**ComboBox, CheckBox, RadioButton:**

from PyQt6.QtWidgets import QMainWindow, QRadioButton, QApplication, QVBoxLayout

from PyQt6 import QtCore, QtGui, QtWidgets

import MySQLdb as mdb

**Подключение модулей дизайнера, базы данных:**

from form(название формы) import \*

from db import \*

**Работа с датой:**

import datetime

today = datetime.date.today().day

**Конвертация UI файла в Python код:**

pyuic6 -x design.ui -o design.py

**Использование UI класса в приложении:**

import sys

from PyQt6.QtWidgets import QApplication, QMainWindow

from design import Ui\_MainWindow # импортируем сгенерированный класс

class MyMainWindow(QMainWindow, Ui\_MainWindow):

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

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

self.setupUi(self) # инициализируем дизайн

if \_\_name\_\_ == "\_\_main\_\_": # запуск

app = QApplication(sys.argv)

mainWindow = MyMainWindow()

mainWindow.show()

sys.exit(app.exec())

***Чекбокс:***

from PyQtó import QtCore, QtGui, QtWidgets

from PyQtó.QtWidgets import QMessageBox

import check\_skrin, qwer\_scrin

from check\_skrin import \*

from qwer\_scrin import \*

сlass Main(QtWidgets.QWidget, Ui\_Form):

def \_\_init\_\_ (self, parent=None):

super (Main, self).\_\_init\_\_(parent)

self. setupUi(self)

self.setWindowIcon(QtGui.QIcon("img/qwr-png"))

self.btClose.clicked.connect(QtWidgets.QApplication.instance).quit)

data\_qw = QwSqL().get\_stoplight # Запрашиваем элементы для флажка из БД

height = 0 # Расcтояние между флажками по горизонтали

self.checks = [] # Список флажков

for i in data\_qw: # В цикле создаем элементы флажка и присваиваем полученные значения из БД

height += 30

self.chLeft = QtWidgets.QCheckBox(parent=self.grBoxCheck)

self.chLeft.setGeometry(QtCore.QRect(20, height, 191, 21))

self.checks.append(self.chLeft)# Добавляем созданные флажки в список

self.chLeft.setText(f"{1[1])") # Устанавливаем значение флажка

self.grBoxCheck.setGeometry (QtCore.QRect(20, 70, 301, 50+height))

self.btInspect.clicked.connect(self.onClickCheck)# Вызываем функцию получения списка нажатых флажков

def onClickCheck(self):

checked = ' '.join([checkbox.text() for checkbox in self.checks if checkbox.isChecked]) # Получаем значения нажатых флажков в строку

if checked:

QwSql().set\_answer (checked) # Если нажат хотя бы один флажок, записываем ответ в БД

***Комбобокс:***

from check\_skrin import \*

import qwer\_scrin

afrom qwer\_scrin import \*

class Main(QtWidgets.QWidget, Ui.Form):

def\_\_init\_\_(self, parent=None):

super(Main, self).\_\_init\_\_(parent)

self.setupUi(self)

self.setWindowIcon(QtGui.QIcon("img/qwr.png"))

# Запрашиваем из БД список элементов и добавляем его в comboBox

data\_qw = QwSql().get\_stoplight()

for i in data\_qw:

self.comboBox.addItem(i[1])

self.comboBox.currentTextChanged.connect(lambda:self.comboChangedBackground(data\_qw))

self.btInspect.clicked.connect(self.onClickColor)

self.btClose.clicked.connect(QtWidgets.QApplication.instance) quit)

# Обработка событий элемента comboBox, при выборе элемента меняется цвет окна

def combochangedBackground (self, data\_qw):

comb = self.comboBox.currentText()

for i in data\_qw:

if comb == i[1]:

col = i[2]

self.setStyleSheet(f'QWidget {{background-color: (col}; color: #55BE8E; font: 14pt Impact;}}')

# Обработка выбранного элементы comboBox с помощью Button,

# по нажатию кнопки определяется активное значение comboBox и выбранный цвет записывается в БД

def onClickColor(self):

comb1 = self.comboBox.currentText()

QwSq().set\_answer (comb1)

***Радио:***

Чтобы распечатать кортеж на форму понадобится цикл for. В цикле создаем

RadioButton ***rad = QtWidgets. Radio Button (parent-self grBoxRadio)***. Присваиваем

полученные ранее из БД значения в RadioButton через ***setObjectName()*** и ***setText()***.

***parent=self.grBoxRadio*** указывает на то, что все кнопки созданные в цикле будут сгруппированы в один общий GroupBox, это нужно для исключения нажатия сразу двух RadioButton:

class MainWin(Qtwidgets.Qwidget, Ui\_Form):

def \_\_init\_\_(self, parent = None):

super(MainWin, self).\_\_init\_\_(parent)

self. setupUi(self)

res\_data = get\_data()

print(res\_data)

height = 10

for i in res\_data: # В цикле создаем RadioButton и дрисваиваем полученные значения из БД

rad = QtWidgets. QRadioButton(parent=self.grBoxRadio)

rad.setGeometry(20, height, 191, 21)

rad.setObjectName(f"{i[0]}")

rad.setText(f"{i[1])")

self.grBoxRadio.setGeometry(QtCore.QRect(60, 40, 231, 50+height))

Чтобы отслеживать в дальнейшем нажатие на кнопки, необходимо подключить ее

сигнал toggled к слоту ***rad.toggled.connect(self.on \_togled).***

Toggled будет отслеживать нажатие RadioButton, всякий раз при нажатии на одну из кнопок, будет происходить вызов функции (on\_toggled):

from PyQto import QtWidgets

import sys, MySQLdb as mdb

from topicRadioUi import \*

*# здесь строка подключения к бд и получение данных*

сlass MainWin(OtWidgets.QWidget, Ui\_Form):

def \_\_init\_\_(self, parent=\_None):

super (MainWin, self).\_\_init\_\_(parent)

self. setupUi(self)

res\_data = get\_data()

print(res\_data)

height = 10

for i in res\_data: # В цикле создаем RadioButton и присваиваем полученные значения из БД

rad = QtWidgets.QRadioButton(parent=self.grBoxRadio)

rad.setGeometry(20, height, 191, 21)

rad.setObjectName(f"{i[0]}")

rad.setText(f"{i[1]}")

self.grBoxRadio.setGeometry(QtCore.QRect(60, 40, 231, 50+height))

rad.toggled.connect(self.on\_togled) # Подключаем сигналы toggled к одному слоту

height += 25

В функции on\_togled объект sender() определяет, какой виджет является отправителем сигнала(RadioButton), с помощью radio.objectName() мы можем получить имя объекта, radio.text() текст объекта.

def on\_togled (self, checked):

radio = self.sender()

if checked:

print(radio.objectName())

print(radio.text())

class MyApplication(QMainWindow, Ui\_MainWindow):

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

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

self.setupUi(self) # инициализируем дизайн

self.load\_radio\_buttons()

self.pushButton.clicked.connect(self.on\_push\_button\_clicked)

*# здесь загрузка объектов из БД*

layout = QVBoxLayout()

for name, in c.fetchall():

radio\_button = QRadioButton(name)

layout.addWidget(radio\_button)

self.groupBox.setLayout(layout)

conn.close()

def on\_push\_button\_clicked(self): # обработчик нажатия кнопки

value = self.lineEdit.text()

self.call\_stored\_procedure(value)

def call\_stored\_procedure(self, value):

# код для вызова хп

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

app = QApplication(sys.argv)

window = MyApplication()

window.show()

sys.exit(app.exec())

**Checkbox:**

from PyQt6 import QtCore, QtGui, QtWidgets

from PyQt6.QtGui import QFont

from PyQt6.QtWidgets import QMessageBox, QApplication, QMainWindow, QVBoxLayout, QCheckBox, QWidget

from itertools import chain

from checkbox\_form import \*

from connection import \*

class Main(QtWidgets.QWidget, Ui\_Form):

def \_\_init\_\_(self, parent=None):

super(Main, self).\_\_init\_\_(parent)

self.setupUi(self)

self.checks = []

self.idMeter = None # id счётчка

self.listRateTariff = None # список значений стоимости тарифа

self.idTariff = None # id тарифа

self.diffReading = None # разница показний

self.discounts = [] # скидки за льготы

# device = QwSql().getDevice()

device = 1

facilities = QwSql().getFacilities(1)

tariffs = QwSql().getTariffs()

for name in facilities[0]:

if name != None:

self.checkbox = QCheckBox(name)

self.checks.append(self.checkbox)

self.verticalLayout.addWidget(self.checkbox)

if \_\_name\_\_ == "\_\_main\_\_":

import sys

app = QtWidgets.QApplication(sys.argv)

wind = Main()

wind.show()

sys.exit(app.exec()

**Combobox:**

from PyQt6 import QtCore, QtGui, QtWidgets

from PyQt6.QtGui import QFont

from PyQt6.QtWidgets import QMessageBox, QApplication, QMainWindow, QVBoxLayout, QCheckBox, QWidget

from itertools import chain

from combobox\_form import \*

from connection import \*

class Main(QtWidgets.QWidget, Ui\_Form):

def \_\_init\_\_(self, parent=None):

super(Main, self).\_\_init\_\_(parent)

self.setupUi(self)

self.checks = []

self.idMeter = None # id счётчка

self.listRateTariff = None # список значений стоимости тарифа

self.idTariff = None # id тарифа

self.diffReading = None # разница показний

self.discounts = [] # скидки за льготы

device = QwSql().getDevice()

facilities = QwSql().getFacilities

tariffs = QwSql().getTariffs()

# добавляем названия в comboBox

for i in device:

self.comboBox.addItem(i[1])

# блокировка первого элемента comboBox

self.comboBox.model().item(0).setEnabled(False)

”””

self.comboBox.currentTextChanged.connect(lambda: self.addFacilities(device, facilities))

def addFacilities(self, device, facilities):

while self.vLayout.count():

item = self.vLayout.takeAt(0)

widget = item.widget()

if widget:

widget.deleteLater()

comb = self.comboBox.currentText() # достаём значение comboBox

new\_data\_qw = list(chain.from\_iterable(device)) # достаём все вложенные списки в один общий

index = new\_data\_qw.index(comb) - 1

element = new\_data\_qw[index]

self.idMeter = element

facilities = facilities(element) # получаем льготы

for name in facilities[0]:

if name != None: # если льгота не пустая добавляем в layOut чекбоксы

self.checkbox = QCheckBox(name)

self.checks.append(self.checkbox)

self.vLayout.addWidget(self.checkbox)

”””

if \_\_name\_\_ == "\_\_main\_\_":

import sys

app = QtWidgets.QApplication(sys.argv)

wind = Main()

wind.show()

sys.exit(app.exec())

**RadioButton:**

from PyQt6 import QtCore, QtGui, QtWidgets

from PyQt6.QtGui import QFont

from PyQt6.QtWidgets import QMessageBox, QApplication, QMainWindow, QVBoxLayout, QCheckBox, QWidget

from itertools import chain

from radiobutton\_form import \*

from connection import \*

class Main(QtWidgets.QWidget, Ui\_Form):

def \_\_init\_\_(self, parent=None):

super(Main, self).\_\_init\_\_(parent)

self.setupUi(self)

self.checks = []

self.idMeter = None # id счётчка

self.listRateTariff = None # список значений стоимости тарифа

self.idTariff = None # id тарифа

self.diffReading = None # разница показний

self.discounts = [] # скидки за льготы

device = QwSql().getDevice()

facilities = QwSql().getFacilities

tariffs = QwSql().getTariffs()

# добавление radioButton в GroupBox

height = 40

for tarif in tariffs:

rad = QtWidgets.QRadioButton(parent=self.groupBox)

rad.setGeometry(20, height, 191, 21)

rad.setObjectName(f"{tarif[1]}")

rad.setText(f"{tarif[1]}")

rad.setAccessibleName(f"{tarif[0]}")

rad.setAccessibleDescription(f"{tarif[2]}")

rad.toggled.connect(self.onToggled) # соединение с sender чтобы определить какой radioButton нажат

height += 40

def onToggled(self, checked):

try:

radio = self.sender()

if checked:

self.listRateTariff = float(radio.accessibleDescription())

self.idTariff = radio.accessibleName()

except Exception as e:

print(e)

if \_\_name\_\_ == "\_\_main\_\_":

import sys

app = QtWidgets.QApplication(sys.argv)

wind = Main()

wind.show()

sys.exit(app.exec())

**RadioButton**

*from* PyQt6 *import* QtCore, QtGui, QtWidgets  
*from* PyQt6.QtGui *import* QFont  
*from* PyQt6.QtWidgets *import* QMessageBox, QApplication, QMainWindow, QVBoxLayout, QCheckBox, QWidget  
*from* itertools *import* chain  
*from* babai *import* \*  
*from* connection *import* \*  
  
  
*class* Main(QtWidgets.QWidget, Ui\_Form):  
 *def \_\_init\_\_*(*self*, parent=*None*):  
 *super*(Main, *self*).\_\_init\_\_(parent)  
 *self*.setupUi(*self*)  
  
 *self*.pushButton.clicked.connect(*self*.xp) *#подключение кнопки к функции которая вызывает хп* zapr = QwSql().get\_Radio() *# записываем в переменую результат запроса для извлечения в radiobutton  
  
 # добавление radioButton в GroupBox* height = 40  
 *for* i *in* zapr: *# в цикле заполняем radiobutton* rad = QtWidgets.QRadioButton(parent=*self*.groupBox)  
 rad.setGeometry(20, height, 191, 21)  
 rad.setObjectName(f"{i[1]}") *# даём объекту имя* rad.setText(f"{i[2]}") *# то что видим y Radiobutton* rad.setAccessibleDescription(f"{i[0]}") *# то что Rdiobutton хранит i[0] - это id таблицы* rad.toggled.connect(*self*.onToggled) *# соединение с sender чтобы определить какой radioButton нажат* height += 40  
  
 *def* onToggled(*self*, checked):  
 *try*:  
 radio = *self*.sender()  
 *if* checked:  
 *self*.radioData = radio.accessibleDescription() *#то что хранит нажатый RAdiobutton  
 print*(*self*.radioData)  
 *except Exception as* e:  
 *print*(e)  
  
 *def* xp(*self*):  
 edit = *self*.lineEdit.text() *#получаем текст с lineEdit* ex = QwSql().set\_data(*self*.radioData, edit) *# вызываем функцию из бд которая что-то записывает  
 self*.label.setText(*str*(ex)) *# то что функция вернула записываем в label  
 self*.label.adjustSize()  
  
  
  
*if* \_\_name\_\_ == "\_\_main\_\_":  
 *import* sys  
  
 app = QtWidgets.QApplication(sys.argv)  
 wind = Main()  
 wind.show()  
 sys.exit(app.exec())