|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство науки и высшего образования Российской Федерации**  Калужский филиал  федерального государственного бюджетного  образовательного учреждения высшего образования  ***«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»***  ***(КФ МГТУ им. Н.Э. Баумана)*** |

**ФАКУЛЬТЕТ** \_***ИУК «Информатика и Управление»*\_\_**\_\_\_\_\_\_\_\_\_\_\_\_

**КАФЕДРА** \_\_***ИУК4 «Программное обеспечение ЭВМ, информационные технологии»***

**ЛАБОРАТОРНАЯ РАБОТА №2**

**ДИСЦИПЛИНА: «Кроссплатформенная разработка ПО»**

|  |  |  |
| --- | --- | --- |
| Выполнил: студент гр. ИУК4-62Б | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ( Калашников А. С.)  (Подпись) (Ф.И.О.) |
| Проверил: | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ( Пчелинцева Н. Н. )  (Подпись) (Ф.И.О.) |
| Дата сдачи (защиты):  Результаты сдачи (защиты): | | |
|  | - Балльная оценка:  - Оценка: | |
| Калуга, 2023 | | |

**Цель**: Получить навык разработки приложения с использованием фреймворка Qt.

**Задачи**:

1. Ознакомиться со средой разработки Qt Creator.

2. Получить навыки написания программ с использованием фреймворка Qt. **Задание**: В качестве предметной области использовать вариант из лабораторной работы №1. Разработать приложение с графическим интерфейсом на основе фреймворка Qt. Приложение должно реализовывать функционал из предыдущей работы. В приложение добавить возможность авторизации пользователей. Приложение должно работать на разных платформах (Linux, Windows).

**Код:**

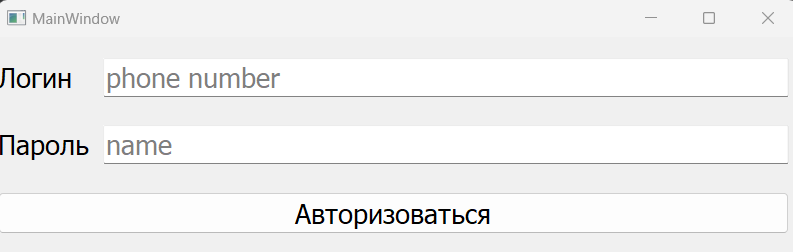
**Main.py**

import datetime  
import sys  
from PyQt5 import uic  
from PyQt5 import QtWidgets  
from PyQt5.QtWidgets import QDialog, QApplication  
from PyQt5.uic import loadUi  
from re import sub  
from decimal import Decimal  
from PyQt5.QtWidgets import QMessageBox  
from datetime import date  
from Autho import \*  
from autho\_db import \*  
from patient\_db import \*  
from contract\_db import \*  
from doctor\_db import \*  
from patient\_card\_db import \*  
from Menu import \*  
from Contract import \*  
from ContractAdd import \*  
from Doctor import \*  
from DoctorAdd import \*  
from Patient import \*  
from PatientAdd import \*  
from PatientCard import \*  
from PatientCardAdd import \*  
from dateutil import parser  
  
class Menu(QtWidgets.QMainWindow):  
 def \_\_init\_\_(self, parent=None):  
 super().\_\_init\_\_(parent)  
 self.ui = Ui\_MainWindow()  
 self.ui.setupUi(self)  
 self.text = ''  
 self.ui.comboTable.addItem("Пациенты")  
 self.ui.comboTable.addItem("Доктора")  
 self.ui.comboTable.addItem("Контракты")  
 self.ui.comboTable.addItem("Карты пациентов")  
 self.ui.comboTable.currentTextChanged.connect(self.set\_new\_data)  
 self.ui.listWidget.itemClicked.connect(self.edit\_table)  
 self.ui.buttonAdd.clicked.connect(self.add\_data)  
  
 def edit\_table(self, item):  
 if "Пациенты" == self.text:  
 patient\_str = item.text()  
 patient\_list = patient\_str.split(" ")  
 menu.close()  
 patient.id = patient\_list[1]  
 search\_patient = get\_data\_patient\_id(patient.id)  
 patient.set\_new\_data(search\_patient)  
 patient.show()  
 elif "Доктора" == self.text:  
 doctor\_str = item.text()  
 doctor\_list = doctor\_str.split(" ")  
 menu.close()  
 print(doctor\_list)  
 doctor.id = doctor\_list[1]  
 search\_doctor = get\_data\_doctor\_id(doctor.id)  
 doctor.set\_new\_data(search\_doctor)  
 doctor.show()  
 elif "Контракты" == self.text:  
 contract\_str = item.text()  
 contract\_list = contract\_str.split(" ")  
 print(contract\_list)  
 menu.close()  
 contract.id = contract\_list[1]  
 search\_contract = get\_data\_contract\_id(contract.id)  
 contract.set\_new\_data(search\_contract)  
 contract.add\_patient\_card()  
 contract.show()  
  
 else:  
 patient\_card\_str = item.text()  
 patient\_card\_list = patient\_card\_str.split(" ")  
 print(patient\_card\_list)  
 menu.close()  
 patientCard.id = patient\_card\_list[1]  
 search\_patient\_card = get\_data\_patient\_card\_id(patientCard.id)  
 patientCard.set\_new\_data(search\_patient\_card)  
 patientCard.show()  
  
  
 def add\_data(self):  
 menu.close()  
 if "Пациенты" == self.text:  
 patientAdd.show()  
 elif "Доктора" == self.text:  
 doctorAdd.show()  
 elif "Контракты" == self.text:  
 contractAdd.show()  
 contractAdd.add\_patient()  
 else:  
 patientCardAdd.show()  
 patientCardAdd.add\_contract()  
 patientCardAdd.add\_doctor()  
  
  
 def set\_new\_data(self, text):  
 if "Пациенты" == text:  
 self.text = text  
 self.ui.listWidget.clear()  
 data\_list = get\_data\_patient()  
 for patient\_data in data\_list:  
 date = patient\_data[2].strftime("%m %d %Y")  
 str\_patient\_data = "Id: "+str(patient\_data[0])+" \nИмя: " + patient\_data[1] + " \nДата: " + date + " \nТелефон: " + patient\_data[3]  
 self.ui.listWidget.addItem(str\_patient\_data)  
  
 elif "Доктора" == text:  
 self.text = text  
 data\_list = get\_data\_doctor()  
 self.ui.listWidget.clear()  
 for doctor\_data in data\_list:  
 str\_doctor\_data = "Id: "+str(doctor\_data[0]) +" \nИмя: " + doctor\_data[1] + " \nКоэфицент: " + str(doctor\_data[2]) +" \nДолжность:" + doctor\_data[3] \  
 + " \nТелефон: " + doctor\_data[4]  
 self.ui.listWidget.addItem(str\_doctor\_data)  
  
 elif "Контракты" == text:  
 self.text = text  
 self.ui.listWidget.clear()  
 data\_list = get\_data\_contract()  
  
 for contract\_data in data\_list:  
 list = get\_data\_patient\_card\_in\_contract\_name\_proc(str(contract\_data[0]))  
 list\_price = get\_data\_patient\_card\_in\_contract\_price(str(contract\_data[0]))  
 price=0  
 for cost in list\_price:  
 price = price+cost[0]  
 if price:  
 update\_data\_price(str(contract\_data[0]), price)  
 if list:  
 str\_name\_proc = ""  
 for element in list:  
 str\_name\_proc = str\_name\_proc + str(element[0]) + " "  
 else:  
 str\_name\_proc = ""  
 date = contract\_data[3].strftime("%m %d %Y")  
 str\_contract\_data = "Id: "+str(contract\_data[0]) +" \nНомер контракта: " + contract\_data[1] + " \nИтог: " + str(contract\_data[2]) +" \nДата:" + date \  
 + " \nСписок процедур: " + str\_name\_proc  
 self.ui.listWidget.addItem(str\_contract\_data)  
  
  
 else:  
 self.text = text  
 self.ui.listWidget.clear()  
 data\_list = get\_data\_patient\_card()  
 for patient\_card\_data in data\_list:  
 date = patient\_card\_data[2].strftime("%m %d %Y")  
 str\_patient\_card\_data = "Id: "+str(patient\_card\_data[0])+" \nНомер карты: " + patient\_card\_data[1] + " \nДата: " + date + " \nНазвание процедуры: " + \  
 patient\_card\_data[3] + " \nЦена за услгу" + str(patient\_card\_data[4]) + " \nДоктор: " + str(patient\_card\_data[5]) + \  
 " \nКонтракт" + str(patient\_card\_data[6]) + " \nИтоговая цена" + str(patient\_card\_data[7])  
 self.ui.listWidget.addItem(str\_patient\_card\_data)  
  
  
  
  
class Doctor(QtWidgets.QMainWindow):  
 def \_\_init\_\_(self, parent=None):  
 super().\_\_init\_\_(parent)  
 self.ui = Ui\_Doctor()  
 self.ui.setupUi(self)  
 self.id = ""  
 self.ui.buttonAdd.clicked.connect(self.add\_new\_data)  
 self.ui.buttonChange.clicked.connect(self.change)  
 self.ui.buttonDelete.clicked.connect(self.delete)  
  
 def add\_new\_data(self):  
 doctor.close()  
 doctorAdd.show()  
  
 def change(self):  
 doctor.close()  
 update\_data\_doctor(self.id, self.ui.name.text(), self.ui.ratio.text(),self.ui.post.text(), self.ui.phoneNumber.text())  
 menu.set\_new\_data("Доктора")  
 menu.show()  
  
 def delete(self):  
 doctor.close()  
 delete\_data\_doctor(self.id)  
 menu.set\_new\_data("Доктора")  
 menu.show()  
  
 def set\_new\_data(self,search\_doctor):  
 if search\_doctor:  
 for new\_doctor in search\_doctor:  
 self.ui.name.setText(new\_doctor[1])  
 self.ui.ratio.setText(str(new\_doctor[2]))  
 self.ui.post.setText(new\_doctor[3])  
 self.ui.phoneNumber.setText(new\_doctor[4])  
class DoctorAdd(QtWidgets.QMainWindow):  
 def \_\_init\_\_(self, parent=None):  
 super().\_\_init\_\_(parent)  
 self.ui = Ui\_DoctorAdd()  
 self.ui.setupUi(self)  
 self.ui.buttonBack.clicked.connect(self.back)  
 self.ui.buttonAdd.clicked.connect(self.add\_new\_data)  
  
 def back(self):  
 doctorAdd.close()  
 menu.show()  
  
 def add\_new\_data(self):  
 doctorAdd.close()  
 set\_data\_doctor(self.ui.name.text(), self.ui.ratio.text(),self.ui.post.text(), self.ui.phoneNumber.text())  
 menu.set\_new\_data("Доктора")  
 menu.show()  
class Patient(QtWidgets.QMainWindow):  
 def \_\_init\_\_(self, parent=None):  
 super().\_\_init\_\_(parent)  
 self.ui = Ui\_Patient()  
 self.ui.setupUi(self)  
 self.id = ""  
 self.ui.buttonAdd.clicked.connect(self.add\_new\_data)  
 self.ui.buttonChange.clicked.connect(self.change)  
 self.ui.buttonDelete.clicked.connect(self.delete)  
  
 def add\_new\_data(self):  
 patient.close()  
 patientAdd.show()  
  
 def change(self):  
 patient.close()  
 day = parser.parse(self.ui.birthDate.text())  
 update\_data\_patient(self.id, self.ui.name.text(), day, self.ui.phoneNumber.text())  
 menu.set\_new\_data("Пациенты")  
 menu.show()  
  
 def delete(self):  
 patient.close()  
 delete\_data\_patient(self.id)  
 menu.set\_new\_data("Пациенты")  
 menu.show()  
  
 def set\_new\_data(self,search\_patient):  
 if search\_patient:  
 for new\_patient in search\_patient:  
 date = new\_patient[2].strftime("%m %d %Y")  
 self.ui.name.setText(new\_patient[1])  
 self.ui.phoneNumber.setText(new\_patient[3])  
 self.ui.birthDate.setText(date)  
  
class PatientAdd(QtWidgets.QMainWindow):  
 def \_\_init\_\_(self, parent=None):  
 super().\_\_init\_\_(parent)  
 self.ui = Ui\_PatientAdd()  
 self.ui.setupUi(self)  
 self.ui.buttonBack.clicked.connect(self.back)  
 self.ui.buttonAdd.clicked.connect(self.add\_new\_data)  
  
 def back(self):  
 patientAdd.close()  
 menu.show()  
  
 def add\_new\_data(self):  
 patientAdd.close()  
 day = parser.parse(self.ui.birthDate.text())  
 set\_data\_patient(self.ui.name.text(), day, self.ui.phoneNumber.text())  
 menu.set\_new\_data("Пациенты")  
 menu.show()  
  
class Contract(QtWidgets.QMainWindow):  
 def \_\_init\_\_(self, parent=None):  
 super().\_\_init\_\_(parent)  
 self.ui = Ui\_Contract()  
 self.ui.setupUi(self)  
 self.id = ""  
 self.patient\_str = ""  
 self.total\_cost = 0  
 self.ui.totalCost.setText(str(self.total\_cost))  
 self.ui.buttonAdd.clicked.connect(self.add\_new\_data)  
 self.ui.buttonChange.clicked.connect(self.change)  
 self.ui.buttonDelete.clicked.connect(self.delete)  
  
 def set\_new\_data(self,search\_contract):  
 if search\_contract:  
 for new\_contract in search\_contract:  
 date = new\_contract[3].strftime("%m %d %Y")  
 self.ui.numberContract.setText(new\_contract[1])  
 self.ui.totalCost.setText(str(new\_contract[2]))  
 self.ui.createDate.setText(date)  
 self.add\_patient()  
 def add\_patient\_card(self):  
 self.ui.listWidget.clear()  
 data\_list = get\_data\_patient\_card\_in\_contract(self.id)  
 print(data\_list)  
 if data\_list:  
 for patient\_card\_data in data\_list:  
 date = patient\_card\_data[2].strftime("%m %d %Y")  
 str\_patient\_card\_data = "Id: "+str(patient\_card\_data[0])+" \nНомер карты: " + patient\_card\_data[  
 1] + " \nДата: " + date + "\nНазвание процедуры: " + \  
 patient\_card\_data[3] + " \nЦена за услгу" + str(  
 patient\_card\_data[4]) + " \nДоктор: " + str(patient\_card\_data[5]) + \  
 " \nКонтракт" + str(patient\_card\_data[6]) + " \nИтоговая цена" + str(patient\_card\_data[7])  
 self.ui.listWidget.addItem(str\_patient\_card\_data)  
 def add\_patient(self):  
 data\_patient = get\_name\_patient()  
 if data\_patient:  
 for new\_patient in data\_patient:  
 self.ui.comboPatient.addItem(str(new\_patient[0])+" "+new\_patient[1])  
 def add\_new\_data(self):  
 contract.close()  
 contractAdd.add\_patient()  
 contractAdd.show()  
  
 def change(self):  
 contract.close()  
 day = parser.parse(self.ui.createDate.text())  
 patient\_str = self.ui.comboPatient.currentText()  
 update\_patient = patient\_str.split(" ")  
 update\_data\_contract(self.id, self.ui.numberContract.text(),  
 self.ui.totalCost.text(), day, update\_patient[0])  
 menu.set\_new\_data("Контракты")  
 menu.show()  
  
 def delete(self):  
 contract.close()  
 delete\_data\_contract\_patient\_card(self.id)  
 delete\_data\_contract(self.id)  
  
 menu.set\_new\_data("Контракты")  
 menu.show()  
  
  
class ContractAdd(QtWidgets.QMainWindow):  
 def \_\_init\_\_(self, parent=None):  
 super().\_\_init\_\_(parent)  
 self.ui = Ui\_ContractAdd()  
 self.ui.setupUi(self)  
 self.id = ""  
 self.patient\_str = ""  
 day = datetime.datetime.today()  
 date = day.strftime("%m %d %Y")  
 self.ui.createDate.setText(str(date))  
 self.ui.totalCost.setText("0")  
 self.ui.buttonBack.clicked.connect(self.back)  
 self.ui.buttonAdd.clicked.connect(self.add\_new\_data)  
  
 def add\_patient(self):  
 data\_patient = get\_name\_patient()  
 if data\_patient:  
 for new\_patient in data\_patient:  
 self.ui.comboPatient.addItem(str(new\_patient[0])+" "+new\_patient[1])  
 def back(self):  
 contractAdd.close()  
 menu.show()  
  
  
 def add\_new\_data(self):  
 contractAdd.close()  
 update\_patient = self.patient\_str.split(" ")  
 day = parser.parse(self.ui.createDate.text())  
 set\_data\_contract(self.ui.numberContract.text(), self.ui.totalCost.text(),day, update\_patient[0])  
 menu.set\_new\_data("Контракты")  
 menu.show()  
  
class PatientCard(QtWidgets.QMainWindow):  
 def \_\_init\_\_(self, parent=None):  
 super().\_\_init\_\_(parent)  
 self.ui = Ui\_PatientCard()  
 self.ui.setupUi(self)  
 self.id = ""  
 self.doctor\_str = ""  
 self.contract\_str = ""  
 self.ui.price.textChanged.connect(lambda: self.set\_total\_price(self.doctor\_str))  
 self.ui.buttonAdd.clicked.connect(self.add\_new\_data)  
 self.ui.buttonChange.clicked.connect(self.change)  
 self.ui.buttonDelete.clicked.connect(self.delete)  
 self.ui.comboDoctor.currentTextChanged.connect(self.set\_total\_price)  
 self.ui.comboContract.currentTextChanged.connect(self.set\_contract\_id)  
  
 def add\_doctor(self):  
 data\_doctor = get\_name\_doctor()  
 if data\_doctor:  
 for new\_doctor in data\_doctor:  
 self.ui.comboDoctor.addItem(str(new\_doctor[0])+" "+new\_doctor[1])  
  
 def add\_contract(self):  
 data\_contract = get\_name\_contract()  
 if data\_contract:  
 for new\_contract in data\_contract:  
 self.ui.comboContract.addItem(str(new\_contract[0])+" "+new\_contract[1])  
 def set\_total\_price(self,text):  
 self.doctor\_str = text  
 str\_id = text.split(" ")  
 ratio = get\_ratio\_doctor\_id(str\_id[0])  
 if ratio and self.ui.price.text():  
 self.ui.totalPrice.setText(str(int(self.ui.price.text()) \* ratio[0]))  
 else:  
 self.ui.totalPrice.setText("0")  
 def set\_contract\_id(self,text):  
 self.contract\_str = text  
  
 def add\_new\_data(self):  
 patientCard.close()  
 patientCardAdd.show()  
  
 def change(self):  
 patientCard.close()  
 day = parser.parse(self.ui.createDate.text())  
 update\_doctor = self.doctor\_str.split(" ")  
 update\_contract = self.contract\_str.split(" ")  
 update\_data\_patient\_card(self.id, self.ui.numberCard.text(), day, self.ui.nameProc.text(),  
 self.ui.price.text(),update\_doctor[0],update\_contract[0],self.ui.totalPrice.text())  
 menu.set\_new\_data("Карты пациентов")  
 menu.show()  
  
 def delete(self):  
 patientCard.close()  
 delete\_data\_patient\_card(self.id)  
 menu.set\_new\_data("Карты пациентов")  
 menu.show()  
  
 def set\_new\_data(self, search\_patient\_card):  
 self.add\_doctor()  
 self.add\_contract()  
 if search\_patient\_card:  
 for new\_patient\_card in search\_patient\_card:  
 date = new\_patient\_card[2].strftime("%m %d %Y")  
 self.ui.numberCard.setText(new\_patient\_card[1])  
 self.ui.createDate.setText(date)  
 self.ui.nameProc.setText(new\_patient\_card[3])  
 self.ui.price.setText(str(new\_patient\_card[4]))  
 self.ui.comboDoctor.addItem(str(new\_patient\_card[5]))  
 self.ui.comboContract.addItem(str(new\_patient\_card[6]))  
 self.ui.totalPrice.setText(str(new\_patient\_card[7]))  
  
class PatientCardAdd(QtWidgets.QMainWindow):  
 def \_\_init\_\_(self, parent=None):  
 super().\_\_init\_\_(parent)  
 self.ui = Ui\_PatientCardAdd()  
 self.ui.setupUi(self)  
 self.id = ""  
 self.doctor\_str = ""  
 self.contract\_str = ""  
 self.ui.price.setText("0")  
 self.ui.buttonAdd.clicked.connect(self.add\_new\_data)  
 self.ui.price.textChanged.connect(lambda: self.set\_total\_price(self.doctor\_str))  
 self.ui.buttonBack.clicked.connect(self.back)  
 self.ui.comboDoctor.currentTextChanged.connect(self.set\_total\_price)  
 self.ui.comboContract.currentTextChanged.connect(self.set\_contract\_id)  
  
 def back(self):  
 patientCardAdd.close()  
 menu.show()  
  
 def add\_new\_data(self):  
 patientAdd.close()  
 update\_doctor = self.doctor\_str.split(" ")  
 update\_contract = self.contract\_str.split(" ")  
 day = parser.parse(self.ui.createDate.text())  
 set\_data\_patient\_card(self.ui.numberCard.text(), day, self.ui.nameProc.text(),  
 self.ui.price.text(), update\_doctor[0], update\_contract[0], self.ui.totalPrice.text())  
 menu.set\_new\_data("Карты пациентов")  
 menu.show()  
 def add\_doctor(self):  
 data\_doctor = get\_name\_doctor()  
 if data\_doctor:  
 for new\_doctor in data\_doctor:  
 self.ui.comboDoctor.addItem(str(new\_doctor[0])+" "+new\_doctor[1])  
  
 def add\_contract(self):  
 data\_contract = get\_name\_contract()  
 if data\_contract:  
 for new\_contract in data\_contract:  
 self.ui.comboContract.addItem(str(new\_contract[0])+" "+new\_contract[1])  
 def set\_total\_price(self,text):  
 self.doctor\_str = text  
 str\_id = text.split(" ")  
 ratio = get\_ratio\_doctor\_id(str\_id[0])  
 if ratio and self.ui.price.text():  
 self.ui.totalPrice.setText(str(int(self.ui.price.text())\*ratio[0]))  
 else:  
 self.ui.totalPrice.setText("0")  
 def set\_contract\_id(self,text):  
 self.contract\_str = text  
  
class Autho(QtWidgets.QMainWindow):  
 def \_\_init\_\_(self, parent=None):  
 super().\_\_init\_\_(parent)  
 self.ui = Ui\_Autho()  
 self.ui.setupUi(self)  
 self.ui.pushButton.clicked.connect(self.authoriz)  
  
 def authoriz(self):  
  
 check = get\_data\_patient\_for\_autho(self.ui.login.text(), self.ui.password.text())  
 if check:  
 authori.close()  
 menu.show()  
 else:  
 msg = QMessageBox()  
 msg.setIcon(QMessageBox.Critical)  
 msg.setText("Авторизация:")  
 msg.setInformativeText('Пользователя с таким номером или паролем не существует')  
 msg.setWindowTitle("Error")  
 msg.exec\_()  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 name = ""  
 app = QtWidgets.QApplication(sys.argv)  
  
 authori = Autho()  
 menu = Menu()  
 contract = Contract()  
 contractAdd = ContractAdd()  
 doctor = Doctor()  
 doctorAdd = DoctorAdd()  
 patient = Patient()  
 patientAdd = PatientAdd()  
 patientCard = PatientCard()  
 patientCardAdd = PatientCardAdd()  
 authori.show()  
 sys.exit(app.exec\_())

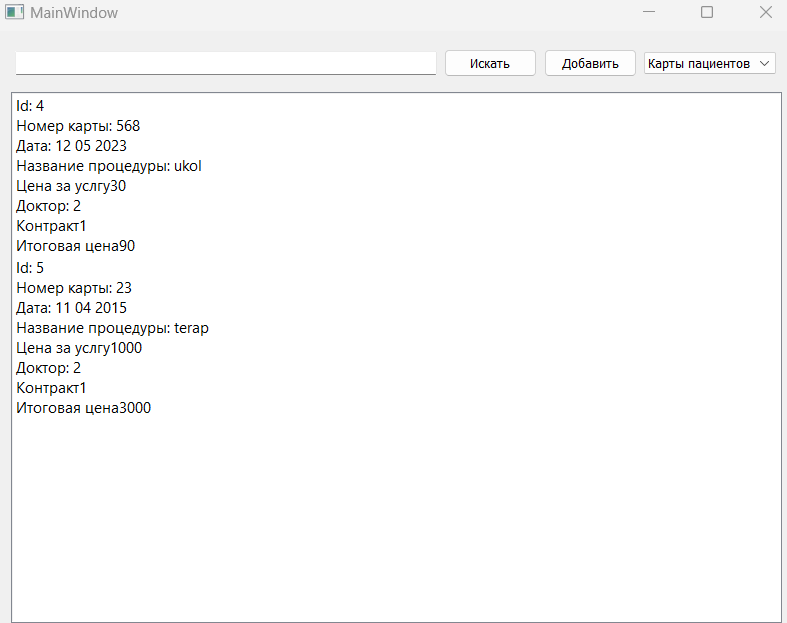
**Autho.py**

import psycopg2  
from config import host, user, password, db\_name  
  
def get\_data\_patient\_for\_autho(phone\_number,name):  
 try:  
 connection = psycopg2.connect(  
 host=host,  
 user=user,  
 password=password,  
 database=db\_name  
 )  
 connection.autocommit = True  
 # get data from a table  
 with connection.cursor() as cursor:  
 cursor.execute(  
 """SELECT   
 id,  
 name  
 FROM   
 patients  
 Where  
 phone\_number = '"""+ phone\_number +"""'  
 and  
 name = '"""+name+"""';"""  
 )  
 patient = cursor.fetchall()  
 return patient  
  
 except Exception as \_ex:  
 connection = False  
 print("[INFO] Error while working with PostgreSQL", \_ex)  
 finally:  
 if connection:  
 connection.close()  
 print("[INFO] PostgreSQL connection closed")

**Решение:**

****

**Рис. 1 Авторизация**

****

**Рис. 2 Вывод**

**Вывод:** в ходе выполнения лабораторной работы были получены практические навыки разработки приложения с использованием фреймворка Qt.