

**Колледж космического машиностроения и технологий**

**КУРСОВОЙ ПРОЕКТ**

по дисциплине МДК.02.02. Технология разработки и защиты баз данных

**РАЗРАБОТКА СИСТЕМЫ контроля и управления крупным сельскохозяйственным предприятием**

Пояснительная записка

КП.09.02.03.23.15ПЗ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Обучающийся группы | | П1-19 | Погуляев Ф.В. | |
| Руководитель курсового проекта | | | Стрельников С.Д. | |
| Результат защиты |  | | |  |

г. Королёв, 2023 г.

**ОГЛАВЛЕНИЕ**

[ВВЕДЕНИЕ 4](#_Toc131123239)

[1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 5](#_Toc131123240)

[1.1. Описание предметной области 5](#_Toc131123241)

[1.2. Описание существующей разработки 5](#_Toc131123242)

[2. ПРОЕКТНАЯ ЧАСТЬ 7](#_Toc131123243)

[2.1. Обоснование инструментов разработки 7](#_Toc131123244)

[2.2. Проектирование сценария работы проекта 8](#_Toc131123245)

[2.3. Разработка диаграммы классов 9](#_Toc131123246)

[2.4. Разработка ER-диаграммы баз данных 10](#_Toc131123247)

[2.5. Описание модулей проекта 11](#_Toc131123248)

[2.6. Тестирование проекта 11](#_Toc131123249)

[2.7. Руководство оператора 12](#_Toc131123250)

[ЗАКЛЮЧЕНИЕ 22](#_Toc131123251)

[Список используемых источников 23](#_Toc131123252)

[Приложение 1. 24](#_Toc131123253)

# **ВВЕДЕНИЕ**

**РАЗРАБОТКА СИСТЕМЫ КОНТРОЛЯ И УПРАВЛЕНИЯ КРУПНЫМ СЕЛЬСКОХОЗЯЙСТВЕННЫМ ПРЕДПРИЯТИЕМ**

Тема курсового проекта – «Разработка системы контроля и управления крупным сельскохозяйственным предприятием

Актуальность данного курсового проекта заключается в повышении качества контроля и удобства доступа к данным о животных, что повышает общую скорость реагирования различных систем предприятия.

Целью данного курсового проекта является разработка клиентской части десктопного приложения, которое будет взаимодействовать с базой данных.

В первой части будет рассмотрена предметная область и существующие продукты по данной теме.

Во второй части будут описаны инструменты и модули, которые были разработаны и описано руководство оператора.

В заключительной части будет и заключение по курсовому проекту.

# **ТЕОРЕТИЧЕСКАЯ ЧАСТЬ**

## **Описание предметной области**

Целью данной работы является разработка десктопного приложения для электронного учета сельскохозяйственного скота, находящегося на балансе предприятия, а также ведения учета прививок. В данной ситуации предлагается следующее решение поставленной задачи:

Десктопное приложение в функционал которого входит:

* Возможность редактирования данных каждого животного;
* Управление базой данных через визуальный интерфейс;
* Фильтр, определяющий статус вакцинации животного;
* Составление отчета в .xls таблице.

## **Описание существующей разработки**

В этом разделе рассмотрены некоторые уже имеющиеся программы по данной тематике.

Практически все из них предоставляют похожий функционал:

* Управление базой данных;
* Запись заказов клиентов;

Примеры программ:

Один из примеров системы контроля и управления крупным сельскохозяйственным предприятием - это 1С:Предприятие 8. Управление сельскохозяйственным предприятием. Эта система разработана на базе 1С:Предприятие 8. Управление производственным предприятием и включает в себя помимо отраслевого функционала все возможности базовой конфигурации (Рисунок 1).

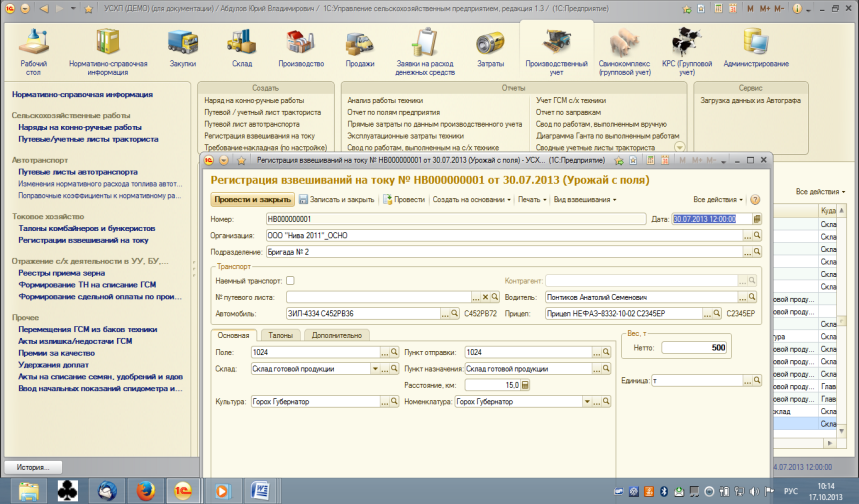


Рис.1 Интерфейс 1С: Предприятие 8. Управление сельскохозяйственным предприятием

Другой пример - это Система Soft.Farm. Это комплексное IT-решение для агропроизводителей, которое объединяет данные из других систем в единый формат и создает прозрачную аналитическую систему сельскохозяйственной деятельности для принятия взвешенных управленческих решений.

Оба этих приложения имеют достаточно высокую стоимость подписки, перегруженный интерфейс и проблемы с настройкой и установкой.

# **ПРОЕКТНАЯ ЧАСТЬ**

## **Обоснование инструментов разработки**

Для создания проекта, основным инструментом был выбран PyCharm: the Python IDE, потому что он является одним из лучших инструментов для разработки на Python. В качестве СУБД был выбран SQLite [1].

Язык Python был выбран по следующим факторам:

* Понятность кода – простое зрительное восприятие программ, написанных на этом языке.
* Наличие мощного фреймворка для разработки клиентских приложений;
* Наличие подробной документации.

Сравнение Python с другими языками программирования представлено в таблице 1.

Таблица 1. сравнения языков программирования

|  |  |  |  |
| --- | --- | --- | --- |
| **Критерий** | **C++** | **С#** | **Python** |
| Читабельность | 5 | 7 | 8 |
| Простота | 4 | 6 | 8 |
| GUI | 9 | 8 | 7 |
| Скорость разработки | 5 | 6 | 8 |
| Знание языка | 5 | 3 | 8 |
| Итого баллов | 28 | 30 | 39 |

Причины, по которым был выбран данный инструмент и СУБД:

**PyCharm**

PyCharm – это среда разработки, разрабатываемая компанией JetBrains для разработки приложений на языках, работающих с JVM. Представляет собой комплекс интегрированных между собой инструментов программирования. Включает в себя:

* Редактор кода;
* Инструменты рефакторинга кода;
* Инструменты разработки Android-Приложений;
* Интеграцию с системой контроля версий Git.

Intellij IDEA делает разработку максимально продуктивной благодаря функциям автодополнения и анализа кода, мгновенной подсветке ошибок и быстрым исправлениям. Автоматические рефакторинги помогают эффективно редактировать код, а удобная навигация позволяет мгновенно перемещаться по проекту.

Возможности форматирования и настройки стиля кода помогают писать понятный код, который легко поддерживать.

**SQLite**

SQLite - это библиотека, которая реализует маленький, быстрый, самодостаточный, высоконадежный, полнофункциональный движок базы данных SQL. SQLite является самым используемым движком базы данных в мире. Он встроен во все мобильные телефоны и большинство компьютеров и поставляется в комплекте со многими другими приложениями, которые люди используют каждый день.

Один из основных преимуществ SQLite — это его компактность и самодостаточность. Он не требует отдельного сервера или конфигурации и может быть легко встроен в другие приложения. Это делает его идеальным для использования в мобильных приложениях или в других сценариях, где требуется легковесная база данных.

Однако у SQLite есть и некоторые ограничения. Например, он не поддерживает многопользовательский доступ к данным на уровне сервера. Это означает, что, если несколько пользователей пытаются одновременно изменить данные в базе данных SQLite, может возникнуть конфликт.

## **Проектирование сценария работы проекта**

В данном разделе приведен сценарий использования программы пользователем приложения (Рисунок 2).

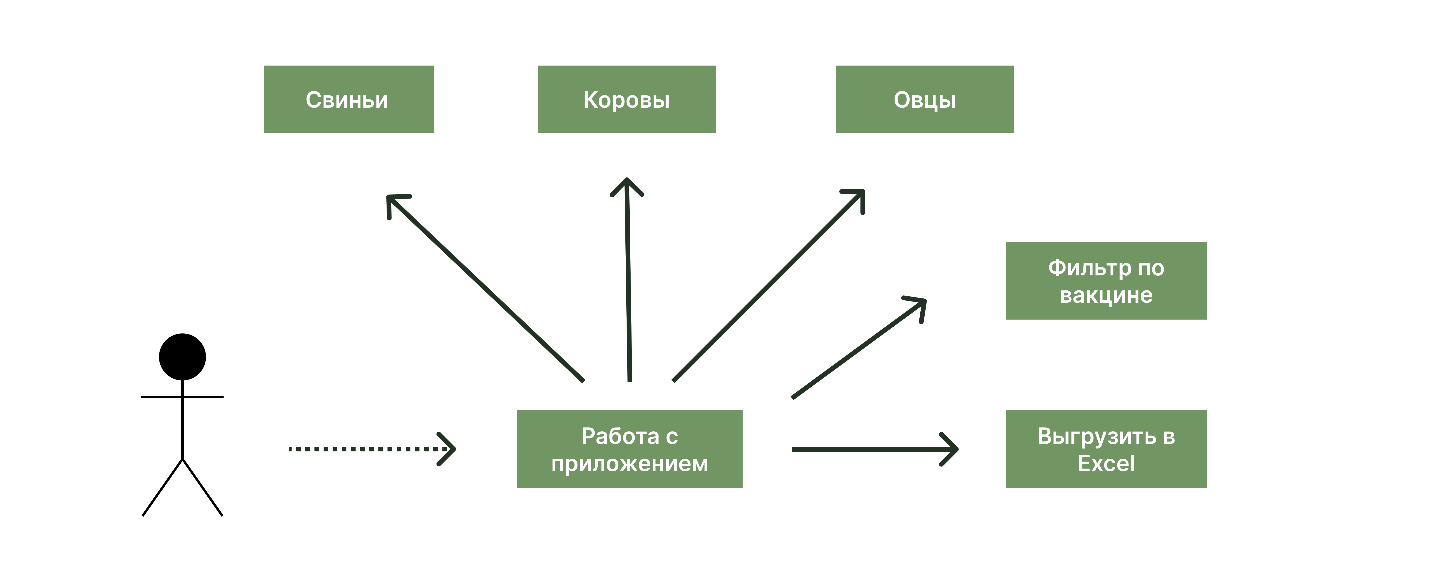


Рисунок 2. Сценарий использования

Пользователь после запуска приложения может выполнить следующие действия:

1. Управление имеющимся рогатым скотом;
2. Формирование отчета в Excel;
3. Добавление/удаление/редактирование данных в таблицах;
4. Отслеживание вакцинированного и не вакцинированного скота.

При выходе из программы, она завершает свою работу.

## **Разработка диаграммы классов**

В данном разделе представлены все классы, использующиеся в проекте, а также их отношения между собой (Рисунок 3).

1. SystemCRM: класс содержит набор функций, импортируемых модулей, реализующих логику приложения, взаимодействие с БД;
2. UiInterface: класс инициализирует виджеты и GUI приложения, производит их первоначальную настройку.
3. Database: класс инициализирует базу данных, если она была не создана, наполняет ее шаблонными данными, содержит набор функцию для взаимодействия с БД.

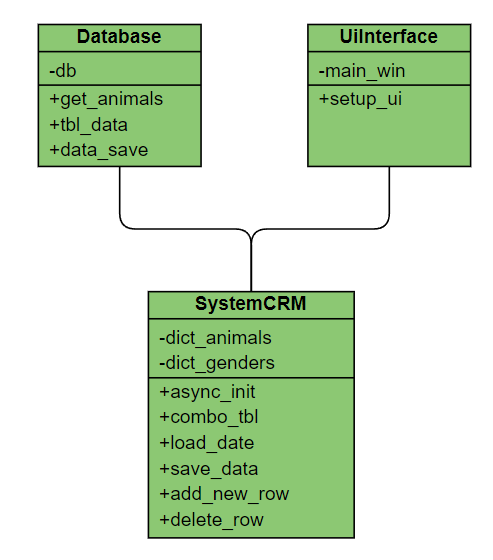


Рисунок 3. Диаграмма классов

## **Разработка ER-диаграммы баз данных**

В приложении для хранения всей нужной информации была создана база данных. На рисунке 4 представлена ER-модель Базы данных. Так же ниже описана структура таблиц и назначение полей.

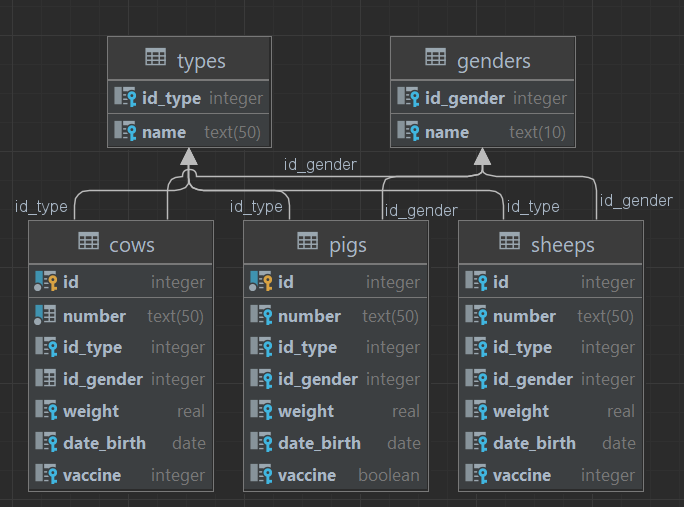


Рисунок 4. ER-модель

База данных состоит из 5 таблиц:

1. Types – таблица для типа скота;
2. Genres – таблица хранит наименование пола;
3. Cows – таблица для хранения данных о коровах;
4. Pigs – таблица для хранения данных о свиньях;
5. Sheeps - таблица для хранения данных об овцах;

## **Описание модулей проекта**

Помимо стартового модуля, в проект входят и другие модули с фрагментами и интерфейсные модули.

Также в стартовый модуль входит код, который запускает саму программу при ее запуске. Полный код всех модулей находится в «Приложение 1».

async\_init() – данный метод запустится при старте программы, как входная точка в асинхронность;

load\_data() – этот метод вызывается для загрузки данных из БД в определенную таблицу;

add\_new\_row() – данный метод вызывается для добавления новой строки в определенную таблицу;

delete\_row() – данный метод вызывается для удаление строки в определенной таблице;

save\_data() – данный метод берёт сохраняет данные редактируемой таблицы.

## **Тестирование проекта**

В этом разделе будут продемонстрированы результаты тестирования программы.

**Тест. Отладка**

В этом разделе показано умение применять средства отладки. Так как я работаю в PyCharm, отладчик уже встроен в программу.

Для проверки была искусственно создана ошибка в коде. Вот что показал отладчик (Рисунок 5, 6).

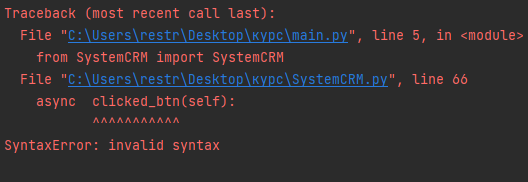


Рисунок 5. Ошибка в коде.

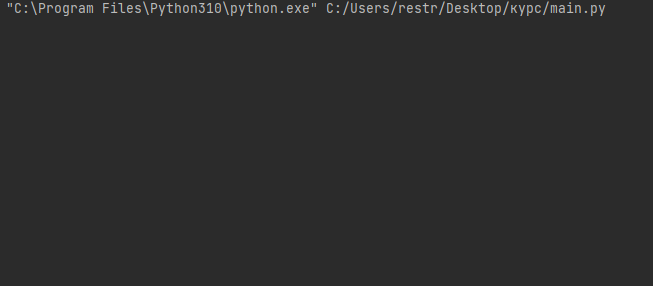


Рисунок 6. Исправление ошибки.

## **Руководство оператора**

В данном программном документе приведено руководство оператора по применению и эксплуатации программы «Farm».

В данном программном документе, в разделе «Назначение программы» указаны сведения о назначении программы и информация, достаточная для понимания функций программы и ее эксплуатации.

В разделе «Условия выполнения программы» указаны условия, необходимые для выполнения программы (минимальный состав аппаратных и программных средств и т.п.).

В данном программном документе, в разделе «Выполнение программы» указана последовательность действий оператора, обеспечивающих загрузку, запуск, выполнение и завершение программы, приведено описание функций, формата и возможных вариантов команд, с помощью которых оператор осуществляет загрузку и управляет выполнением программы, а также ответы программы на эти команды.

Оформление программного документа «Руководство оператора» произведено по требованиям ЕСПД (ГОСТ 19.101-77 [[1]](#footnote-1)1), ГОСТ 19.103-77 [[2]](#footnote-2)2), ГОСТ 19.104-78\* [[3]](#footnote-3)3), ГОСТ 19.105-78\* [[4]](#footnote-4)4), ГОСТ 19.106-78\* [[5]](#footnote-5)5), ГОСТ 19.505-79\* [[6]](#footnote-6)6), ГОСТ 19.604-78\* [[7]](#footnote-7)7)).

Далее на страницах 15-22 приведено содержание разработанного документа «Руководство оператора» на спроектированное приложение.

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

Специальное программное обеспечение «Farm» используется для изучения специализированного технического IT английского.

* 1. **Эксплуатационное назначение программы**

Специальное программное обеспечение «Farm» может эксплуатироваться в любых сельскохозяйственных предприятиях, обладающих любым поголовьем скота.

* 1. **Состав функций**
     1. **Функция регистрации.**

Эта функция позволяет зарегистрироваться новому пользователю.

* + 1. **Функция авторизации.**

Эта функция позволяет пользователю авторизоваться.

* + 1. **Функция просмотра статей.**

Эта функция позволяет просмотреть определённые статьи в приложении.

* + 1. **Функция тестирования.**

Эта функция позволяет проходить тестирование по словам из определённых статей в приложении.

* + 1. **Функция правописания слов.**

Эта функция позволяет упражняться путем правописания определённых слов из статей в приложении.

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

* ОС: Windows 10 и выше;
* Процессор: 32-битные и 64-битные: ARM, MIPS, x86. Тактовая частота не ниже 1,8 ГГц;
* ОЗУ: 2 ГБ;
* Место на жестком диске: 10 МБ;
* Видеоадаптер: DirectX версии не ниже 9 с драйвером WDDM 1.0.;
* Дисплей: 720 x 1600.
  1. **Минимальный состав программных средств**

Дополнительные программные средства не требуются.

* 1. **Требование к пользователю**

Конечный пользователь программы должен обладать практическими навыками работы с графическим пользовательским интерфейсом операционной системы.

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

Для запуска программы требуется запустить ярлык исполняемого файла.

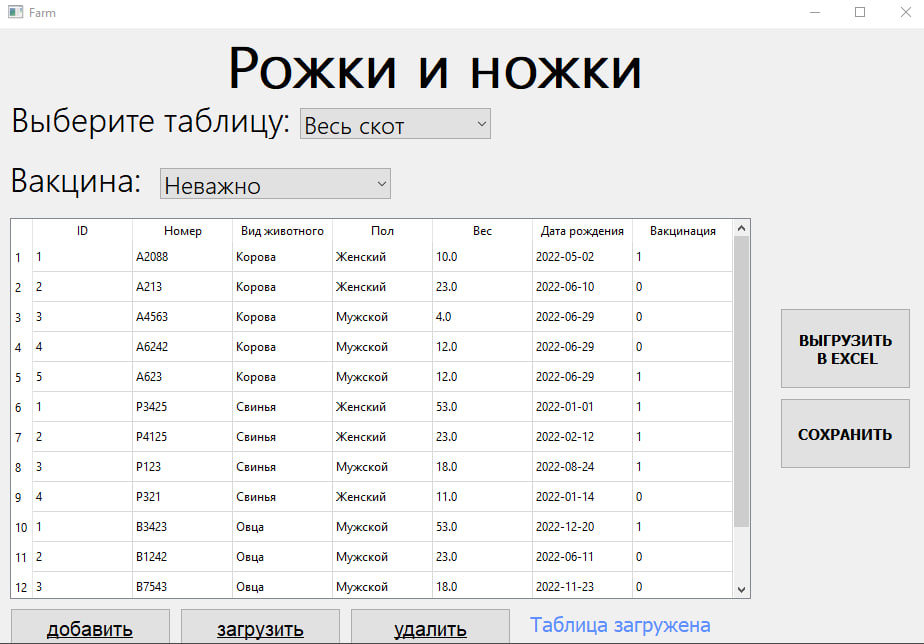


Рис. 7. Основной интерфейс приложения

На рис. 7 отображен основной интерфейс приложения.

Представлены следующие элементы:

1. Фильтр вида скота
2. Фильтр вакцинации
3. Кнопка добавления строк
4. Кнопка загрузки редактированных данных в таблицу.
5. Кнопка удаления выбранных строк.
6. Кнопка выгрузки данных в .xls таблицу.
7. Кнопка сохранения изменений.

Редактирование данных происходит через двойной клик по необходимому полю (Рисунок 8).

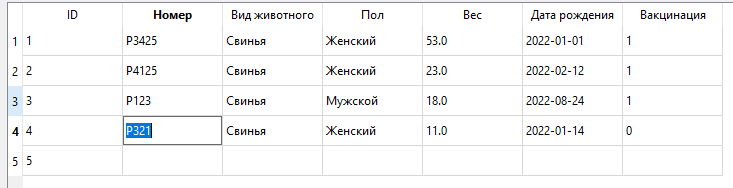


Рис. 8. Редактирование данных

Настройка фильтра животных совершается в выпадающем окне, вызываемом через одиночный клик по указанному полю (Рисунок 9).

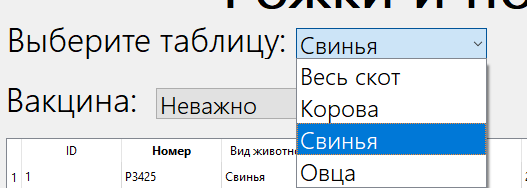


Рис. 9. Фильтр вида животных

Фильтр вакцинации настраивается в выпадающем окне, вызываемом через одиночный клик по указанному полю (Рисунок 10).

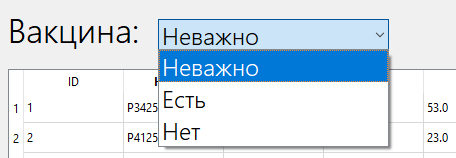


Рис.10. Фильтр вакцинации

Для добавления строки необходимо выбрать необходимый вид скота и нажать на кнопку «добавить» (Рисунок 12).

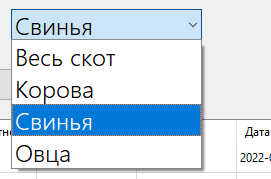


Рис. 11. Выбор вида животного

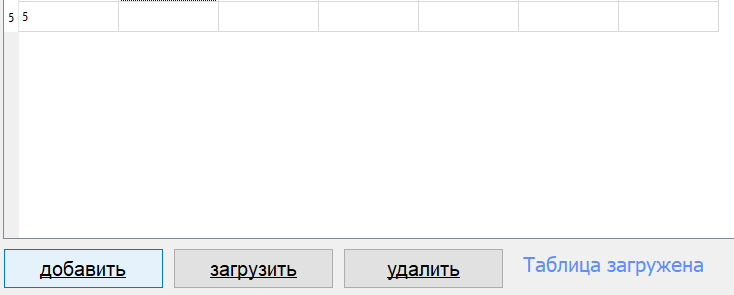


Рис. 12. Добавление строки

Для загрузки новой строки в базу данных необходимо нажать кнопку «загрузить» (Рисунок 13).

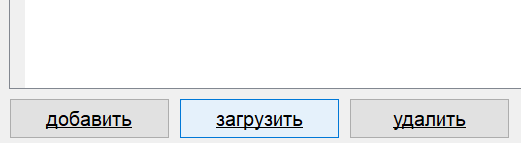


Рис. 13. Загрузка новой строки в таблицу

Для удаления строки необходимо выбрать строку и нажать кнопку «удалить». (Рисунок 14)

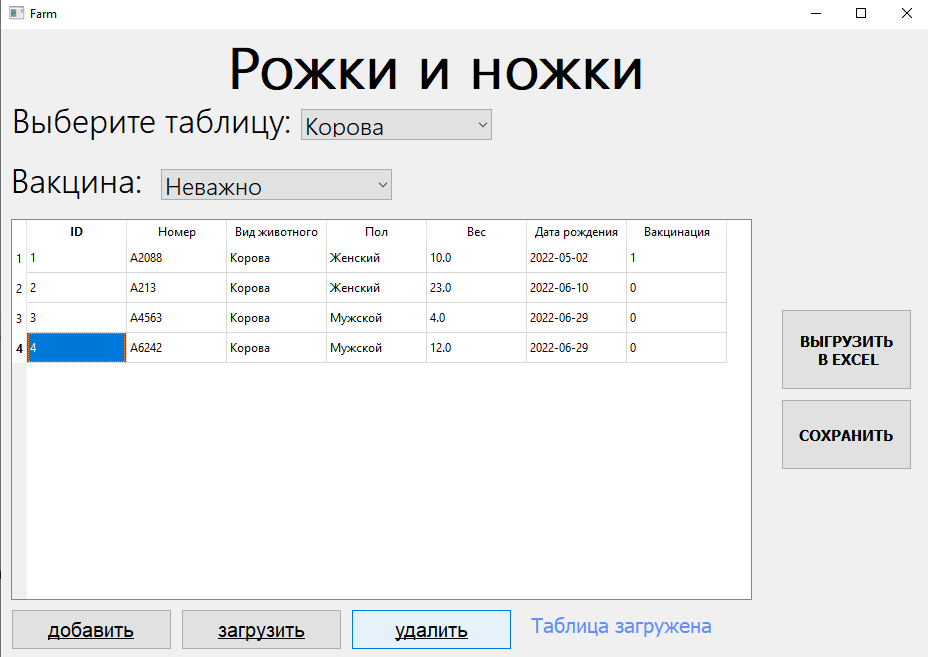


Рис. 14. Удаление строки

Для выгрузки данных в .xls таблицу необходимо нажать кнопку «Выгрузить в Excel» (Рисунок 15).

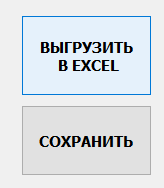


Рис.15. Выгрузка данных

Для сохранения измененных данных необходимо нажать кнопку «Сохранить» (Рисунок 16).

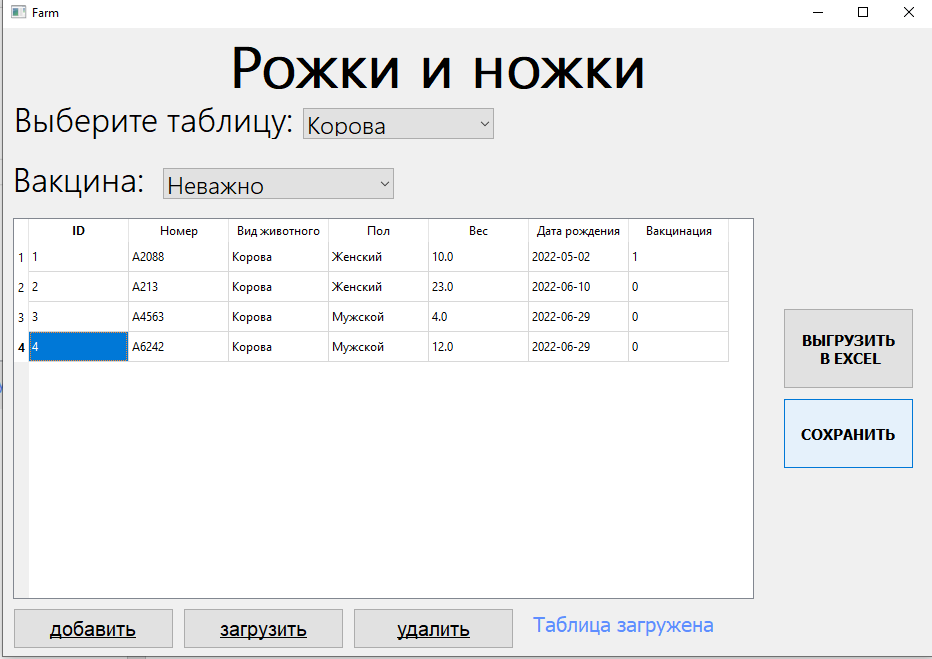


Рис. 16. Сохранение данных

Статус операции и возможные ошибки отображаются в левом нижнем углу интерфейса (Рисунок 17).

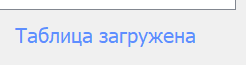


Рис. 17. Статус операции

Выгруженные .xls таблицы сохраняются в директории программы в папке «ExcelData» (Рисунок 18).

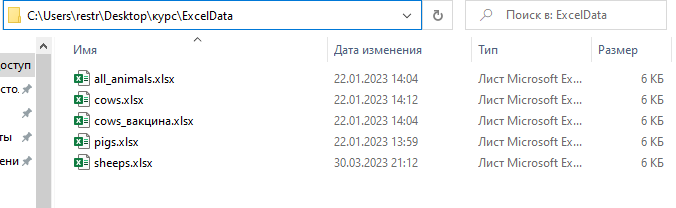


Рис. 18. Директория выгруженных таблиц

* 1. **Завершение работы программы**

Чтобы завершить работу программы нужно принудительно завершить процесс через стандартные элементы интерфейса.

# **ЗАКЛЮЧЕНИЕ**

В результате выполнения курсового проекта была написана программа «Farm» для упрощения и удобства контроля и редактирования данных сельскохозяйственного предприятия.

В ходе работы были проанализированы: предметная область, существующие разработки, получены навыки по созданию десктопных приложений и работе с БД.

Также планируется продолжение работы над данным проектом с целью расширения возможностей и удобства приложения для пользователей.

# **Список используемых источников**

1. “Beginning PyQt: A Hands-on Approach to GUI Programming with PyQt6” Автор: Joshua Willman. Издательство: Apress. Дата публикации: 2-е издание (17 марта 2022 года)
2. “Python and Qt: The Best Parts” Автор: Michael Herrmann. Обновлено: Март 2022 года
3. “Create GUI Applications with Python & Qt6” Автор: Martin Fitzpatrick. Дата публикации: Март 2021 года
4. “Beginning PyQt: A Hands-on Approach to GUI Programming” Автор: Joshua Willman. Издательство: Apress. Дата публикации: 1-е издание (29 мая 2020 года)
5. “Modern PyQt: Create GUI Applications for Project Management, Computer Vision, and Data Analysis” Автор: Joshua Willman. Издательство: Apress. Дата публикации: 1-е издание (9 декабря 2020 года)

**Приложение 1.**

**Листинг 1. «main.py»**

import asyncio

import sys

from PyQt5 import QtWidgets

from asyncqt import QEventLoop

from SystemCRM import SystemCRM

if \_\_name\_\_ == '\_\_main\_\_':

app = QtWidgets.QApplication(sys.argv)

loop = QEventLoop(app)

asyncio.set\_event\_loop(loop)

system\_crm = SystemCRM()

with loop:

sys.exit(loop.run\_forever())

**Листинг 2. « SystemCRM.py»**

import SQLiteData

from GUI.UiInterface import UiInterface

import pandas as pd

from PyQt5 import QtWidgets

from PyQt5.QtWidgets import QMainWindow

from asyncqt import asyncSlot

def singleton(ui\_class):

instances = {}

def getinstance(\*args, \*\*kwargs):

if ui\_class not in instances:

instances[ui\_class] = ui\_class(\*args, \*\*kwargs)

return instances[ui\_class]

return getinstance

@singleton

class SystemCRM(QMainWindow):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.query\_data = None

self.tbl\_name = None

self.vaccine = None

self.get\_tbl = None

self.type\_btn = None

self.suppliers\_dict = {}

self.ui = UiInterface(self)

self.db = SQLiteData.ex

self.new\_row\_warning = True

self.delete\_all\_warning = False

self.async\_init()

self.dict\_animals = {"Корова": [1, "cows"], "Свинья": [2, "pigs"],

"Овца": [3, "sheeps"], "Весь скот": [4, "all\_animals"]}

self.dict\_genders = {"Женский": 1, "Мужской": 2}

self.show()

@asyncSlot()

async def async\_init(self):

all\_animals = await self.db.get\_animals()

self.ui.main\_win.combo\_tbls.addItems(all\_animals)

self.ui.main\_win.combo\_tbls.currentIndexChanged.connect(self.combo\_tbl)

await self.combo\_tbl()

self.ui.main\_win.combo\_tbls2.currentIndexChanged.connect(self.combo\_tbl)

await self.clicked\_btn()

@asyncSlot()

async def combo\_tbl(self):

self.get\_tbl = self.ui.main\_win.combo\_tbls.currentText()

self.vaccine = self.ui.main\_win.combo\_tbls2.currentText()

self.tbl\_name = self.dict\_animals[self.get\_tbl][1]

self.query\_data = []

if self.vaccine == "Неважно" and self.get\_tbl == "Весь скот":

self.query\_data = [0, 1]

elif self.vaccine == "Есть":

self.query\_data = [1, '']

elif self.vaccine == "Нет":

self.query\_data = [0, '']

else:

self.query\_data = [0, 1]

await self.load\_date(self.ui.main\_win.main\_tbl)

async def clicked\_btn(self):

win = self.ui.main\_win

win.btn\_load\_orders.clicked.connect(lambda: self.load\_date(self.ui.main\_win.main\_tbl))

win.btn\_add\_orders.clicked.connect(lambda: self.add\_new\_row(self.ui.main\_win.main\_tbl))

win.btn\_del\_orders.clicked.connect(lambda: self.delete\_row(self.ui.main\_win.main\_tbl))

win.btn\_save\_orders.clicked.connect(lambda: self.save\_data(self.ui.main\_win.main\_tbl))

win.btn\_xls\_orders.clicked.connect(lambda: self.load\_excel(self.ui.main\_win.main\_tbl))

@asyncSlot()

async def load\_date(self, tbl):

tbl.setRowCount(0)

data = await self.db.tbl\_data(self.get\_tbl, self.query\_data)

for row\_number, row\_data in enumerate(data):

tbl.insertRow(row\_number)

for col\_number, col\_data in enumerate(row\_data):

tbl.setItem(row\_number, col\_number, QtWidgets.QTableWidgetItem(str(col\_data)))

self.new\_row\_warning = True

self.ui.main\_win.lbl\_info\_tbl.setText('Таблица загружена')

self.delete\_all\_warning = False

@asyncSlot()

async def save\_data(self, tbl):

self.ui.main\_win.lbl\_info\_tbl.setText('')

try:

data = []

for row in range(tbl.rowCount()):

data.append([])

if not tbl.item(row, 0).text().isdigit():

raise Exception

for col in range(tbl.columnCount()):

if col == 2:

type\_animal = self.dict\_animals[tbl.item(row, col).text()][0]

data[row].append(type\_animal)

elif col == 3:

gender = self.dict\_genders[tbl.item(row, col).text()]

data[row].append(gender)

else:

data[row].append(tbl.item(row, col).text())

self.new\_row\_warning = True

await self.db.data\_save(data, self.tbl\_name)

self.ui.main\_win.lbl\_info\_tbl.setText('Таблица сохранена')

except AttributeError:

self.ui.main\_win.lbl\_info\_tbl.setText('Введите все поял корректно')

def add\_new\_row(self, tbl):

if self.get\_tbl != "Весь скот" and self.vaccine == "Неважно":

if self.new\_row\_warning:

row\_position = tbl.rowCount()

new\_id = str(int(tbl.item(row\_position - 1, 0).text()) + 1)

tbl.insertRow(row\_position)

tbl.setItem(row\_position, 0, QtWidgets.QTableWidgetItem(new\_id))

self.new\_row\_warning = False

else:

self.ui.main\_win.lbl\_info\_tbl.setText('Сохраните таблицу')

else:

self.ui.main\_win.lbl\_info\_tbl.setText('Выберите таблицу скота без фильтров')

def delete\_row(self, tbl):

if self.get\_tbl != "Весь скот" and self.vaccine == "Неважно":

if tbl.rowCount() > 0 and tbl.currentRow() != -1:

current\_row = tbl.currentRow()

tbl.removeRow(current\_row)

else:

self.ui.main\_win.lbl\_info\_tbl.setText('Выберите таблицу скота без фильтров')

def load\_excel(self, tbl):

k = 0

data = [[tbl.item(row, column).text() for row in range(tbl.rowCount())] for column in range(tbl.columnCount())]

dict\_xls = {"ID": [], "Номер": [], "Животное": "", "Пол": [],

"Вес": [], "Дата рождения": [], "Вакцинация": []}

for key in dict\_xls:

dict\_xls[key] = data[k]

k += 1

data\_frame = pd.DataFrame(dict\_xls)

if self.vaccine == "Есть":

data\_frame.to\_excel(f"./ExcelData/{self.tbl\_name}\_вакцина.xlsx", index=False)

self.ui.main\_win.lbl\_info\_tbl.setText(f'Данные загружены в {self.tbl\_name}\_вакцина.xlsx')

elif self.vaccine == "Нет":

data\_frame.to\_excel(f"./ExcelData/{self.tbl\_name}\_без\_вакцины.xlsx", index=False)

self.ui.main\_win.lbl\_info\_tbl.setText(f'Данные загружены в {self.tbl\_name}\_без\_вакцины.xlsx')

else:

data\_frame.to\_excel(f"./ExcelData/{self.tbl\_name}.xlsx", index=False)

self.ui.main\_win.lbl\_info\_tbl.setText(f'Данные загружены в {self.tbl\_name}.xlsx')

**Листинг 3. «Database.py»**

import aiosqlite

class Database:

def \_\_init\_\_(self):

self.db = "./SQLiteData/Database.db"

async def get\_animals(self):

query = """ SELECT DISTINCT type

FROM all\_animals"""

async with aiosqlite.connect(self.db) as db:

db.row\_factory = lambda cursor, row: row[0]

async with db.execute(query) as cursor:

return await cursor.fetchall()

async def tbl\_data(self, combo\_text, vaccine):

match combo\_text:

case "Корова":

query = """SELECT \* FROM cows\_view WHERE vaccine IN ((?), (?))"""

case "Овца":

query = """ SELECT \* FROM sheeps\_view WHERE vaccine IN ((?), (?))"""

case "Свинья":

query = """ SELECT \* FROM pigs\_view WHERE vaccine IN ((?), (?))"""

case \_:

query = """ SELECT \* FROM all\_animals WHERE vaccine IN ((?), (?))"""

async with aiosqlite.connect(self.db) as db:

async with db.execute(query, vaccine) as cursor:

return await cursor.fetchall()

async def data\_save(self, data, name\_tbl):

query = """INSERT INTO {} (id, number, id\_type, id\_gender, weight, date\_birth, vaccine)

VALUES (?, ?, ?, ?, ?, ?, ?)""".format(name\_tbl)

async with aiosqlite.connect(self.db) as db:

async with db.cursor() as cursor:

await cursor.execute("DELETE FROM {}".format(name\_tbl))

await cursor.executemany(query, data)

count\_rows = cursor.rowcount

await db.commit()

return count\_rows

**Листинг 4. «UiInterface»**

from PyQt5.uic import loadUi

class UiInterface:

"""Загрузка ui файла"""

def \_\_init\_\_(self, main\_win):

self.main\_win = main\_win

self.setup\_ui()

def setup\_ui(self):

loadUi('./GUI/ui\_farm.ui', self.main\_win)

self.main\_win.setWindowTitle('Farm')

1. 1) ГОСТ 19.101-77 ЕСПД. Виды программ и программных документов [↑](#footnote-ref-1)
2. 2) ГОСТ 19.103-77 ЕСПД. Обозначение программ и программных документов [↑](#footnote-ref-2)
3. 3) ГОСТ 19.104-78\* ЕСПД. Основные надписи [↑](#footnote-ref-3)
4. 4) ГОСТ 19.105-78\* ЕСПД. Общие требования к программным документам [↑](#footnote-ref-4)
5. 5) ГОСТ 19.106-78\* ЕСПД. Общие требования к программным документам, выполненным печатным способом [↑](#footnote-ref-5)
6. 6) ГОСТ 19.505-79\* ЕСПД. Руководство оператора. Требования к содержанию и оформлению [↑](#footnote-ref-6)
7. 7) ГОСТ 19.604-78\* ЕСПД. Правила внесения изменений в программные документы, выполненные печатным способом [↑](#footnote-ref-7)