Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Вятский государственный университет»

Колледж ВятГУ

**ОТЧЕТ**

**ПО ДОМАШНЕЙ КОНТРОЛЬНОЙ РАБОТЕ №9**

**«РАБОТА С ТИПИЗИРОВАННЫМИ ФАЙЛАМИ»**

**ПО МДК 05.02 РАЗРАБОТКА КОДА ИНФОРМАЦИОННЫХ СИСТЕМ**

Выполнил: студент учебной группы

ИСПк-202-52-00

Порубов Кирилл Александрович

Преподаватель:

Сергеева Елизавета Григорьевна

Киров

2023

**Вариант: 1**

**Цель работы:** получить представление об организации хранения и использования информации посредством типизированных файлов, закрепить навыки создания пользовательского интерфейса.

**Задание:**

1. Сформировать структуру записи, с которой будет осуществляться работа, и согласовать ее с преподавателем.

2. Разработать приложение, позволяющее извлекать набор записей из типизированного файла, визуализировать данный набор, а также позволяющее изменять данных и выполнять сохранение проделанных изменений.

**Код программы:**

import sys  
from PyQt5.QtGui import QIcon  
from PyQt5.QtSql import QSqlDatabase, QSqlQuery  
from PyQt5.QtWidgets import QApplication, QMainWindow, QMessageBox, QPushButton, QVBoxLayout, QWidget, QLabel, QLineEdit, QHBoxLayout, QListWidget, QListWidgetItem, QAction, QDialog, QDialogButtonBox  
  
  
class EditUserDialog(QDialog):  
 def \_\_init\_\_(self, user\_id, name, surname, user\_group, parent=None):  
 super().\_\_init\_\_(parent)  
  
 self.user\_id = user\_id  
  
 # Создание меток и полей для ввода имени, фамилии и группы пользователя  
 self.name\_label = QLabel("Имя:")  
 self.name\_input = QLineEdit()  
 self.name\_input.setText(name)  
  
 self.surname\_label = QLabel("Фамилия:")  
 self.surname\_input = QLineEdit()  
 self.surname\_input.setText(surname)  
  
 self.group\_label = QLabel("Группа:")  
 self.group\_input = QLineEdit()  
 self.group\_input.setText(user\_group)  
  
  
 self.setWindowTitle("Редактор")  
  
 # Создание горизонтального компоновщика для меток и полей ввода  
 self.layout = QVBoxLayout()  
 self.layout.addWidget(self.name\_label)  
 self.layout.addWidget(self.name\_input)  
  
 self.layout.addWidget(self.surname\_label)  
 self.layout.addWidget(self.surname\_input)  
  
 self.layout.addWidget(self.group\_label)  
 self.layout.addWidget(self.group\_input)  
  
 # Создание кнопок "Сохранить" и "Отмена"  
 self.button\_box = QDialogButtonBox(QDialogButtonBox.Save | QDialogButtonBox.Cancel)  
 self.button\_box.accepted.connect(self.accept)  
 self.button\_box.rejected.connect(self.reject)  
 self.save\_button = self.button\_box.button(QDialogButtonBox.Save)  
 self.save\_button.setText("Сохранить")  
 self.cancel\_button = self.button\_box.button(QDialogButtonBox.Cancel)  
 self.cancel\_button.setText("Отмена")  
  
  
 self.layout.addWidget(self.button\_box)  
 self.setLayout(self.layout)  
  
 def get\_user\_data(self):  
 name = self.name\_input.text().strip()  
 surname = self.surname\_input.text().strip()  
 user\_group = self.group\_input.text().strip()  
 return name, surname, user\_group  
  
  
class MainWindow(QMainWindow):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
  
 # Инициализация графического интерфейса  
 self.init\_ui()  
  
 # Инициализация базы данных  
 self.init\_db()  
  
 # Создание базы данных  
 self.create\_database()  
  
 def init\_ui(self):  
 # Создание главного виджета  
 self.central\_widget = QWidget()  
 self.setCentralWidget(self.central\_widget)  
  
 self.layout = QVBoxLayout(self.central\_widget)  
  
 # Создание меток и полей для ввода имени, фамилии и группы пользователя  
 self.name\_label = QLabel("Имя студента:")  
 self.name\_input = QLineEdit()  
  
 self.surname\_label = QLabel("Фамилия студента:")  
 self.surname\_input = QLineEdit()  
  
 self.group\_label = QLabel("Группа:")  
 self.group\_input = QLineEdit()  
  
 self.name\_layout = QHBoxLayout()  
 self.name\_layout.addWidget(self.name\_label)  
 self.name\_layout.addWidget(self.name\_input)  
  
 self.name\_layout.addWidget(self.surname\_label)  
 self.name\_layout.addWidget(self.surname\_input)  
  
 self.name\_layout.addWidget(self.group\_label)  
 self.name\_layout.addWidget(self.group\_input)  
  
 self.add\_button = QPushButton("Добавить студента")  
 self.edit\_button = QPushButton("Редактировать студента")  
 self.delete\_button = QPushButton("Отчислить студента")  
  
 self.add\_button.clicked.connect(self.add\_user)  
 self.edit\_button.clicked.connect(self.edit\_user)  
 self.delete\_button.clicked.connect(self.delete\_user)  
  
 # Создание списка пользователей  
 self.user\_list = QListWidget()  
  
 self.layout.addLayout(self.name\_layout)  
 self.layout.addWidget(self.add\_button)  
 self.layout.addWidget(self.edit\_button)  
 self.layout.addWidget(self.user\_list)  
 self.layout.addWidget(self.delete\_button)  
  
 # Создание действия для кнопки "Выход"  
 exit\_action = QAction(QIcon("exit.png"), "Выход", self)  
 exit\_action.setShortcut("Ctrl+Q")  
 exit\_action.triggered.connect(self.close)  
  
 # Создание меню  
 menu\_bar = self.menuBar()  
 file\_menu = menu\_bar.addMenu("Файл")  
 file\_menu.addAction(exit\_action)  
  
 # Установка заголовка окна и размеров  
 self.setWindowTitle("Список студентов")  
 self.setGeometry(100, 100, 400, 300)  
  
 def init\_db(self):  
 # Инициализация базы данных SQLite  
 self.db = QSqlDatabase.addDatabase("QSQLITE")  
 self.db.setDatabaseName("database.db")  
 if not self.db.open():  
 QMessageBox.critical(self, "Ошибка", "Не удалось подключиться к базе данных")  
 sys.exit(1)  
  
 def add\_user(self):  
 # Получение имени, фамилии и группы пользователя из полей ввода  
 name = self.name\_input.text().strip()  
 surname = self.surname\_input.text().strip()  
 user\_group = self.group\_input.text().strip()  
  
 if name and surname and user\_group:  
 query = QSqlQuery()  
 query.prepare("INSERT INTO users (name, surname, user\_group) VALUES (:name, :surname, :user\_group)")  
 query.bindValue(":name", name)  
 query.bindValue(":surname", surname)  
 query.bindValue(":user\_group", user\_group)  
 if query.exec\_():  
 self.name\_input.clear()  
 self.surname\_input.clear()  
 self.group\_input.clear()  
  
 self.show\_users()  
 self.user\_list.sortItems()  
 else:  
 QMessageBox.warning(self, "Предупреждение", "Не удалось добавить студента")  
 else:  
 QMessageBox.warning(self, "Предупреждение", "Введите имя, фамилию и группу студента")  
  
 def edit\_user(self):  
 # Получение выбранного пользователя из списка  
 selected\_item = self.user\_list.currentItem()  
  
 if selected\_item:  
 selected\_id = selected\_item.data(32)  
 query = QSqlQuery()  
 query.prepare("SELECT \* FROM users WHERE id = :id")  
 query.bindValue(":id", selected\_id)  
 if query.exec\_() and query.next():  
 name = query.value(1)  
 surname = query.value(2)  
 user\_group = query.value(3)  
  
 # Создание диалогового окна для редактирования пользователя  
 dialog = EditUserDialog(selected\_id, name, surname, user\_group, parent=self)  
 if dialog.exec\_() == QDialog.Accepted:  
 new\_name, new\_surname, new\_user\_group = dialog.get\_user\_data()  
  
 # Обновление информации о пользователе в базе данных  
 update\_query = QSqlQuery()  
 update\_query.prepare("UPDATE users SET name = :name, surname = :surname, user\_group = :user\_group WHERE id = :id")  
 update\_query.bindValue(":name", new\_name)  
 update\_query.bindValue(":surname", new\_surname)  
 update\_query.bindValue(":user\_group", new\_user\_group)  
 update\_query.bindValue(":id", selected\_id)  
 if update\_query.exec\_():  
 # Обновление информации о пользователях  
 self.show\_users()  
 self.user\_list.sortItems()  
 else:  
 QMessageBox.warning(self, "Предупреждение", "Не удалось обновить информацию о студенте")  
 else:  
 QMessageBox.warning(self, "Предупреждение", "Выберите студента для редактирования")  
  
 def delete\_user(self):  
 # Получение выбранных пользователей из списка  
 selected\_items = self.user\_list.selectedItems()  
  
 if selected\_items:  
 # Удаление выбранных пользователей из базы данных  
 for item in selected\_items:  
 selected\_id = item.data(32)  
 query = QSqlQuery()  
 query.prepare("DELETE FROM users WHERE id = ?")  
 query.addBindValue(selected\_id)  
 query.exec\_()  
  
 # Обновление информации о пользователях  
 self.show\_users()  
 else:  
 QMessageBox.warning(self, "Предупреждение", "Выберите студента для отчисления")  
  
 def show\_users(self):  
 # Очистка списка пользователей  
 self.user\_list.clear()  
  
 # Получение списка пользователей из базы данных  
 query = QSqlQuery()  
 query.exec\_("SELECT \* FROM users ORDER BY user\_group")  
  
 # Отображение информации о пользователях  
 while query.next():  
 id = query.value(0)  
 name = query.value(1)  
 surname = query.value(2)  
 user\_group = query.value(3)  
 item = QListWidgetItem(f"ID: {id}, Имя: {name}, Фамилия: {surname}, Группа: {user\_group}")  
 item.setData(32, id)  
 self.user\_list.addItem(item)  
  
 def create\_database(self):  
 query = QSqlQuery()  
 query.exec\_("""  
 CREATE TABLE IF NOT EXISTS users (  
 id INTEGER PRIMARY KEY,  
 name TEXT,  
 surname TEXT,  
 user\_group TEXT  
 )  
 """)  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 app = QApplication(sys.argv)  
 window = MainWindow()  
 window.show()  
 window.show\_users()  
 sys.exit(app.exec())

**Результаты выполнения программы:**

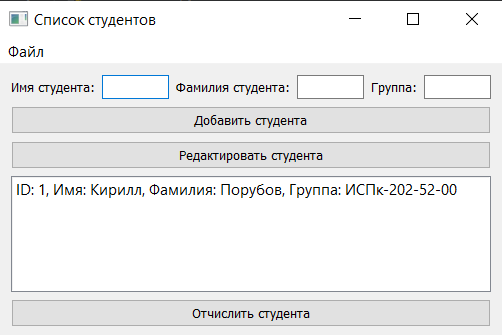
****

Рисунок 1 – База данных «Список студентов»

**Вывод**

В рамках домашней контрольной работы №9 по работе с типизированными файлами была выполнена следующая работа:

Была разработана база данных "Список студентов" на языке программирования Python с использованием библиотеки QtSql и фреймворка PyQt5. Цель работы заключалась в получении представления об организации хранения и использования информации через типизированные файлы, а также закреплении навыков создания пользовательского интерфейса.

В ходе выполнения работы были выполнены следующие задачи:

Была сформирована структура записи, которая использовалась для работы с базой данных. Эта структура была согласована с преподавателем.

Было разработано приложение, которое позволяет извлекать набор записей из типизированного файла, визуализировать данный набор данных и осуществлять изменение данных. Приложение также предоставляет возможность сохранения внесенных изменений.

Работа выполнена в полном соответствии с поставленными задачами и целями. База данных "Список студентов" обеспечивает эффективное хранение информации о студентах и позволяет удобно визуализировать и изменять данные. В результате работы получены необходимые навыки работы с типизированными файлами и создания пользовательского интерфейса.

В целом, выполненная работа успешно соответствует требованиям домашней контрольной работы №9 и целям МДК 05.02 "Разработка кода информационных систем".