**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:

**Листинг 1. «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')