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

«Разработка кроссплатформенного приложения с использованием фреймворка Qt»

ДИСЦИПЛИНА: «Кроссплатформенная разработка ПО»

Выполнил: студент гр. ИУК4-62Б ________________ (Карельский М.К.)
(Подпись)

Проверил: ________________ (Пчелинцева Н.И.)
(Подпись)

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).

Вариант 8

Предметная область – типография.

Описание и UML-диаграмма предметной области:

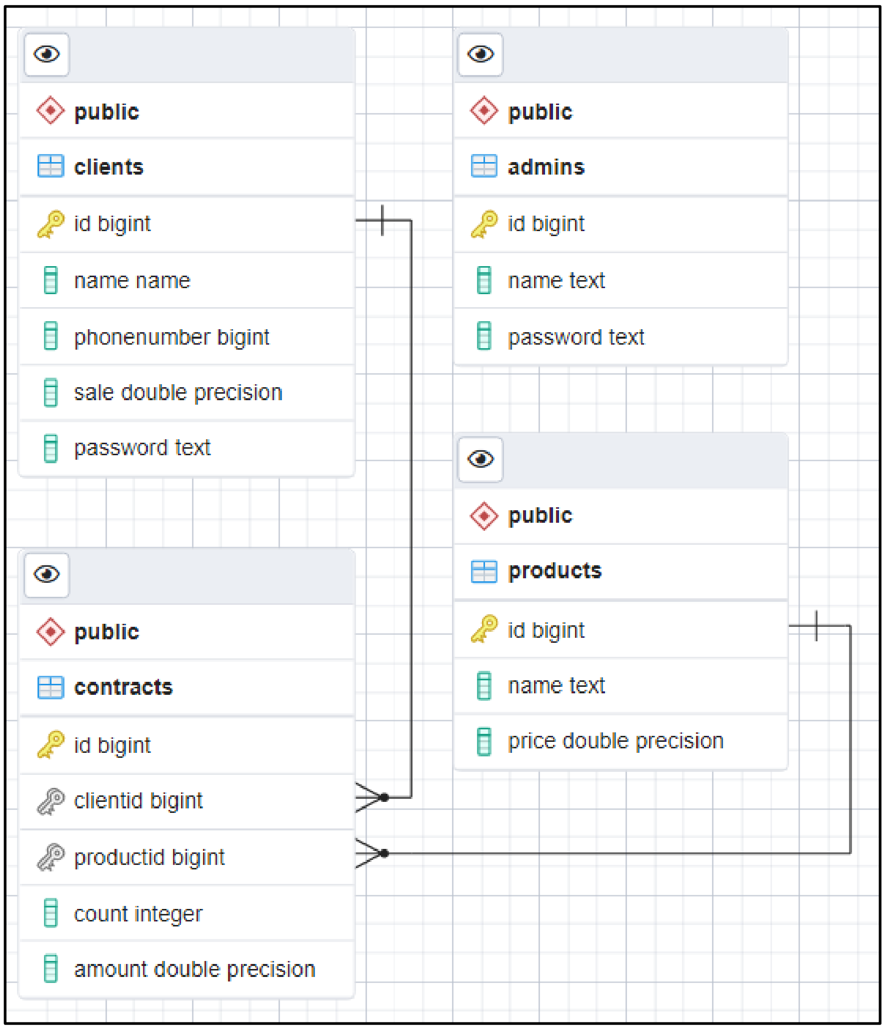


Рис. 1. База данных

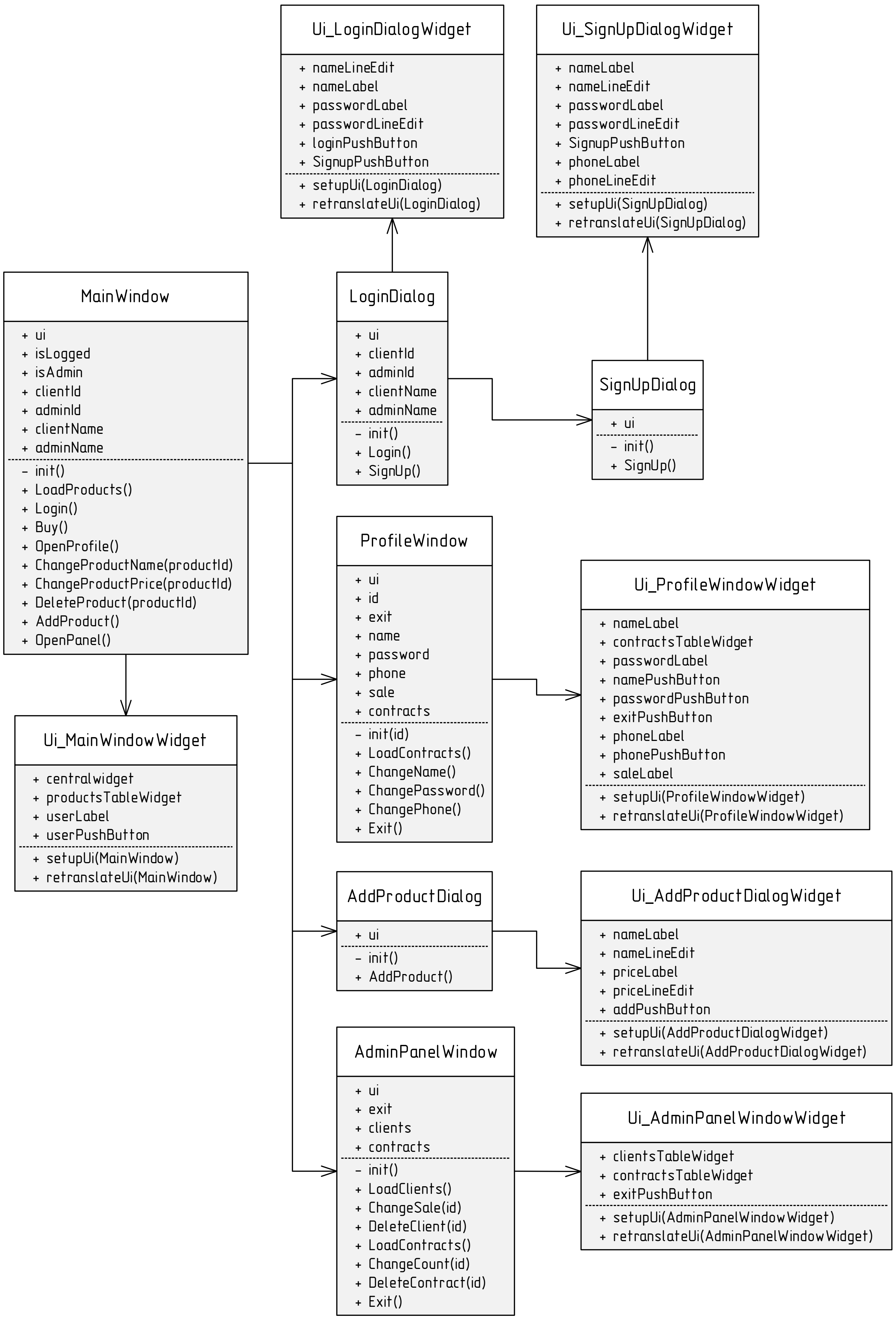


Рис. 2. Структура классов

Листинг:

Automap.py

```
from contextlib import contextmanager
from sqlalchemy.ext.automap import automap_base
from sqlalchemy import create_engine
from sqlalchemy.orm import sessionmaker

engine =
create_engine("postgresql+psycopg2://postgres:2458173671@localhost:5432/postgres
",
                       future=True)
Base = automap_base()

Base.prepare(autoload_with=engine)

Clients = Base.classes.clients
Contracts = Base.classes.contracts
Products = Base.classes.products
Admins = Base.classes.admins

@contextmanager
def get_db_session(engine):
    SessionLocal = sessionmaker(autocommit=False, autoflush=False, bind=engine)
    db = SessionLocal()
    try:
        yield db
    except Exception:
        db.rollback()
        raise
    finally:
        db.close()
```

Connection.py

```
from configparser import ConfigParser
import psycopg2
from sqlalchemy import create_engine

class DatabaseConfig():
    def __init__(self, filename='config.ini', section='postgresql'):
        parser = ConfigParser()
        parser.read(filename)

        self.params = {}
        if parser.has_section(section):
            params = parser.items(section)
            for param in params:
                self.params[param[0]] = param[1]
        else:
            raise Exception(
                'Section {0} not found in the {1} file'.format(section,
filename))

class Connection():
    def __init__(self, config: DatabaseConfig = DatabaseConfig()):
        self.connection: psycopg2.connection = None
        self.cursor: psycopg2.cursor = None
        self.params = config.params
        self.engine = create_engine(

f"postgresql+psycopg2://{self.params['user']}:{self.params['password']}@{self.pa
rams['host']}:5432/{self.params['database']}", pool_size=10, max_overflow=20)

    def set_connection(f):
        def wrapper(*args):
            if args[0].connection is None or args[0].connection.closed:
                args[0].__connect()

            ret = f(*args)

            if args[0].connection is not None or not args[0].connection.closed:
                args[0].__close()
            return ret
        return wrapper

    def __connect(self):
        self.connection = psycopg2.connect(**self.params)
        self.cursor = self.connection.cursor()

    def __close(self):
        if self.cursor is not None:
            self.cursor.close()

        if self.connection is not None:
            self.connection.close()

    @set_connection
    def _execute(self, query: str, params: tuple = ()) -> str:
        if query.startswith("SELECT"):
            self.cursor.execute(query, params)
            data = self.cursor.fetchall()
            return data
        elif query.startswith(("UPDATE", "INSERT", "DELETE")):
            self.cursor.execute(query, params)
            self.connection.commit()

    @set_connection
    def _commit(self):
        self.connection.commit()

    @set_connection
    def _exists(self, query: str, params: tuple = ()):
        self.cursor.execute(query, params)
        return self.cursor.fetchone() is not None

    @set_connection
    def parse_table_names(self):
        self.cursor.execute(
            "SELECT tablename FROM pg_tables WHERE schemaname = 'public'",
            [])

        return tuple(map(lambda x: x[0], self.cursor.fetchall()))

    @set_connection
    def parse_column_names(self, name: str):
        self.cursor.execute(
            "SELECT COLUMN_NAME FROM information_schema.columns WHERE
table_schema = 'public' AND table_name = %s", (
                name, )
        )

        return tuple(map(lambda x: x[0], self.cursor.fetchall()))

    @set_connection
    def parse_table_data(self, name: str):
        self.cursor.execute(
            f"SELECT * FROM {name}", []
        )

        return self.cursor.fetchall()
```

AddProductDialogWidget.py

```
from PyQt5 import QtCore, QtGui, QtWidgets

class Ui_AddProductDialogWidget(object):
    def setupUi(self, AddProductDialogWidget):
        AddProductDialogWidget.setObjectName("AddProductDialogWidget")
        AddProductDialogWidget.resize(400, 173)
        self.nameLabel = QtWidgets.QLabel(AddProductDialogWidget)
        self.nameLabel.setGeometry(QtCore.QRect(20, 20, 111, 41))
        font = QtGui.QFont()
        font.setPointSize(14)
        self.nameLabel.setFont(font)
        self.nameLabel.setObjectName("nameLabel")
        self.nameLineEdit = QtWidgets.QLineEdit(AddProductDialogWidget)
        self.nameLineEdit.setGeometry(QtCore.QRect(150, 20, 231, 41))
        font = QtGui.QFont()
        font.setPointSize(14)
        self.nameLineEdit.setFont(font)
        self.nameLineEdit.setObjectName("nameLineEdit")
        self.priceLabel = QtWidgets.QLabel(AddProductDialogWidget)
        self.priceLabel.setGeometry(QtCore.QRect(20, 70, 111, 41))
        font = QtGui.QFont()
        font.setPointSize(14)
        self.priceLabel.setFont(font)
        self.priceLabel.setObjectName("priceLabel")
        self.priceLineEdit = QtWidgets.QLineEdit(AddProductDialogWidget)
        self.priceLineEdit.setGeometry(QtCore.QRect(150, 70, 231, 41))
        font = QtGui.QFont()
        font.setPointSize(14)
        self.priceLineEdit.setFont(font)
        self.priceLineEdit.setObjectName("priceLineEdit")
        self.addPushButton = QtWidgets.QPushButton(AddProductDialogWidget)
        self.addPushButton.setGeometry(QtCore.QRect(150, 120, 231, 41))
        font = QtGui.QFont()
        font.setPointSize(14)
        self.addPushButton.setFont(font)
        self.addPushButton.setObjectName("addPushButton")

        self.retranslateUi(AddProductDialogWidget)
        QtCore.QMetaObject.connectSlotsByName(AddProductDialogWidget)

    def retranslateUi(self, AddProductDialogWidget):
        _translate = QtCore.QCoreApplication.translate

AddProductDialogWidget.setWindowTitle(_translate("AddProductDialogWidget",
"Добавление товара"))
        self.nameLabel.setText(_translate("AddProductDialogWidget", "Название"))
        self.priceLabel.setText(_translate("AddProductDialogWidget", "Цена,
р."))
        self.addPushButton.setText(_translate("AddProductDialogWidget",
"Создать"))
```

AdminPanelWindowWidget.py

```
from PyQt5 import QtCore, QtGui, QtWidgets

class Ui_AdminPanelWindowWidget(object):
    def setupUi(self, AdminPanelWindowWidget):
        AdminPanelWindowWidget.setObjectName("AdminPanelWindowWidget")
        AdminPanelWindowWidget.resize(650, 484)
        self.clientsTableWidget = QtWidgets.QTableWidget(AdminPanelWindowWidget)
        self.clientsTableWidget.setGeometry(QtCore.QRect(20, 20, 611, 192))
        self.clientsTableWidget.setObjectName("clientsTableWidget")
        self.clientsTableWidget.setColumnCount(0)
        self.clientsTableWidget.setRowCount(0)
        self.clientsTableWidget.verticalHeader().setVisible(False)
        self.contractsTableWidget =
QtWidgets.QTableWidget(AdminPanelWindowWidget)
        self.contractsTableWidget.setGeometry(QtCore.QRect(20, 230, 611, 192))
        self.contractsTableWidget.setObjectName("contractsTableWidget")
        self.contractsTableWidget.setColumnCount(0)
        self.contractsTableWidget.setRowCount(0)
        self.contractsTableWidget.verticalHeader().setVisible(False)
        self.exitPushButton = QtWidgets.QPushButton(AdminPanelWindowWidget)
        self.exitPushButton.setGeometry(QtCore.QRect(510, 430, 121, 41))
        font = QtGui.QFont()
        font.setPointSize(14)
        self.exitPushButton.setFont(font)
        self.exitPushButton.setObjectName("exitPushButton")

        self.retranslateUi(AdminPanelWindowWidget)
        QtCore.QMetaObject.connectSlotsByName(AdminPanelWindowWidget)

    def retranslateUi(self, AdminPanelWindowWidget):
        _translate = QtCore.QCoreApplication.translate

AdminPanelWindowWidget.setWindowTitle(_translate("AdminPanelWindowWidget",
"Панель админа"))
        self.exitPushButton.setText(_translate("AdminPanelWindowWidget",
"Выйти"))
```

LoginDialogWidget.py

```
from PyQt5 import QtCore, QtGui, QtWidgets

class Ui_LoginDialogWidget(object):
    def setupUi(self, LoginDialog):
        LoginDialog.setObjectName("LoginDialog")
        LoginDialog.resize(400, 254)
        font = QtGui.QFont()
        font.setPointSize(14)
        self.nameLineEdit = QtWidgets.QLineEdit(LoginDialog)
        self.nameLineEdit.setFont(font)
        self.nameLineEdit.setGeometry(QtCore.QRect(130, 20, 251, 41))
        self.nameLineEdit.setObjectName("nameLineEdit")
        self.nameLabel = QtWidgets.QLabel(LoginDialog)
        self.nameLabel.setGeometry(QtCore.QRect(20, 20, 61, 41))
        font = QtGui.QFont()
        font.setPointSize(14)
        self.nameLabel.setFont(font)
        self.nameLabel.setObjectName("nameLabel")
        self.passwordLabel = QtWidgets.QLabel(LoginDialog)
        self.passwordLabel.setGeometry(QtCore.QRect(20, 80, 101, 41))
        font = QtGui.QFont()
        font.setPointSize(14)
        self.passwordLabel.setFont(font)
        self.passwordLabel.setObjectName("passwordLabel")
        self.passwordLineEdit = QtWidgets.QLineEdit(LoginDialog)
        self.passwordLineEdit.setFont(font)
        self.passwordLineEdit.setGeometry(QtCore.QRect(130, 80, 251, 41))
        self.passwordLineEdit.setObjectName("passwordLineEdit")
        self.loginPushButton = QtWidgets.QPushButton(LoginDialog)
        self.loginPushButton.setGeometry(QtCore.QRect(22, 140, 361, 41))
        font = QtGui.QFont()
        font.setPointSize(14)
        self.loginPushButton.setFont(font)
        self.loginPushButton.setObjectName("loginPushButton")
        self.SignupPushButton = QtWidgets.QPushButton(LoginDialog)
        self.SignupPushButton.setGeometry(QtCore.QRect(20, 200, 361, 41))
        font = QtGui.QFont()
        font.setPointSize(14)
        self.SignupPushButton.setFont(font)
        self.SignupPushButton.setObjectName("SignupPushButton")

        self.retranslateUi(LoginDialog)
        QtCore.QMetaObject.connectSlotsByName(LoginDialog)

    def retranslateUi(self, LoginDialog):
        _translate = QtCore.QCoreApplication.translate
        LoginDialog.setWindowTitle(_translate("LoginDialog", "Авторизация"))
        self.nameLabel.setText(_translate("LoginDialog", "Name"))
        self.passwordLabel.setText(_translate("LoginDialog", "Password"))
        self.loginPushButton.setText(_translate("LoginDialog", "Login"))
        self.SignupPushButton.setText(_translate("LoginDialog", "Sign up"))
```

MainWindowWidget.py

```
from PyQt5 import QtCore, QtGui, QtWidgets

class Ui_MainWindowWidget(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(799, 556)
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.productsTableWidget = QtWidgets.QTableWidget(self.centralwidget)
        self.productsTableWidget.setGeometry(QtCore.QRect(20, 70, 751, 461))
        self.productsTableWidget.setObjectName("productsTableWidget")
        self.productsTableWidget.setColumnCount(0)
        self.productsTableWidget.setRowCount(0)
        self.productsTableWidget.verticalHeader().setVisible(False)
        self.userLabel = QtWidgets.QLabel(self.centralwidget)
        self.userLabel.setGeometry(QtCore.QRect(20, 20, 601, 31))
        font = QtGui.QFont()
        font.setPointSize(14)
        self.userLabel.setFont(font)
        self.userLabel.setObjectName("userLabel")
        self.userPushButton = QtWidgets.QPushButton(self.centralwidget)
        self.userPushButton.setGeometry(QtCore.QRect(650, 20, 121, 31))
        self.userPushButton.setObjectName("userPushButton")
        MainWindow.setCentralWidget(self.centralwidget)

        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "Типография"))
        self.userLabel.setText(_translate("MainWindow", "Вы не авторизованы"))
        self.userPushButton.setText(_translate("MainWindow", "Войти"))
```

ProfileWindowWidget.py

```
from PyQt5 import QtCore, QtGui, QtWidgets

class Ui_ProfileWindowWidget(object):
    def setupUi(self, ProfileWindowWidget):
        ProfileWindowWidget.setObjectName("ProfileWindowWidget")
        ProfileWindowWidget.resize(400, 356)
        self.nameLabel = QtWidgets.QLabel(ProfileWindowWidget)
        self.nameLabel.setGeometry(QtCore.QRect(20, 10, 241, 31))
        font = QtGui.QFont()
        font.setPointSize(14)
        self.nameLabel.setFont(font)
        self.nameLabel.setObjectName("nameLabel")
        self.contractsTableWidget = QtWidgets.QTableWidget(ProfileWindowWidget)
        self.contractsTableWidget.setGeometry(QtCore.QRect(20, 140, 361, 151))
        self.contractsTableWidget.setObjectName("contractsTableWidget")
        self.contractsTableWidget.setColumnCount(0)
        self.contractsTableWidget.setRowCount(0)
        self.contractsTableWidget.verticalHeader().setVisible(False)
        self.passwordLabel = QtWidgets.QLabel(ProfileWindowWidget)
        self.passwordLabel.setGeometry(QtCore.QRect(20, 50, 241, 31))
        font = QtGui.QFont()
        font.setPointSize(14)
        self.passwordLabel.setFont(font)
        self.passwordLabel.setObjectName("passwordLabel")
        self.namePushButton = QtWidgets.QPushButton(ProfileWindowWidget)
        self.namePushButton.setGeometry(QtCore.QRect(290, 10, 93, 31))
        self.namePushButton.setObjectName("namePushButton")
        self.passwordPushButton = QtWidgets.QPushButton(ProfileWindowWidget)
        self.passwordPushButton.setGeometry(QtCore.QRect(290, 50, 93, 31))
        self.passwordPushButton.setObjectName("passwordPushButton")
        self.exitPushButton = QtWidgets.QPushButton(ProfileWindowWidget)
        self.exitPushButton.setGeometry(QtCore.QRect(290, 310, 93, 31))
        self.exitPushButton.setObjectName("exitPushButton")
        self.phoneLabel = QtWidgets.QLabel(ProfileWindowWidget)
        self.phoneLabel.setGeometry(QtCore.QRect(20, 90, 241, 31))
        font = QtGui.QFont()
        font.setPointSize(14)
        self.phoneLabel.setFont(font)
        self.phoneLabel.setObjectName("phoneLabel")
        self.phonePushButton = QtWidgets.QPushButton(ProfileWindowWidget)
        self.phonePushButton.setGeometry(QtCore.QRect(290, 90, 93, 31))
        self.phonePushButton.setObjectName("phonePushButton")
        self.saleLabel = QtWidgets.QLabel(ProfileWindowWidget)
        self.saleLabel.setGeometry(QtCore.QRect(20, 310, 241, 31))
        font = QtGui.QFont()
        font.setPointSize(14)
        self.saleLabel.setFont(font)
        self.saleLabel.setObjectName("saleLabel")

        self.retranslateUi(ProfileWindowWidget)
        QtCore.QMetaObject.connectSlotsByName(ProfileWindowWidget)

    def retranslateUi(self, ProfileWindowWidget):
        _translate = QtCore.QCoreApplication.translate
        ProfileWindowWidget.setWindowTitle(_translate("ProfileWindowWidget",
"Профиль"))
        self.nameLabel.setText(_translate("ProfileWindowWidget", "Name"))
        self.passwordLabel.setText(_translate("ProfileWindowWidget",
"Password"))
        self.namePushButton.setText(_translate("ProfileWindowWidget",
"Изменить"))
        self.passwordPushButton.setText(_translate("ProfileWindowWidget",
"Изменить"))
        self.exitPushButton.setText(_translate("ProfileWindowWidget", "Выйти"))
        self.phoneLabel.setText(_translate("ProfileWindowWidget", "Phone"))
        self.phonePushButton.setText(_translate("ProfileWindowWidget",
"Изменить"))
        self.saleLabel.setText(_translate("ProfileWindowWidget", "Sale"))
```

SignUpDialogWidget.py

```
from PyQt5 import QtCore, QtGui, QtWidgets

class Ui_SignUpDialogWidget(object):
    def setupUi(self, SignUpDialog):
        SignUpDialog.setObjectName("SignUpDialog")
        SignUpDialog.resize(400, 256)
        self.nameLabel = QtWidgets.QLabel(SignUpDialog)
        self.nameLabel.setGeometry(QtCore.QRect(20, 20, 61, 41))
        font = QtGui.QFont()
        font.setPointSize(14)
        self.nameLabel.setFont(font)
        self.nameLabel.setObjectName("nameLabel")
        self.nameLineEdit = QtWidgets.QLineEdit(SignUpDialog)
        self.nameLineEdit.setFont(font)
        self.nameLineEdit.setGeometry(QtCore.QRect(130, 20, 251, 41))
        self.nameLineEdit.setObjectName("nameLineEdit")
        self.passwordLabel = QtWidgets.QLabel(SignUpDialog)
        self.passwordLabel.setGeometry(QtCore.QRect(20, 80, 101, 41))
        font = QtGui.QFont()
        font.setPointSize(14)
        self.passwordLabel.setFont(font)
        self.passwordLabel.setObjectName("passwordLabel")
        self.passwordLineEdit = QtWidgets.QLineEdit(SignUpDialog)
        self.passwordLineEdit.setGeometry(QtCore.QRect(130, 80, 251, 41))
        self.passwordLineEdit.setObjectName("passwordLineEdit")
        self.passwordLineEdit.setFont(font)
        self.SignupPushButton = QtWidgets.QPushButton(SignUpDialog)
        self.SignupPushButton.setGeometry(QtCore.QRect(20, 200, 361, 41))
        font = QtGui.QFont()
        font.setPointSize(14)
        self.SignupPushButton.setFont(font)
        self.SignupPushButton.setObjectName("SignupPushButton")
        self.phoneLabel = QtWidgets.QLabel(SignUpDialog)
        self.phoneLabel.setGeometry(QtCore.QRect(20, 140, 101, 41))
        font = QtGui.QFont()
        font.setPointSize(14)
        self.phoneLabel.setFont(font)
        self.phoneLabel.setObjectName("phoneLabel")
        self.phoneLineEdit = QtWidgets.QLineEdit(SignUpDialog)
        self.phoneLineEdit.setFont(font)
        self.phoneLineEdit.setGeometry(QtCore.QRect(130, 140, 251, 41))
        self.phoneLineEdit.setObjectName("phoneLineEdit")

        self.retranslateUi(SignUpDialog)
        QtCore.QMetaObject.connectSlotsByName(SignUpDialog)

    def retranslateUi(self, SignUpDialog):
        _translate = QtCore.QCoreApplication.translate
        SignUpDialog.setWindowTitle(_translate("SignUpDialog", "Регистрация"))
        self.nameLabel.setText(_translate("SignUpDialog", "Name"))
        self.passwordLabel.setText(_translate("SignUpDialog", "Password"))
        self.SignupPushButton.setText(_translate("SignUpDialog", "Sign up"))
        self.phoneLabel.setText(_translate("SignUpDialog", "Phone"))
```

AddProductDialog.py

```
from PyQt5 import QtWidgets, QtCore
from Widgets.AddProductDialogWidget import Ui_AddProductDialogWidget
from Db.Models.Automap import engine, Products
from sqlalchemy import insert

class AddProductDialog(QtWidgets.QDialog):
    def __init__(self):
        super(AddProductDialog, self).__init__()
        self.ui = Ui_AddProductDialogWidget()
        self.ui.setupUi(self)
        self.setModal(True)
        self.setWindowFlags(self.windowFlags() &
~QtCore.Qt.WindowContextHelpButtonHint)

        self.ui.addPushButton.clicked.connect(self.AddProduct)

    def AddProduct(self):
        statement = insert(Products).values(name=self.ui.nameLineEdit.text(),
price=self.ui.priceLineEdit.text())
        with engine.connect() as connect:
            connect.execute(statement)
            connect.commit()

        self.accept()
```

AdminPanelWindow.py

```
from PyQt5 import QtWidgets, QtCore
from Widgets.AdminPanelWindowWidget import Ui_AdminPanelWindowWidget
from Db.Models.Automap import get_db_session, engine, Clients, Contracts,
Products
from PyQt5.QtWidgets import QTableWidgetItem, QPushButton, QHeaderView,
QInputDialog

class AdminPanelWindow(QtWidgets.QDialog):
    def __init__(self):
        super(AdminPanelWindow, self).__init__()
        self.ui = Ui_AdminPanelWindowWidget()
        self.ui.setupUi(self)
        self.setModal(True)
        self.setWindowFlags(self.windowFlags() &
~QtCore.Qt.WindowContextHelpButtonHint)

        self.exit = False

        self.LoadClients()
        self.LoadContracts()

        self.ui.exitPushButton.clicked.connect(self.Exit)


    def LoadClients(self):
        with get_db_session(engine) as session:
            self.clients = sorted(session.query(Clients).all(), key=lambda x:
x.id)

        self.ui.clientsTableWidget.setColumnCount(5)
        self.ui.clientsTableWidget.setHorizontalHeaderLabels(
            ("Клиент", "Номер", "Скидка", "", "")
        )

        header = self.ui.clientsTableWidget.horizontalHeader()
        header.setSectionResizeMode(0, QHeaderView.Stretch)
        header.setSectionResizeMode(1, QHeaderView.ResizeToContents)
        header.setSectionResizeMode(2, QHeaderView.ResizeToContents)
        header.setSectionResizeMode(3, QHeaderView.ResizeToContents)
        header.setSectionResizeMode(4, QHeaderView.ResizeToContents)

        count = len(self.clients)
        self.ui.clientsTableWidget.setRowCount(count)

        for i in range(count):
            item = QTableWidgetItem(self.clients[i].name)
            self.ui.clientsTableWidget.setItem(i, 0, item)

            item = QTableWidgetItem(str(self.clients[i].phonenumber))
            self.ui.clientsTableWidget.setItem(i, 1, item)
            item = QTableWidgetItem(str(self.clients[i].sale))
            self.ui.clientsTableWidget.setItem(i, 2, item)

            item = QPushButton(self)
            item.setObjectName("EditSaleButton_" + str(self.clients[i].id))
            item.setText("Изменить скидку")
            item.clicked.connect(lambda ignore, id=self.clients[i].id:
self.ChangeSale(id))
            self.ui.clientsTableWidget.setCellWidget(i, 3, item)

            item = QPushButton(self)
            item.setObjectName("DeleteClientButton_" + str(self.clients[i].id))
            item.setText("Удалить")
            item.clicked.connect(lambda ignore, id=self.clients[i].id:
self.DeleteClient(id))
            self.ui.clientsTableWidget.setCellWidget(i, 4, item)

    def ChangeSale(self, id):
        sale, ok = QInputDialog.getText(self, 'Изменение скидки', 'Новая
скидка', flags=QtCore.Qt.WindowCloseButtonHint)

        if ok:
            sale = float(sale)
            with get_db_session(engine) as session:
                session.query(Clients).filter(Clients.id == id).update({
                    "sale": sale
                })
                session.commit()

            self.LoadClients()

    def DeleteClient(self, id):
        with get_db_session(engine) as session:
            session.query(Clients).filter(Clients.id == id).delete()
            session.commit()
        self.LoadClients()

    def LoadContracts(self):
        with get_db_session(engine) as session:
            self.contracts = sorted(session.query(Contracts).all(), key=lambda
x: x.id)

        self.ui.contractsTableWidget.setColumnCount(6)
        self.ui.contractsTableWidget.setHorizontalHeaderLabels(
            ("Клиент", "Товар", "Кол-во", "Сумма, р.", "", "")
        )

        header = self.ui.contractsTableWidget.horizontalHeader()
        header.setSectionResizeMode(0, QHeaderView.Stretch)
        header.setSectionResizeMode(1, QHeaderView.ResizeToContents)
        header.setSectionResizeMode(2, QHeaderView.ResizeToContents)
        header.setSectionResizeMode(3, QHeaderView.ResizeToContents)
        header.setSectionResizeMode(4, QHeaderView.ResizeToContents)
        header.setSectionResizeMode(5, QHeaderView.ResizeToContents)

        count = len(self.contracts)
        self.ui.contractsTableWidget.setRowCount(count)

        for i in range(count):
            with get_db_session(engine) as session:
                client = session.query(Clients).filter(
                    Clients.id == self.contracts[i].clientid).first()
                item = QTableWidgetItem(client.name if client != None else
"None")
                self.ui.contractsTableWidget.setItem(i, 0, item)

            with get_db_session(engine) as session:
                product = session.query(Products).filter(
                    Products.id == self.contracts[i].productid).first()
                item = QTableWidgetItem(product.name if product != None else
"None")
                self.ui.contractsTableWidget.setItem(i, 1, item)

            item = QTableWidgetItem(str(self.contracts[i].count))
            self.ui.contractsTableWidget.setItem(i, 2, item)

            item = QTableWidgetItem(str(self.contracts[i].amount))
            self.ui.contractsTableWidget.setItem(i, 3, item)

            item = QPushButton(self)
            item.setObjectName("EditCountButton_" + str(self.contracts[i].id))
            item.setText("Изменить кол-во")
            item.clicked.connect(lambda ignore, id=self.contracts[i].id:
self.ChangeCount(id))
            self.ui.contractsTableWidget.setCellWidget(i, 4, item)

            item = QPushButton(self)
            item.setObjectName("DeleteContractButton_" +
str(self.contracts[i].id))
            item.setText("Удалить")
            item.clicked.connect(lambda ignore, id=self.contracts[i].id:
self.DeleteContract(id))
            self.ui.contractsTableWidget.setCellWidget(i, 5, item)

    def ChangeCount(self, id):
        count, ok = QInputDialog.getText(self, 'Изменение количества', 'Новое
количество', flags=QtCore.Qt.WindowCloseButtonHint)

        if ok:
            count = int(count)
            with get_db_session(engine) as session:
                session.query(Contracts).filter(Contracts.id == id).update({
                    "count": count
                })
                session.commit()

            self.LoadContracts()

    def DeleteContract(self, id):
        with get_db_session(engine) as session:
            session.query(Contracts).filter(Contracts.id == id).delete()
            session.commit()
        self.LoadContracts()

    def Exit(self):
        self.exit = True
        self.close()
```

LoginDialog.py

```
from PyQt5 import QtWidgets, QtCore
from Widgets.LoginDialogWidget import Ui_LoginDialogWidget
from Windows.SignUpDialog import SignUpDialog
from Db.Models.Automap import get_db_session, engine, Clients, Admins

class LoginDialog(QtWidgets.QDialog):
    def __init__(self):
        super(LoginDialog, self).__init__()
        self.ui = Ui_LoginDialogWidget()
        self.ui.setupUi(self)
        self.setModal(True)
        self.setWindowFlags(self.windowFlags() &
~QtCore.Qt.WindowContextHelpButtonHint)

        self.clientId = None
        self.adminId = None

        self.ui.loginPushButton.clicked.connect(self.Login)
        self.ui.SignupPushButton.clicked.connect(self.SignUp)

    def Login(self):
        with get_db_session(engine) as session:
            client = session.query(Clients).filter(
                Clients.name == self.ui.nameLineEdit.text()).first()
        with get_db_session(engine) as session:
            admin = session.query(Admins).filter(
                Admins.name == self.ui.nameLineEdit.text()).first()
        if (client == None and admin == None):
            self.ui.nameLineEdit.setStyleSheet("QLineEdit { background-color:
red; }")
        else:
            if client != None and client.password ==
self.ui.passwordLineEdit.text():
                self.clientId = client.id
                self.clientName = client.name
                self.accept()
            elif admin.password == self.ui.passwordLineEdit.text():
                self.adminId = admin.id
                self.adminName = admin.name
                self.accept()
            else:
                self.ui.nameLineEdit.setStyleSheet("QLineEdit { background-
color: white; }")
                self.ui.passwordLineEdit.setStyleSheet("QLineEdit { background-
color: red; }")

    def SignUp(self):
        signUpDialog = SignUpDialog()
        signUpDialog.exec_()
```

MainWindow.py

```
from PyQt5 import QtWidgets, QtCore
from Widgets.MainWindowWidget import Ui_MainWindowWidget
from PyQt5.QtWidgets import QTableWidgetItem, QPushButton, QSpinBox,
QMessageBox, QInputDialog, QHeaderView
from Db.Models.Automap import get_db_session, engine, Products, Contracts
from Windows.LoginDialog import LoginDialog
from Windows.ProfileWindow import ProfileWindow
from Windows.AddProductDialog import AddProductDialog
from Windows.AdminPanelWindow import AdminPanelWindow

class MainWindow(QtWidgets.QMainWindow):
    def __init__(self):
        super(MainWindow, self).__init__()
        self.ui = Ui_MainWindowWidget()
        self.ui.setupUi(self)

        self.isLogged = False
        self.isAdmin = False

        self.LoadProducts()
        self.ui.userPushButton.clicked.connect(self.Login)

    def LoadProducts(self):
        with get_db_session(engine) as session:
            self.products = sorted(session.query(Products).all(), key=lambda x:
x.id)

        header = self.ui.productsTableWidget.horizontalHeader()

        if self.isLogged:
            if self.isAdmin:
                self.ui.productsTableWidget.clear()
                self.ui.productsTableWidget.setColumnCount(5)
                self.ui.productsTableWidget.setHorizontalHeaderLabels(
                    ("Название", "Цена, р.", "", "", "")
                )
                header.setSectionResizeMode(2, QHeaderView.ResizeToContents)
                header.setSectionResizeMode(3, QHeaderView.ResizeToContents)
                header.setSectionResizeMode(4, QHeaderView.ResizeToContents)
            else:
                self.ui.productsTableWidget.setColumnCount(4)
                self.ui.productsTableWidget.setHorizontalHeaderLabels(
                    ("Название", "Цена, р.", "Кол-во", "")
                )
                header.setSectionResizeMode(2, QHeaderView.ResizeToContents)
                header.setSectionResizeMode(3, QHeaderView.ResizeToContents)
        else:
            self.ui.productsTableWidget.setColumnCount(2)
            self.ui.productsTableWidget.setHorizontalHeaderLabels(
                ("Название", "Цена, р.")
            )

        header.setSectionResizeMode(0, QHeaderView.Stretch)
        header.setSectionResizeMode(1, QHeaderView.ResizeToContents)

        productCount = len(self.products)
        if self.isLogged and self.isAdmin:
            self.ui.productsTableWidget.setRowCount(productCount + 1)
        else:
            self.ui.productsTableWidget.setRowCount(productCount)

        for i in range(productCount):
            item = QTableWidgetItem(str(self.products[i].name))
            self.ui.productsTableWidget.setItem(i, 0, item)

            item = QTableWidgetItem(str(self.products[i].price))
            self.ui.productsTableWidget.setItem(i, 1, item)

            if self.isLogged:
                if self.isAdmin:
                    item = QPushButton(self)
                    item.setObjectName("EditNameButton_" +
str(self.products[i].id))
                    item.setText("Изменить название")
                    item.clicked.connect(lambda ignore, id=self.products[i].id:
self.ChangeProductName(id))
                    self.ui.productsTableWidget.setCellWidget(i, 2, item)

                    item = QPushButton(self)
                    item.setObjectName("EditPriceButton_" +
str(self.products[i].id))
                    item.setText("Изменить цену")
                    item.clicked.connect(lambda ignore, id=self.products[i].id:
self.ChangeProductPrice(id))
                    self.ui.productsTableWidget.setCellWidget(i, 3, item)

                    item = QPushButton(self)
                    item.setObjectName("DeleteButton_" +
str(self.products[i].id))
                    item.setText("Удалить")
                    item.clicked.connect(lambda ignore, id=self.products[i].id:
self.DeleteProduct(id))
                    self.ui.productsTableWidget.setCellWidget(i, 4, item)
                else:
                    spinBox = QSpinBox(self)
                    spinBox.setObjectName("BuySpinBox_" +
str(self.products[i].id))
                    self.ui.productsTableWidget.setCellWidget(i, 2, spinBox)

                    item = QPushButton(self)
                    item.setObjectName("BuyButton_" + str(self.products[i].id))
                    item.setText("Купить")
                    item.clicked.connect(lambda ignore, id=self.products[i].id,
                                         name=self.products[i].name, sb=spinBox:
                                            self.Buy(id, name, sb.text()))
                    self.ui.productsTableWidget.setCellWidget(i, 3, item)

        if self.isLogged and self.isAdmin:
            item = QPushButton(self)
            item.setObjectName("AddButton")
            item.setText("Добавить")
            item.clicked.connect(self.AddProduct)
            self.ui.productsTableWidget.setCellWidget(productCount, 0, item)

    def Login(self):
        loginDialog = LoginDialog()
        if loginDialog.exec_() == QtWidgets.QDialog.Accepted:
            self.isLogged = True

            self.clientId = loginDialog.clientId
            self.adminId = loginDialog.adminId

            if self.clientId != None:
                self.clientName = loginDialog.clientName

                self.ui.userLabel.setText("Вы вошли как: " + self.clientName)
                self.LoadProducts()

                self.ui.userPushButton.setText("Личный кабинет")
                self.ui.userPushButton.clicked.disconnect(self.Login)
                self.ui.userPushButton.clicked.connect(self.OpenProfile)
            else:
                self.adminName = loginDialog.adminName
                self.isAdmin = True

                self.ui.userLabel.setText("Вы вошли как админ: " +
self.adminName)
                self.LoadProducts()
                self.ui.userPushButton.setText("Панель")
                self.ui.userPushButton.clicked.disconnect(self.Login)
                self.ui.userPushButton.clicked.connect(self.OpenPanel)

    def Buy(self, id, name, count):
        buyAcceptanceMessageBox = QMessageBox()
        buyAcceptanceMessageBox.setIcon(QMessageBox.Question)
        buyAcceptanceMessageBox.setWindowTitle("Подтверждение покупки")
        buyAcceptanceMessageBox.setText("Купить " + name + " в количестве " +
count + "?")
        buyAcceptanceMessageBox.setStandardButtons(QMessageBox.Yes |
QMessageBox.No)
        result = buyAcceptanceMessageBox.exec()

        if result == QMessageBox.Yes:
            with get_db_session(engine) as session:
                contract = Contracts(clientid=self.clientId, productid=id,
count=count)
                session.add(contract)
                session.commit()

    def OpenProfile(self):
        profileWindow = ProfileWindow(self.clientId)
        profileWindow.exec_()
        if profileWindow.name != self.clientName:
            self.clientName = profileWindow.name
            self.ui.userLabel.setText("Вы вошли как: " + self.clientName)

        if profileWindow.exit:
            self.clientId = None
            self.clientName = None
            self.isLogged = False
            self.ui.userLabel.setText("Вы не авторизованы")
            self.ui.userPushButton.setText("Войти")
            self.ui.userPushButton.clicked.connect(self.Login)
            self.ui.userPushButton.clicked.disconnect(self.OpenProfile)
            self.LoadProducts()

    def ChangeProductName(self, productId):
        name, ok = QInputDialog.getText(self, 'Изменение названия', 'Новое
название', flags=QtCore.Qt.WindowCloseButtonHint)

        if ok:
            with get_db_session(engine) as session:
                session.query(Products).filter(Products.id ==
productId).update({
                    "name": name
                })
                session.commit()

            self.LoadProducts()
    def ChangeProductPrice(self, productId):
        price, ok = QInputDialog.getText(self, 'Изменение цены', 'Новая цена',
flags=QtCore.Qt.WindowCloseButtonHint)

        if ok:
            price = float(price)
            with get_db_session(engine) as session:
                session.query(Products).filter(Products.id ==
productId).update({
                    "price": price
                })
                session.commit()

            self.LoadProducts()

    def DeleteProduct(self, productId):
        with get_db_session(engine) as session:
            session.query(Products).filter(Products.id == productId).delete()
            session.commit()
        self.LoadProducts()

    def AddProduct(self):
        addProductDialog = AddProductDialog()
        if addProductDialog.exec_() == QtWidgets.QDialog.Accepted:
            self.LoadProducts()

    def OpenPanel(self):
        adminPanelWindow = AdminPanelWindow()
        adminPanelWindow.exec_()

        if adminPanelWindow.exit:
            self.adminId = None
            self.adminName = None
            self.isLogged = False
            self.isAdmin = False
            self.ui.userLabel.setText("Вы не авторизованы")
            self.ui.userPushButton.setText("Войти")
            self.ui.userPushButton.clicked.connect(self.Login)
            self.ui.userPushButton.clicked.disconnect(self.OpenPanel)
            self.LoadProducts()
```

ProfileWindow.py

```
from PyQt5 import QtWidgets, QtCore
from Widgets.ProfileWindowWidget import Ui_ProfileWindowWidget
from Db.Models.Automap import get_db_session, engine, Clients, Contracts,
Products, Admins
from PyQt5.QtWidgets import QTableWidgetItem, QMessageBox, QHeaderView,
QInputDialog

class ProfileWindow(QtWidgets.QDialog):
    def __init__(self, id):
        super(ProfileWindow, self).__init__()
        self.ui = Ui_ProfileWindowWidget()
        self.ui.setupUi(self)
        self.setModal(True)
        self.setWindowFlags(self.windowFlags() &
~QtCore.Qt.WindowContextHelpButtonHint)

        self.id = id
        self.exit = False

        with get_db_session(engine) as session:
            client = session.query(Clients).filter(
                Clients.id == id).first()
            self.name = client.name
            self.password = client.password
            self.phone = client.phonenumber
            self.sale = client.sale

        self.ui.nameLabel.setText(self.name)
        self.ui.passwordLabel.setText(self.password)
        self.ui.phoneLabel.setText(str(self.phone))
        self.ui.saleLabel.setText("Скидка: " + str(self.sale) + "%")

        self.LoadContracts()

        self.ui.namePushButton.clicked.connect(self.ChangeName)
        self.ui.passwordPushButton.clicked.connect(self.ChangePassword)
        self.ui.phonePushButton.clicked.connect(self.ChangePhone)
        self.ui.exitPushButton.clicked.connect(self.Exit)

    def LoadContracts(self):
        with get_db_session(engine) as session:
            self.contracts = session.query(Contracts).filter(Contracts.clientid
== self.id).all()

        self.ui.contractsTableWidget.setColumnCount(3)
        self.ui.contractsTableWidget.setHorizontalHeaderLabels(
            ("Товар", "Кол-во", "Сумма, р.")
        )

        count = len(self.contracts)
        self.ui.contractsTableWidget.setRowCount(count)
        for i in range(count):
            with get_db_session(engine) as session:
                product = session.query(Products).filter(
                    Products.id == self.contracts[i].productid).first()
                item = QTableWidgetItem(product.name if product != None else
"None")
                self.ui.contractsTableWidget.setItem(i, 0, item)

            item = QTableWidgetItem(str(self.contracts[i].count))
            self.ui.contractsTableWidget.setItem(i, 1, item)
```

```
                item = QTableWidgetItem(str(self.contracts[i].amount))
                self.ui.contractsTableWidget.setItem(i, 2, item)

            header = self.ui.contractsTableWidget.horizontalHeader()
            header.setSectionResizeMode(0, QHeaderView.Stretch)
            header.setSectionResizeMode(1, QHeaderView.ResizeToContents)
            header.setSectionResizeMode(2, QHeaderView.ResizeToContents)

        def ChangeName(self):
            name, ok = QInputDialog.getText(self, 'Изменение имени', 'Новое имя',
    flags=QtCore.Qt.WindowCloseButtonHint)

            if ok:
                with get_db_session(engine) as session:
                    client = session.query(Clients).filter(
                        Clients.name == name).first()
                with get_db_session(engine) as session:
                    admin = session.query(Admins).filter(
                        Admins.name == self.ui.nameLineEdit.text()).first()
                if (client != None or admin != None):
                    takenLoginMessageBox = QMessageBox()
                    takenLoginMessageBox.setIcon(QMessageBox.Critical)
                    takenLoginMessageBox.setWindowTitle("Ошибка")
                    takenLoginMessageBox.setText("Введенный логин уже занят")
                    takenLoginMessageBox.setStandardButtons(QMessageBox.Ok)
                    takenLoginMessageBox.exec()
                    return

                with get_db_session(engine) as session:
                    session.query(Clients).filter(Clients.id == self.id).update({
                        "name": name
                    })
                    session.commit()

                self.name = name
                self.ui.nameLabel.setText(name)

        def ChangePassword(self):
            password, ok = QInputDialog.getText(self, 'Изменение пароля', 'Новый
    пароль', flags=QtCore.Qt.WindowCloseButtonHint)
            if ok:
                with get_db_session(engine) as session:
                    session.query(Clients).filter(Clients.id == self.id).update({
                        "password": password
                    })
                    session.commit()

                self.password = password
                self.ui.passwordLabel.setText(password)

        def ChangePhone(self):
```

```
        phone, ok = QInputDialog.getText(self, 'Изменение номера', 'Новый
номер', flags=QtCore.Qt.WindowCloseButtonHint)
        if ok:
            phone = int(phone)
            with get_db_session(engine) as session:
                session.query(Clients).filter(Clients.id == self.id).update({
                    "phonenumber": phone
                })
                session.commit()

            self.phone = phone
            self.ui.phoneLabel.setText(str(phone))

    def Exit(self):
        self.exit = True
        self.close()
```

SignUpDialog.py

```
from PyQt5 import QtWidgets, QtCore
from Widgets.SignUpDialogWidget import Ui_SignUpDialogWidget
from Db.Models.Automap import engine, Clients, get_db_session, Admins
from sqlalchemy import insert

class SignUpDialog(QtWidgets.QDialog):
    def __init__(self):
        super(SignUpDialog, self).__init__()
        self.ui = Ui_SignUpDialogWidget()
        self.ui.setupUi(self)
        self.setModal(True)
        self.setWindowFlags(self.windowFlags() &
~QtCore.Qt.WindowContextHelpButtonHint)

        self.ui.SignupPushButton.clicked.connect(self.SignUp)

    def SignUp(self):
        with get_db_session(engine) as session:
            client = session.query(Clients).filter(
                Clients.name == self.ui.nameLineEdit.text()).first()
        with get_db_session(engine) as session:
            admin = session.query(Admins).filter(
                Admins.name == self.ui.nameLineEdit.text()).first()
        if (client != None or admin != None):
            self.ui.nameLineEdit.setStyleSheet("QLineEdit { background-color:
red; }")
        else:
            statement = insert(Clients).values(name=self.ui.nameLineEdit.text(),
password=self.ui.passwordLineEdit.text(),

 phonenumber=self.ui.phoneLineEdit.text(), sale=0)
            with engine.connect() as connect:
                connect.execute(statement)
                connect.commit()

        self.accept()
```

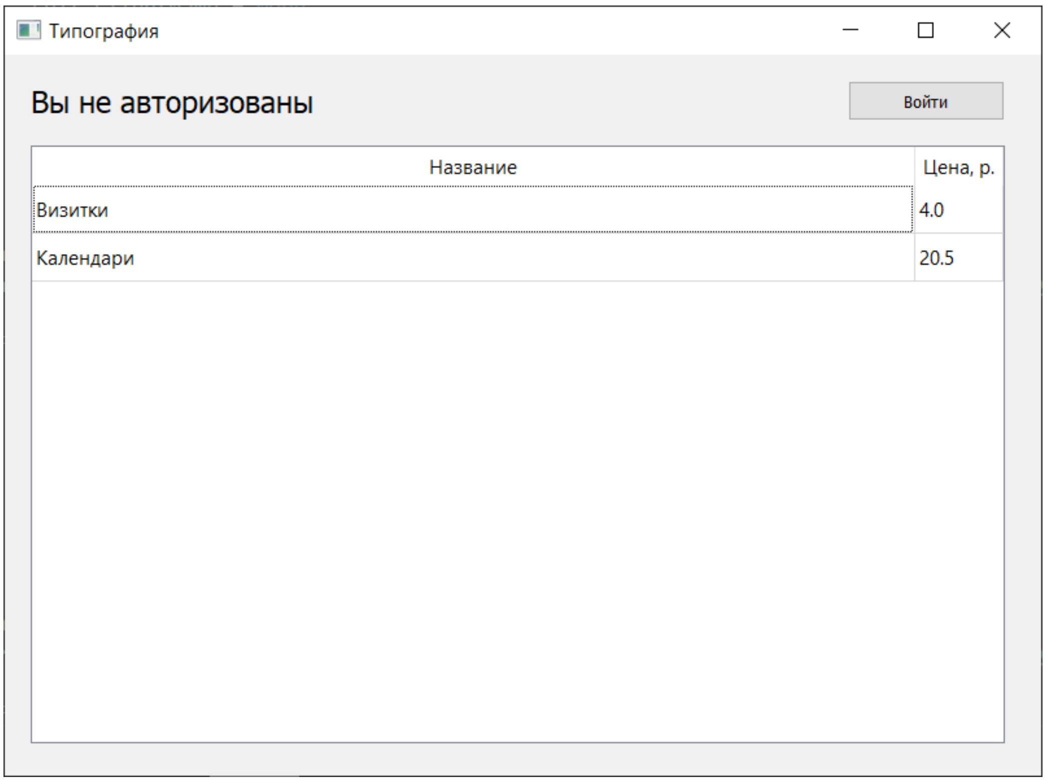
main.py

```
from Windows.MainWindow import MainWindow
from PyQt5 import QtWidgets
import sys

app = QtWidgets.QApplication([])
application = MainWindow()
application.show()

sys.exit(app.exec())
```

Результат:



Вы не авторизованы

Название	Цена, р.
Визитки	4.0
Календари	20.5

Рис. 3. Главное окно без авторизации

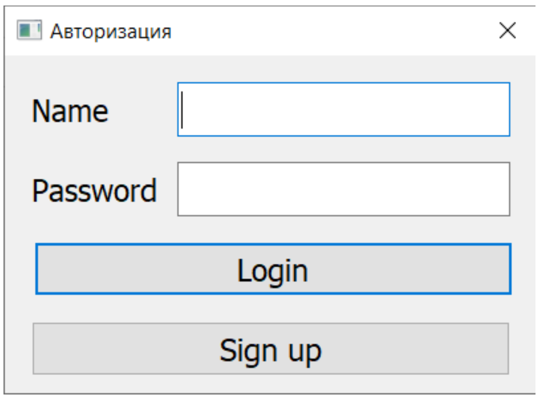


Рис. 4. Окно входа

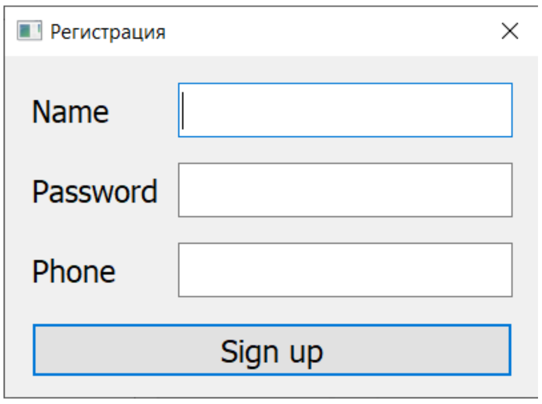
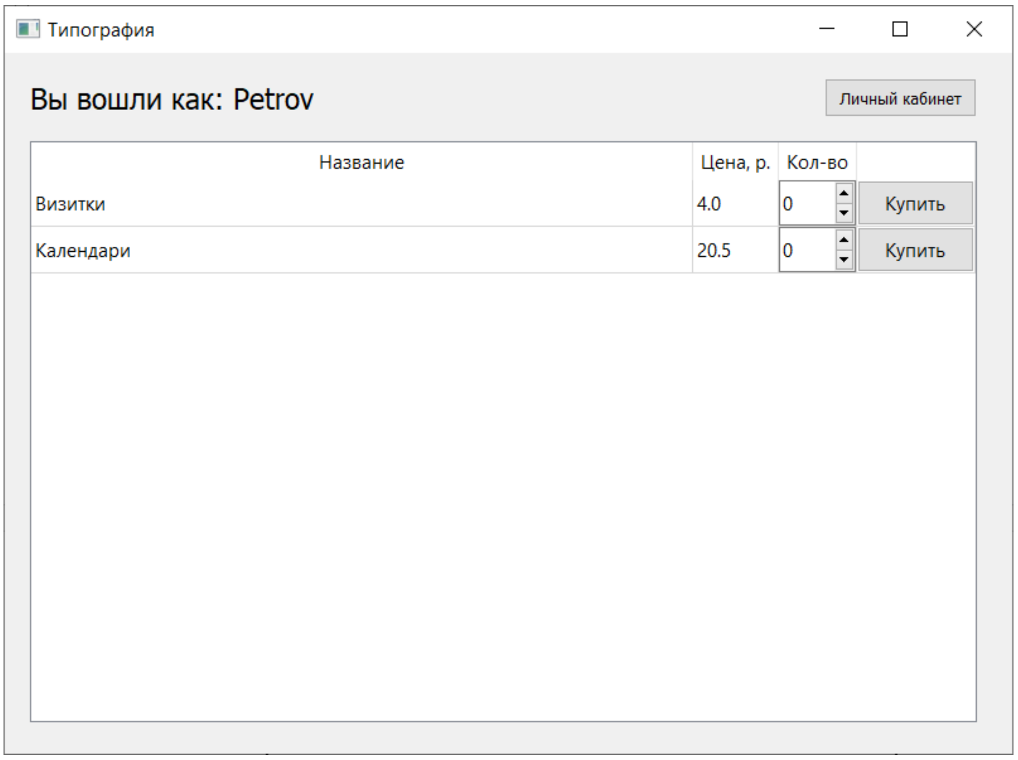


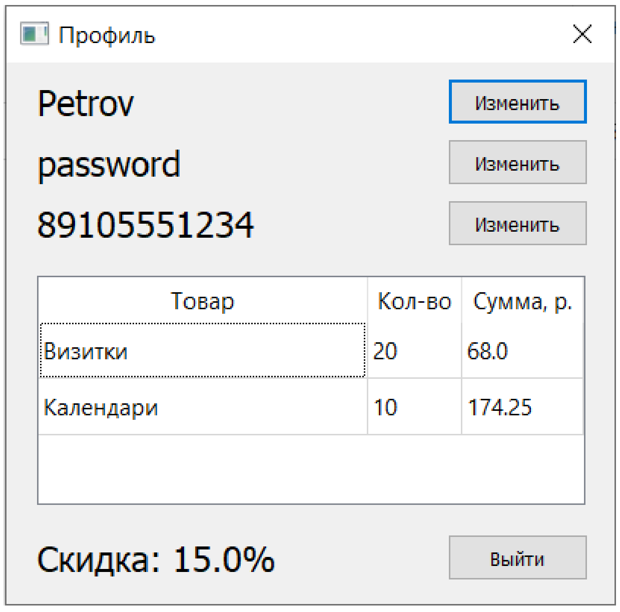
Рис. 5. Окно регистрации



Вы вошли как: Petrov

Название	Цена, р.	Кол-во	
Визитки	4.0	0	Купить
Календари	20.5	0	Купить

Рис. 6. Главное окно для клиента

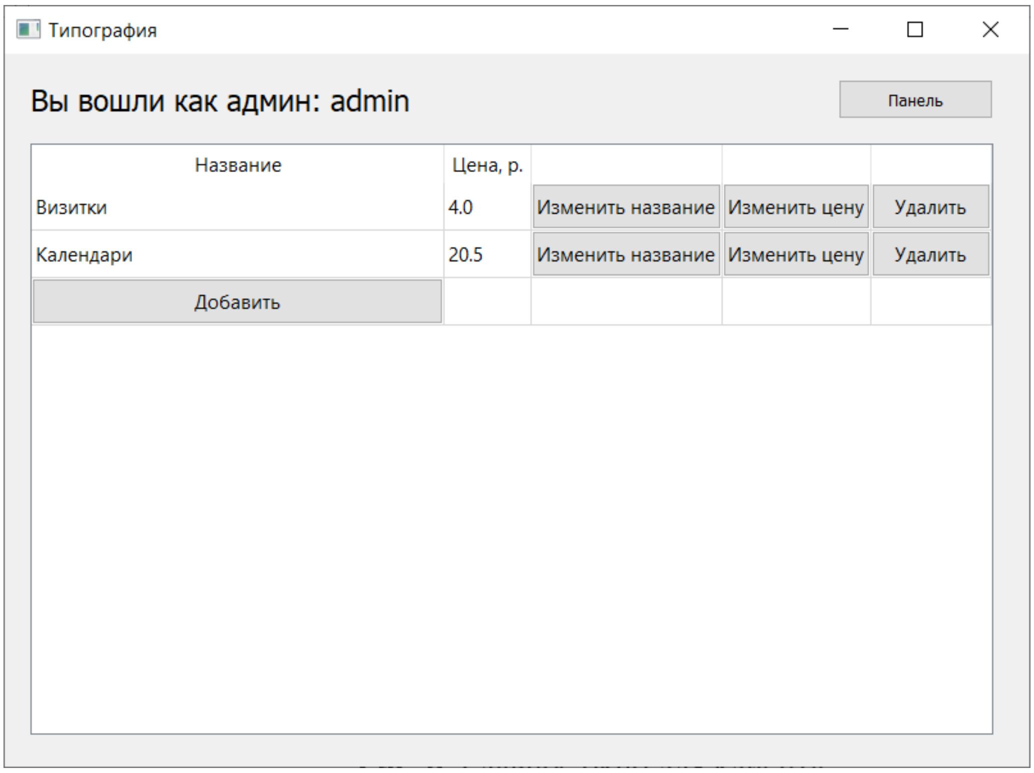


Petrov
password
89105551234

Товар	Кол-во	Сумма, р.
Визитки	20	68.0
Календари	10	174.25

Скидка: 15.0%

Рис. 7. Окно профиля клиента

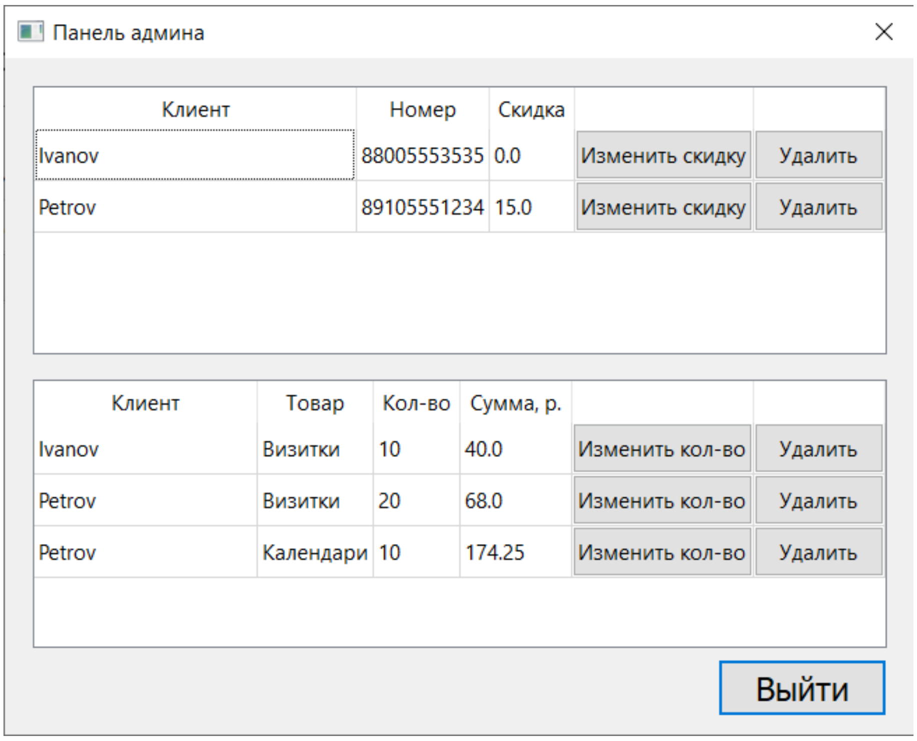


Типография

Вы вошли как админ: admin

Название	Цена, р.			
Визитки	4.0	Изменить название	Изменить цену	Удалить
Календари	20.5	Изменить название	Изменить цену	Удалить
Добавить				

Рис. 8. Главное окно для админа



Панель админа

Клиент	Номер	Скидка		
Ivanov	88005553535	0.0	Изменить скидку	Удалить
Petrov	89105551234	15.0	Изменить скидку	Удалить

Клиент	Товар	Кол-во	Сумма, р.		
Ivanov	Визитки	10	40.0	Изменить кол-во	Удалить
Petrov	Визитки	20	68.0	Изменить кол-во	Удалить
Petrov	Календари	10	174.25	Изменить кол-во	Удалить

Рис. 9. Окно панели админа

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