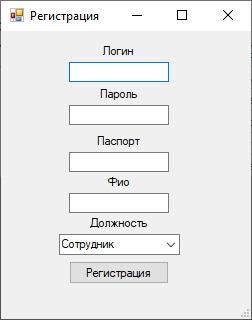


Рис. 5

* 1. **Вход в приложение с правами доступа «Сотрудник»**
     1. Чтобы выполнить вход под ролью «Сотрудник», пользователь должен ввести в окне «Авторизация» логин и пароль, который ему выдал директор, и нажать кнопку «Войти» (рис. 6).

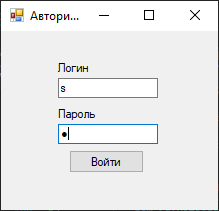


Рис. 6

* + 1. После авторизации Сотрудника откроется форма на которой Сотрудник сможет просмотреть количество корма на складе и взять из него необходимое количество для кормления животных (рис. 7).

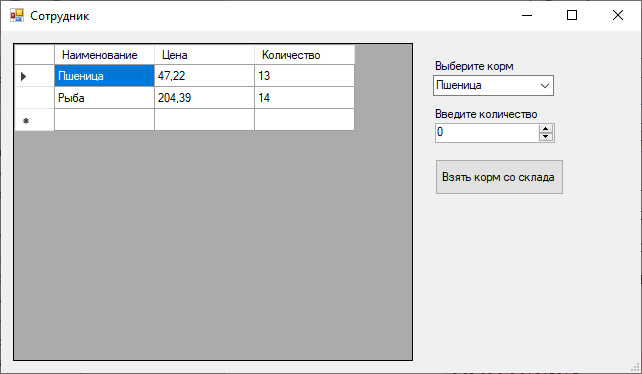


Рис. 7

**3.4** **Вход в приложение с правами доступа «Старший менеджер»**

3.4.1.Чтобы выполнить вход под ролью «Старший менеджер», пользователь должен ввести в окне «Авторизация» логин и пароль, который ему выдал директор, и нажать кнопку «Войти» (рис. 8).

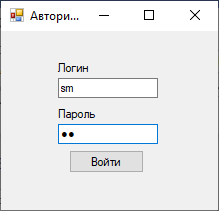


Рис. 8

3.4.2 На вкладке «Вольеры» Менеджер может просматривать все вольеры, количество животных в нём находящиеся в данный момент времени, а так же добавлять новые вольеры (рис. 9)

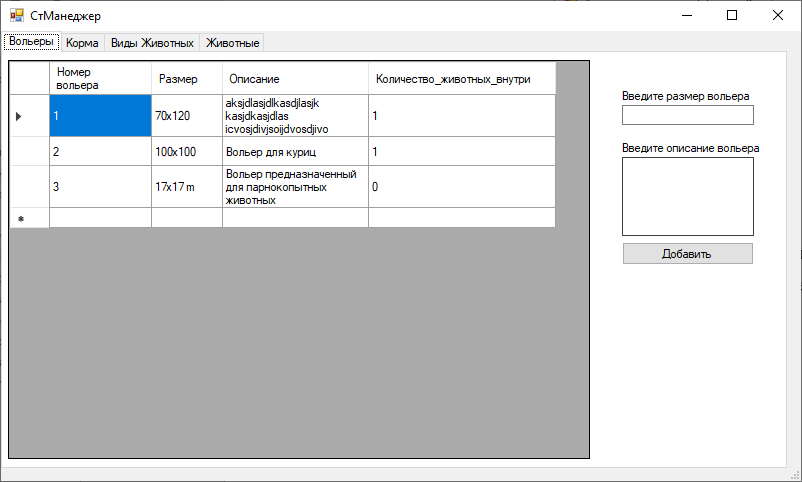


Рис. 9

3.4.3 На вкладке «Корма» менеджер может просматривать какой и какое количество корма находится сейчас на складе (рис. 10), а так же добавлять новые виды корма в информационную базу данных

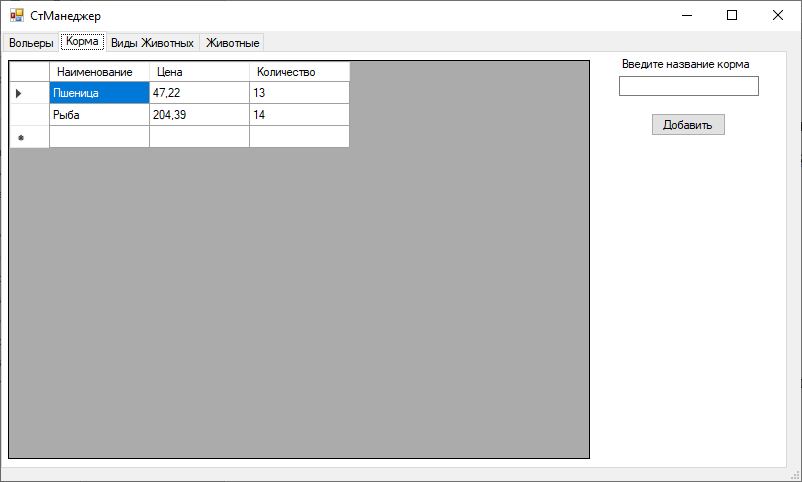


Рис. 10

3.4.4 На вкладке «Виды Животных» менеджер может просматривать и добавлять виды животных и их рацион питания (рис. 11)

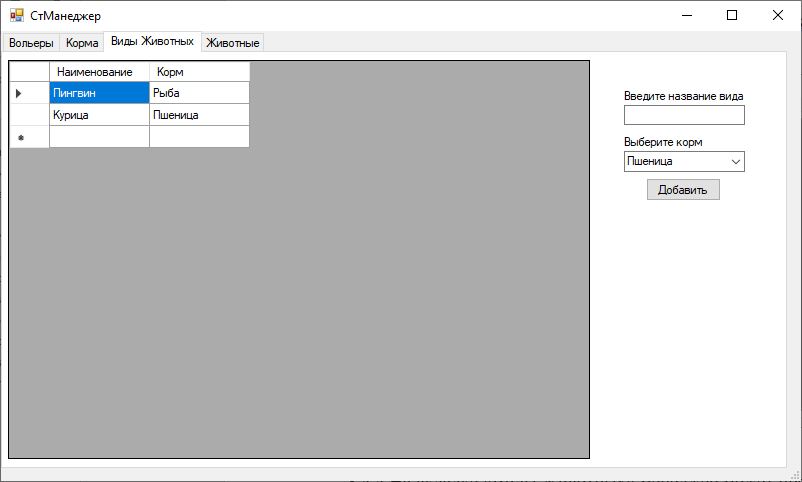


Рис. 11

3.4.5 На вкладке «Животные» менеджер может просматривать и добавлять и удалять непосредственно самих животных, давая им кличку, номер вольера и его вид (рис.12)

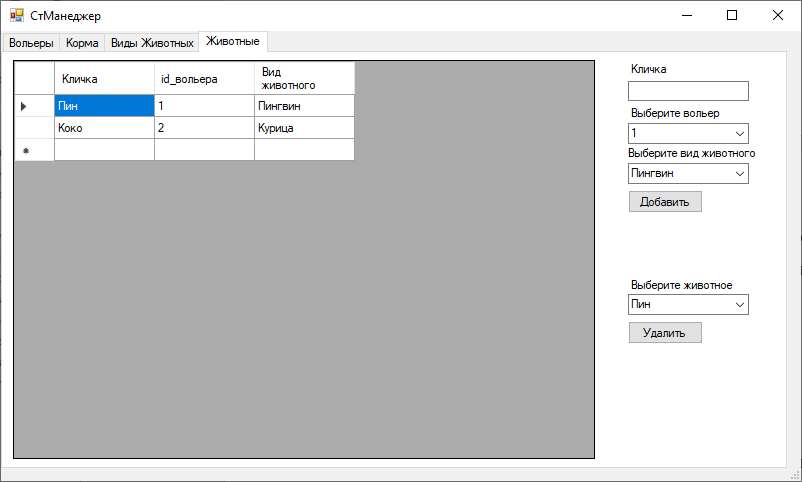


Рис. 12

3.5.1 Чтобы выполнить вход под ролью «Младший менеджер», пользователь должен ввести в окне «Авторизация» логин и пароль, который ему выдал директор, и нажать кнопку «Войти» (рис. 13)

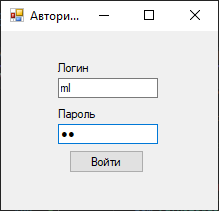


Рис. 13

3.5.2 На вкладке «Склад» менеджер может просматривать количество корма на складе, изменять цену на корм и заказывать его (рис. 14)

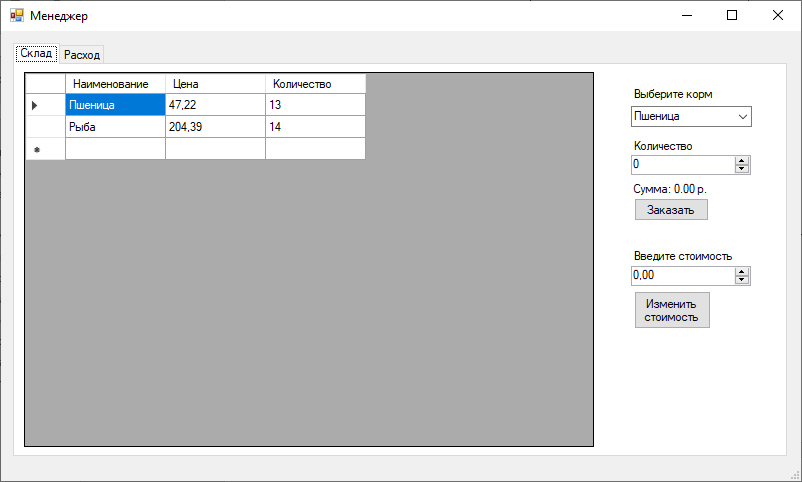


Рис. 14

3.5.3 На вкладке «Расход» менеджер может смотреть какой сотрудник, в какое время, какой корм, какое количество и на какую сумму взял со склада сотрудник (рис. 15)

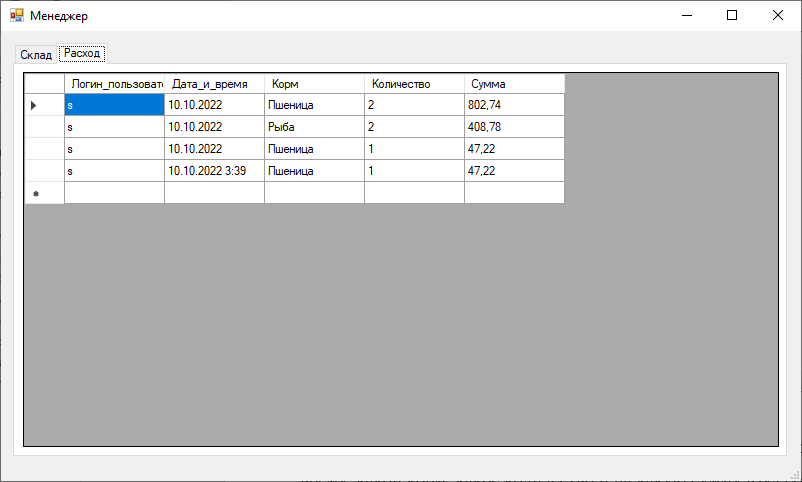


Рис. 15

**3.6 Завершение работы программы**

Чтобы завершить работу программы необходимо закрыть приложение путём нажатия на крест в правом верхнем углу приложения (рис. 16)

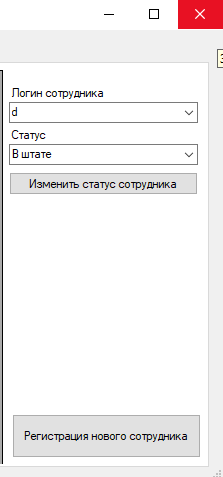


Рис. 16

1. **Сообщения оператору**

**4.1 Сообщения**

Приветствие пользователя при входе (рис. 17)

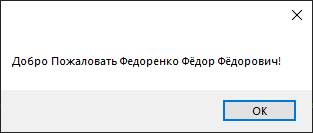


Рис. 17

Система сообщает об успешном взятии корма со склада (рис. 18)

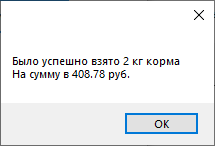


Рис. 18

Система сообщает об успешном изменении статуса у сотрудника (рис. 19)

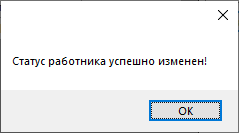


Рис. 19

Система сообщает об успешной записи нового сотрудника в базу данных (рис.20)

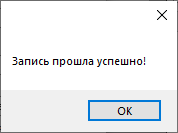


Рис. 20

Система сообщает об успешном изменении цены на корм (рис. 21)

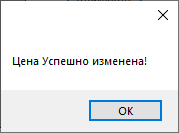


Рис. 21

Система сообщает об успешном заказе корма (рис. 22)

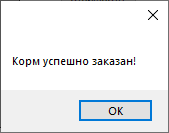


Рис. 22

Система сообщает об успешном взятии корма со склада (рис. 23)

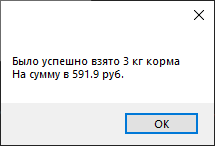


Рис. 23