**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**КЕМЕРОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**Институт Цифры**

**ОТЧЕТ**

**О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ №5**

по дисциплине «Разработка современных приложений на Python»

студента 3 курса

**Копытова Андрея Викторовича**

Направление 02.03.02– «Фундаментальная информатика и информационные технологии»

Преподаватель:

ст. пр. Зимин А. И.

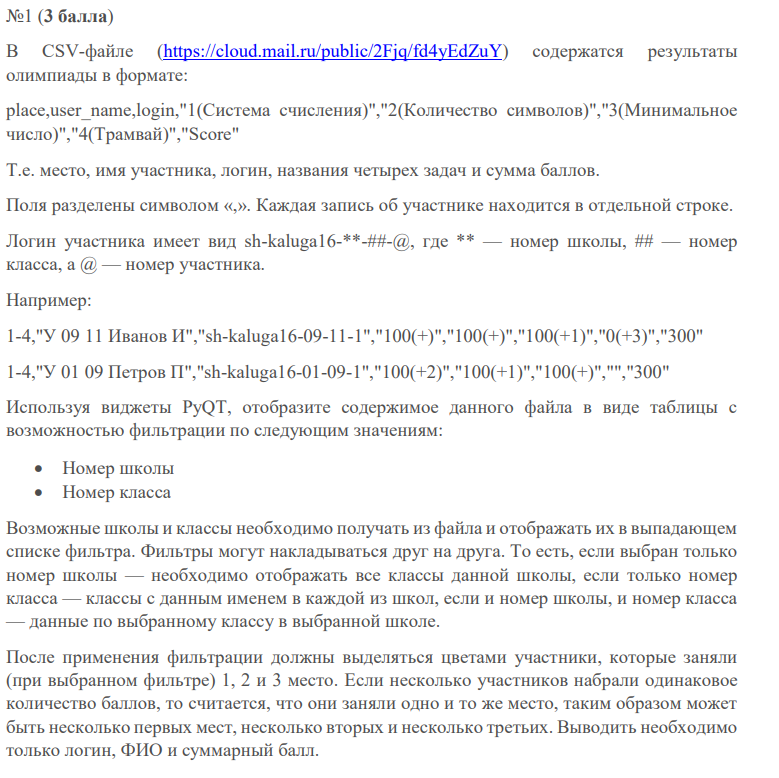
Работа защищена

«\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_»

“\_\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_2023 г.

Кемерово 2023

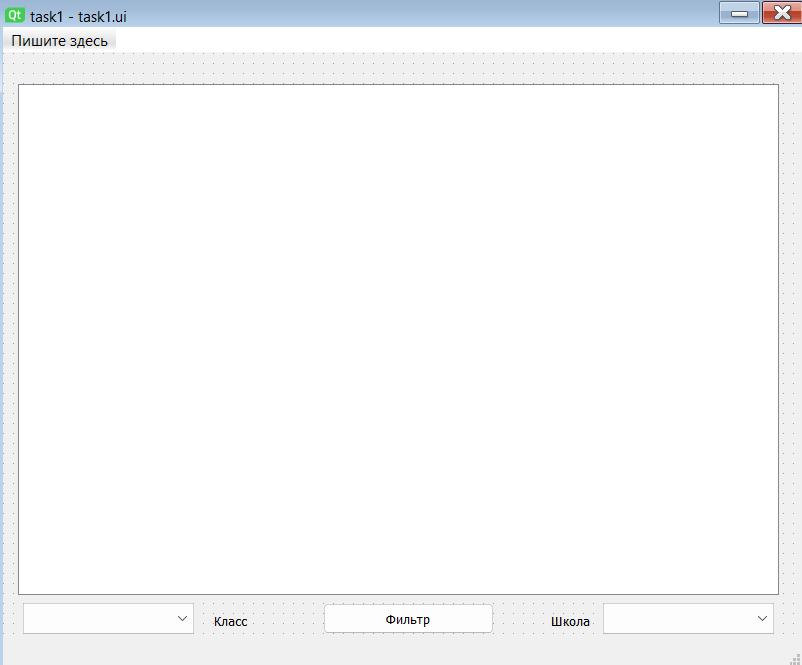
**Основная часть**

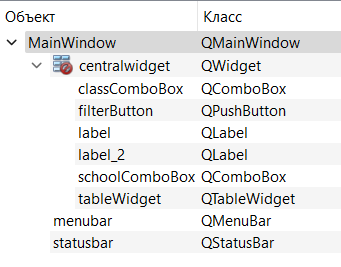


task1.py

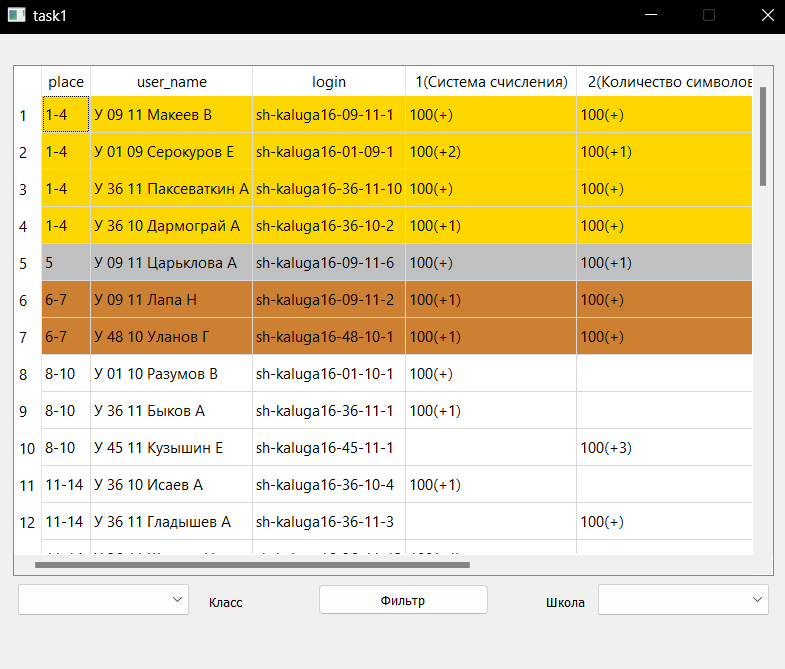
import csv  
import sys  
  
from PyQt5 import uic  
from PyQt5.QtWidgets import QApplication  
from PyQt5.QtWidgets import QMainWindow, QTableWidgetItem  
from PyQt5.QtGui import QColor  
  
  
class MyWidget(QMainWindow):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 uic.loadUi('task1.ui', self)  
 self.setFixedSize(800, 640)  
 self.loadTable('rez.csv')  
  
 self.filterData("", "")  
 self.filterButton.clicked.connect(self.applyFilter)  
  
 def loadTable(self, table\_name):  
 with open(table\_name, encoding="utf8") as csvfile:  
 reader = csv.reader(csvfile, delimiter=',')  
 self.data = list(reader)  
 title = self.data[0]  
 self.tableWidget.setColumnCount(len(title))  
 self.tableWidget.setHorizontalHeaderLabels(title)  
 self.tableWidget.setRowCount(len(self.data) - 1)  
  
 for i, row in enumerate(self.data[1:]):  
 for j, elem in enumerate(row):  
 self.tableWidget.setItem(i, j, QTableWidgetItem(elem))  
  
 self.tableWidget.resizeColumnsToContents()  
  
 schools = set()  
 clss = set()  
  
 self.schoolComboBox.addItem(None)  
 self.classComboBox.addItem(None)  
  
 for row in range(self.tableWidget.rowCount()):  
 school, cls = self.getSchoolAndClass(row)  
  
 schools.add(school)  
 clss.add(cls)  
  
 schools = sorted(schools)  
 clss = sorted(clss)  
  
 for school in schools:  
 self.schoolComboBox.addItem(school)  
 for cls in clss:  
 self.classComboBox.addItem(cls)  
  
 def applyFilter(self):  
 school\_number = self.schoolComboBox.currentText()  
 class\_number = self.classComboBox.currentText()  
  
 if not school\_number and not class\_number:  
 for row in range(self.tableWidget.rowCount()):  
 self.tableWidget.setRowHidden(row, False)  
 else:  
 self.filterData(school\_number, class\_number)  
  
 def filterData(self, school\_filter, class\_filter):  
 for row in range(self.tableWidget.rowCount()):  
 school, cls = self.getSchoolAndClass(row)  
  
 if (not school\_filter or school\_filter == school) and (not class\_filter or class\_filter == cls):  
 self.tableWidget.setRowHidden(row, False)  
 else:  
 self.tableWidget.setRowHidden(row, True)  
  
 self.colorRows()  
  
 def colorRows(self):  
 scores = {}  
 unique\_scores = []  
  
 for row in range(self.tableWidget.rowCount()):  
 login = self.tableWidget.item(row, 2).text()  
 score = int(self.tableWidget.item(row, 7).text())  
  
 if score not in scores:  
 scores[score] = []  
 unique\_scores.append(score)  
  
 scores[score].append((row, login, score))  
  
 unique\_scores.sort(reverse=True)  
  
 for i, score in enumerate(unique\_scores[:3]):  
 for row, login, \_ in scores[score]:  
 for col in range(self.tableWidget.columnCount()):  
 item = self.tableWidget.item(row, col)  
 if item:  
 if i == 0:  
 item.setBackground(QColor(255, 215, 0))  
 elif i == 1:  
 item.setBackground(QColor(192, 192, 192))  
 elif i == 2:  
 item.setBackground(QColor(205, 127, 50))  
 else:  
 item.setBackground(QColor(255, 255, 255))  
  
 def getSchoolAndClass(self, row):  
 login = self.tableWidget.item(row, 2).text()  
 school, cls = login.split('-')[2:4]  
 return school, cls  
  
  
def except\_hook(cls, exception, traceback):  
 sys.\_\_excepthook\_\_(cls, exception, traceback)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app = QApplication(sys.argv)  
 ex = MyWidget()  
 ex.show()  
 sys.excepthook = except\_hook  
 sys.exit(app.exec\_())

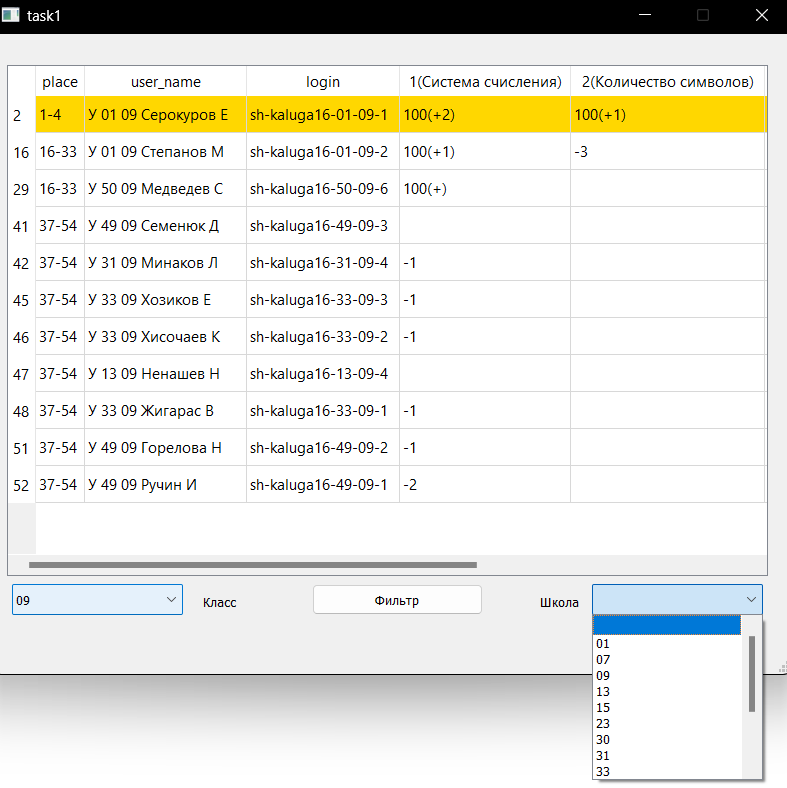
Интерфейс

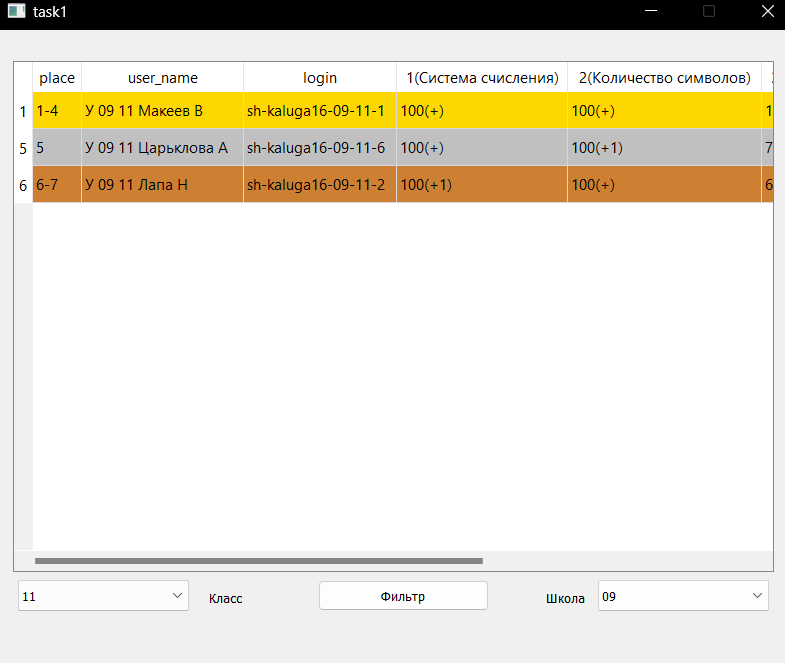


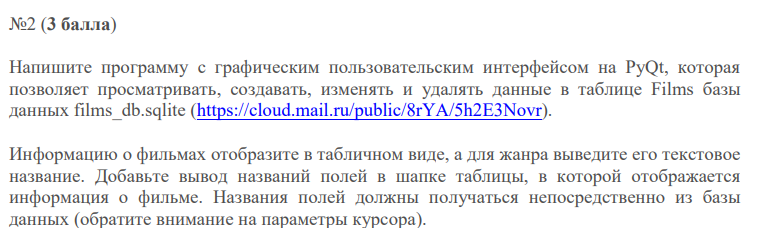


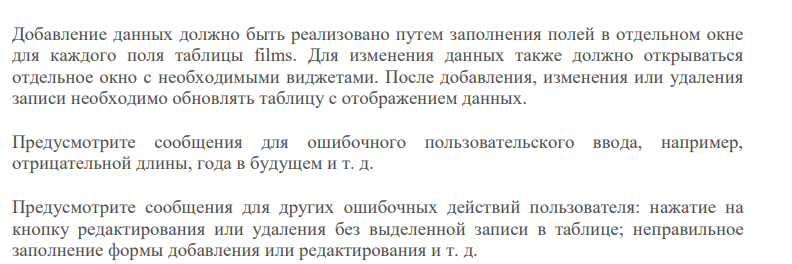
Демонстрация работы







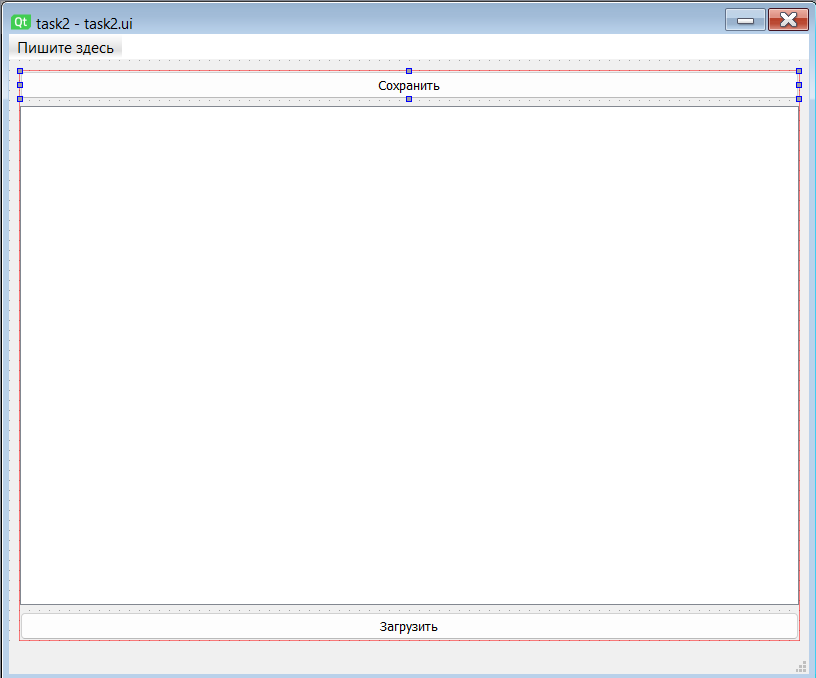


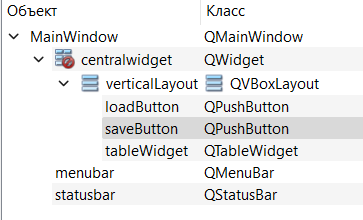


task2.py

import sqlite3  
import sys  
from PyQt5 import uic  
from PyQt5.QtWidgets import QMainWindow, QApplication, QTableWidgetItem  
from sqlite3 import Error  
  
  
class MyWidget(QMainWindow):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 uic.loadUi('task2.ui', self)  
 self.setFixedSize(800, 640)  
 self.db = Repository('films\_db.sqlite')  
  
 self.loadButton.clicked.connect(self.fillTableWidget)  
 self.saveButton.clicked.connect(self.updateDatabase)  
  
 def fillTableWidget(self):  
 films = self.db.getFilms()  
 self.tableWidget.clear()  
 num\_columns = len(films[0])  
 num\_rows = len(films)  
 self.tableWidget.setColumnCount(num\_columns)  
 self.tableWidget.setRowCount(num\_rows)  
 for i, film in enumerate(films):  
 for j, field in enumerate(film):  
 self.tableWidget.setItem(i, j, QTableWidgetItem(str(field)))  
  
 def updateDatabase(self):  
 for row in range(self.tableWidget.rowCount()):  
 film = Film(  
 self.tableWidget.item(row, 0).text(),  
 self.tableWidget.item(row, 1).text(),  
 self.tableWidget.item(row, 2).text(),  
 self.tableWidget.item(row, 3).text(),  
 self.tableWidget.item(row, 4).text()  
 )  
 self.db.updateFilm(film)  
  
  
class Film:  
 def \_\_init\_\_(self, id, title, year, genre, duration):  
 self.id = id  
 self.title = title  
 self.year = year  
 self.genre = genre  
 self.duration = duration  
  
  
class Repository:  
 def \_\_init\_\_(self, database\_name):  
 self.database = database\_name  
  
 def getConnection(self):  
 connection = None  
 try:  
 connection = sqlite3.connect(self.database)  
 return connection  
 except Error as e:  
 print(e)  
  
 return connection  
  
 def getFilms(self):  
 conn = self.getConnection()  
 cursor = conn.cursor()  
 cursor.execute("SELECT \* FROM films")  
 films = cursor.fetchall()  
  
 return films  
  
 def updateFilm(self, film):  
 conn = self.getConnection()  
 cursor = conn.cursor()  
 cursor.execute("UPDATE films SET title = ?, year = ?, genre = ?, duration = ? WHERE id = ?",  
 (film.title, film.year, film.genre, film.duration, film.id))  
 conn.commit()  
  
  
def except\_hook(cls, exception, traceback):  
 sys.\_\_excepthook\_\_(cls, exception, traceback)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app = QApplication(sys.argv)  
 ex = MyWidget()  
 ex.show()  
 sys.excepthook = except\_hook  
 sys.exit(app.exec\_())

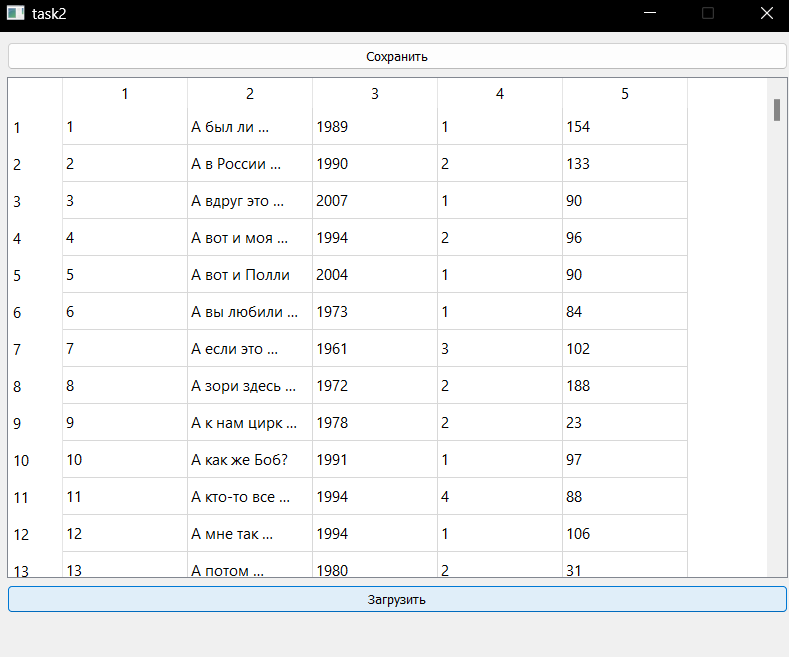
Интерфейс



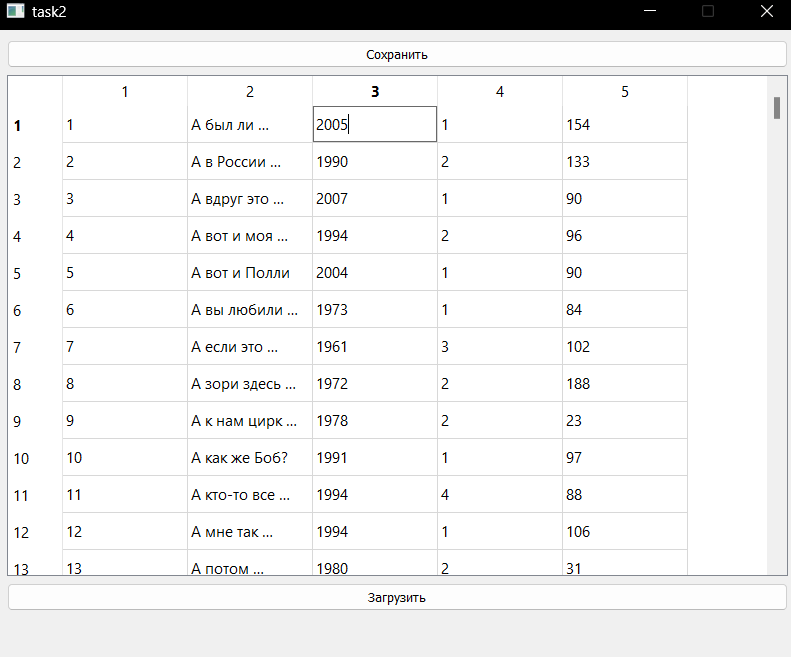


Демонстрация работы

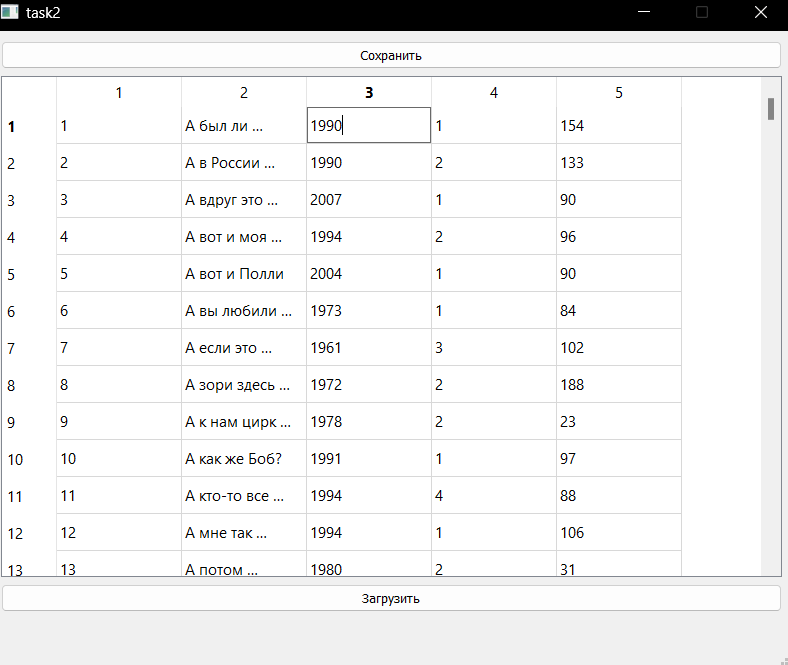
*Загрузили базу данных*



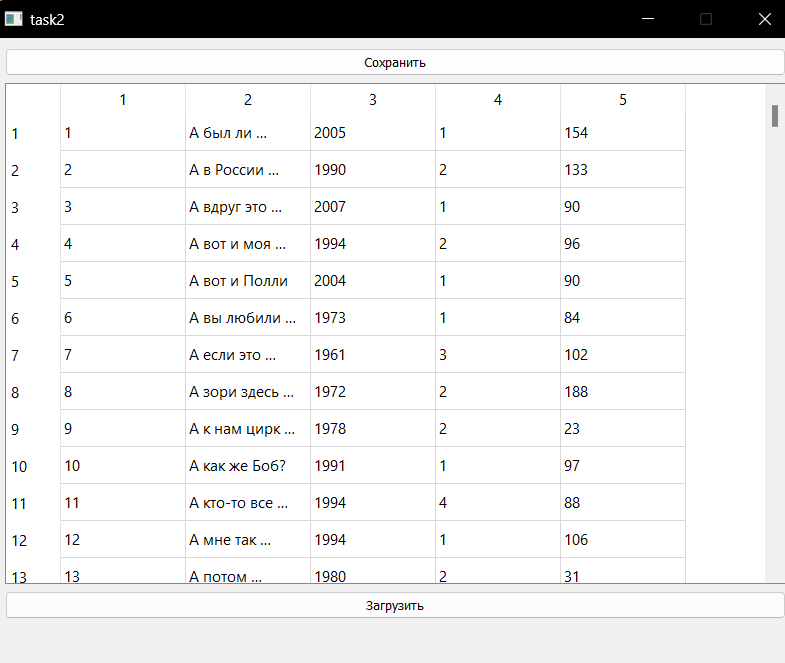
*Меняем произвольное поле и нажимаем «Сохранить»*

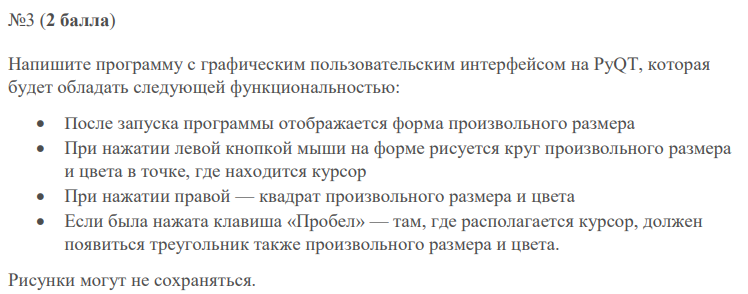


*Снова меняем значение этого поля*

**

*Нажимаем «Загрузить» и видим, что предыдущее изменение базы данных было загружено*

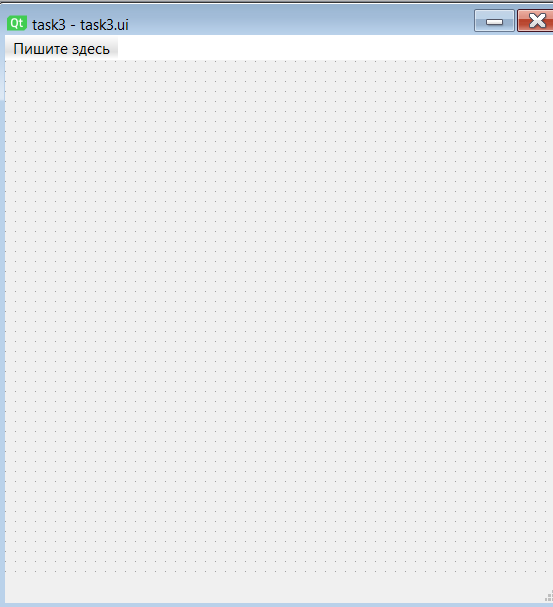
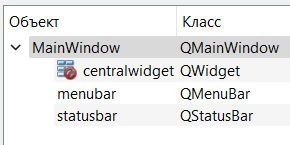
**

**

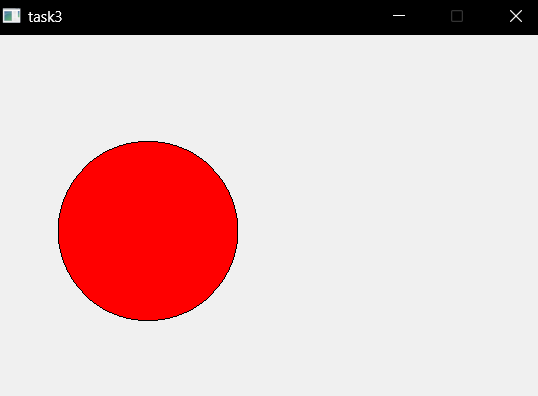
task3.py

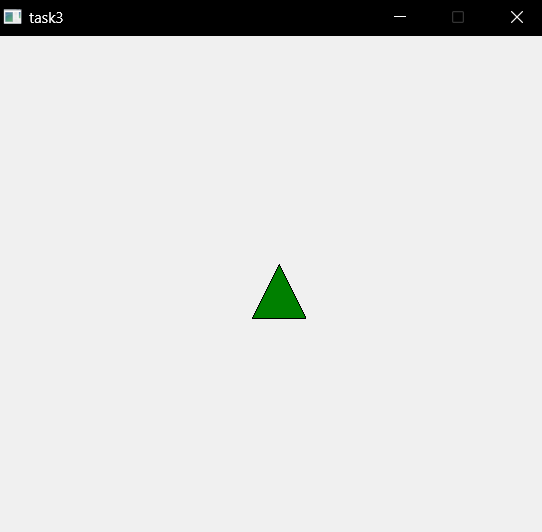
from PyQt5 import uic  
from PyQt5.QtGui import QPolygon  
from PyQt5.QtCore import Qt, QPoint  
from PyQt5.QtGui import QPainter, QColor  
from PyQt5.QtWidgets import QApplication, QMainWindow  
from random import choice, randint  
import sys  
  
  
class MyWidget(QMainWindow):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 uic.loadUi('task3.ui', self)  
 self.setFixedSize(553, 568)  
  
 self.D\_x, self.D\_y = -1, -1  
 self.x, self.y = -1, -1  
 self.GOG = None  
 self.colors = ['Red', 'Orange', 'Yellow', 'Green', 'Cyan',  
 'Blue', 'Magenta', 'Purple', 'Brown', 'Black', 'Pink']  
  
 def setupUi(self, Form):  
 self.setMouseTracking(True)  
 Form.resize(500, 500)  
  
 def mousePressEvent(self, event):  
 self.x = event.x()  
 self.y = event.y()  
 if event.button() == Qt.RightButton:  
 self.GOG = 1  
 elif event.button() == Qt.LeftButton:  
 self.GOG = -1  
 self.update()  
  
 def keyPressEvent(self, event):  
 if event.key() == Qt.Key\_Space:  
 self.GOG = 2  
 self.update()  
  
 def mouseMoveEvent(self, event):  
 self.D\_x = event.x()  
 self.D\_y = event.y()  
  
 def paintEvent(self, event):  
 qp = QPainter()  
 qp.begin(self)  
 self.drawing(qp)  
 qp.end()  
  
 def drawing(self, qp):  
 if self.x > -1 and self.y > -1 and self.GOG == 1:  
 qp.setBrush(QColor(choice(self.colors)))  
  
 leigt = randint(50, 200)  
  
 qp.drawRect(self.x - leigt // 2, self.y - leigt // 2, leigt, leigt)  
 ex.show()  
  
 elif self.x > -1 and self.y > -1 and self.GOG == -1:  
 qp.setBrush(QColor(choice(self.colors)))  
  
 leigt = randint(50, 200)  
  
 qp.drawEllipse(self.x - leigt // 2, self.y - leigt // 2, leigt,  
 leigt)  
  
 elif self.D\_x > -1 and self.D\_y > -1 and self.GOG == 2:  
 qp.setBrush(QColor(choice(self.colors)))  
  
 leigt = randint(50, 200)  
  
 points = QPolygon(  
 [QPoint((self.D\_x + leigt // 2), (self.D\_y + leigt // 2)),  
 QPoint(self.D\_x, self.D\_y - leigt // 2),  
 QPoint(self.D\_x - leigt // 2,  
 self.D\_y + leigt // 2)])  
 qp.drawPolygon(points)  
  
  
def except\_hook(cls, exception, traceback):  
 sys.\_\_excepthook\_\_(cls, exception, traceback)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app = QApplication(sys.argv)  
 ex = MyWidget()  
 ex.show()  
 sys.excepthook = except\_hook  
 sys.exit(app.exec\_())

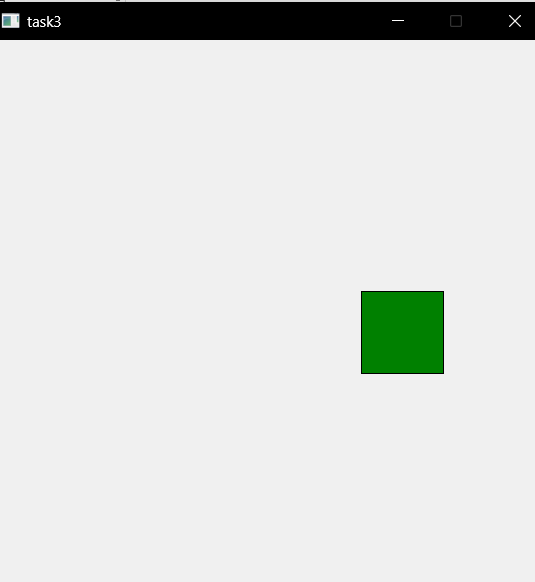
Интерфейс

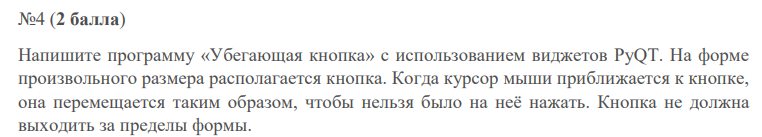
 

Демонстрация работы





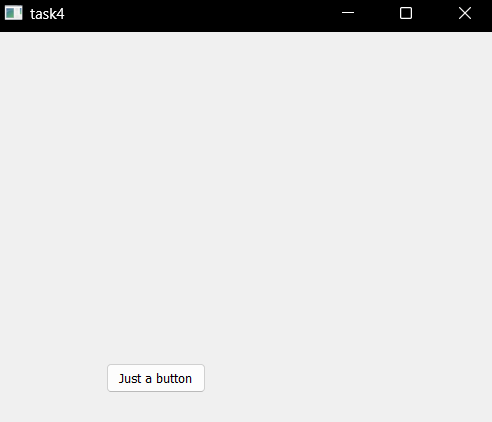
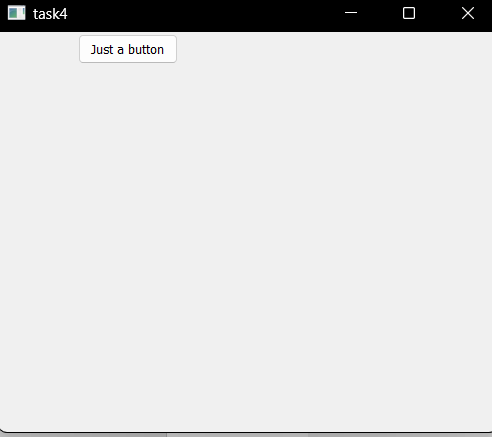


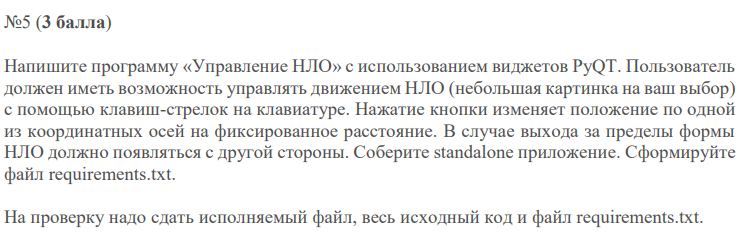


task4.py

import sys  
import random  
from PyQt5.QtWidgets import QApplication, QPushButton, QMainWindow  
from PyQt5.QtCore import pyqtSignal  
  
  
class Button(QPushButton):  
 mouseMoved = pyqtSignal()  
  
 def mouseMoveEvent(self, event):  
 self.mouseMoved.emit()  
  
  
class MyWidget(QMainWindow):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 self.coords = [40, 40]  
 self.btn\_size = [120, 40]  
 self.d = 15  
 self.setGeometry(300, 300, 500, 400)  
 self.setWindowTitle('task4')  
  
 self.btn = Button(self)  
 self.btn.setMouseTracking(True)  
 self.btn.setText("Just a button")  
 self.btn.move(\*self.coords)  
 self.btn.mouseMoved.connect(self.moveButton)  
 self.show()  
  
 def moveButton(self):  
 self.coords[0] = random.randint(0, 500 - self.btn\_size[0])  
 self.coords[1] = random.randint(0, 400 - self.btn\_size[1])  
 self.btn.move(\*self.coords)  
  
  
def except\_hook(cls, exception, traceback):  
 sys.\_\_excepthook\_\_(cls, exception, traceback)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app = QApplication(sys.argv)  
 ex = MyWidget()  
 sys.excepthook = except\_hook  
 ex.show()  
 sys.exit(app.exec\_())

Демонстрация работы

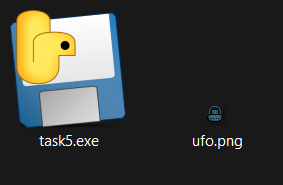


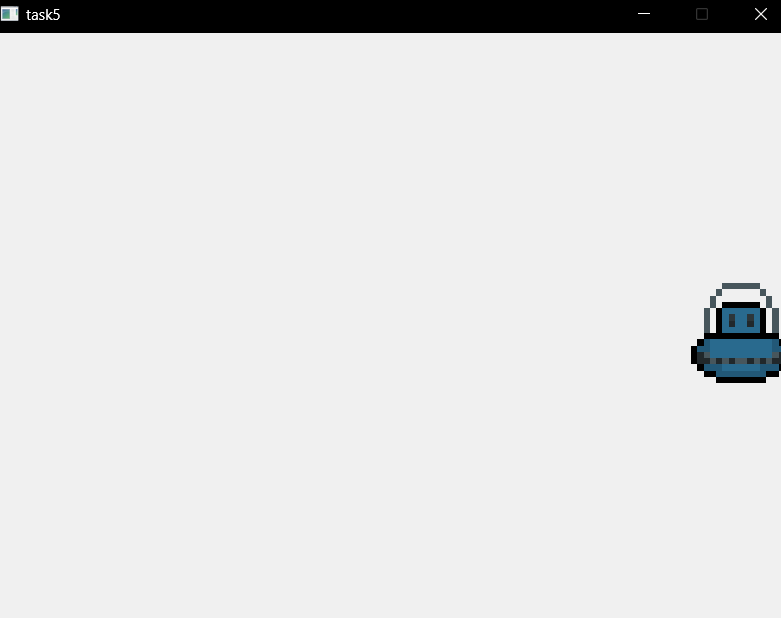
task5.py

from PyQt5.QtWidgets import QApplication, QWidget  
from PyQt5.QtGui import QPixmap, QPainter, QKeyEvent  
from PyQt5.QtCore import Qt  
import sys  
  
  
class MainWindow(QWidget):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 self.setWindowTitle("task5")  
 self.setFixedSize(800, 600)  
  
 self.ufo = QPixmap("ufo.png").scaled(100, 100)  
 self.ufo\_position = self.width() // 2, self.height() // 2  
  
 def keyPressEvent(self, event: QKeyEvent):  
 if event.key() == Qt.Key\_A:  
 self.ufo\_position = (self.ufo\_position[0] - 25, self.ufo\_position[1])  
 elif event.key() == Qt.Key\_D:  
 self.ufo\_position = (self.ufo\_position[0] + 25, self.ufo\_position[1])  
 elif event.key() == Qt.Key\_W:  
 self.ufo\_position = (self.ufo\_position[0], self.ufo\_position[1] - 25)  
 elif event.key() == Qt.Key\_S:  
 self.ufo\_position = (self.ufo\_position[0], self.ufo\_position[1] + 25)  
  
 if self.ufo\_position[0] < 0:  
 self.ufo\_position = (self.width(), self.ufo\_position[1])  
 elif self.ufo\_position[0] > self.width():  
 self.ufo\_position = (0, self.ufo\_position[1])  
 elif self.ufo\_position[1] < 0:  
 self.ufo\_position = (self.ufo\_position[0], self.height())  
 elif self.ufo\_position[1] > self.height():  
 self.ufo\_position = (self.ufo\_position[0], 0)  
  
 self.update()  
  
 self.update()  
  
 def paintEvent(self, event):  
 painter = QPainter(self)  
 painter.drawPixmap(\*self.ufo\_position, self.ufo)  
  
  
def except\_hook(cls, exception, traceback):  
 sys.\_\_excepthook\_\_(cls, exception, traceback)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app = QApplication(sys.argv)  
 ex = MainWindow()  
 ex.show()  
 sys.excepthook = except\_hook  
 sys.exit(app.exec\_())

Демонстрация работы

*Содержимое директории “…/task5/dist/”*

**

**

*Продолжаем движение вправо*

**