ИССЛЕДОВАТЕЛЬСКАЯ РАБОТА

«СОЗДАНИЕ ПРИЛОЖЕНИЯ ПО ВЕДЕНИЮ УЧЕТА УЧАСТНИКОВ СОРЕВНОВАНИЙ НА ЯЗЫКЕ ПРОГРАММИРОВАНИЯ PYTHON С ИСПОЛЬЗОВАНИЕМ СУБД SQLITE»

Выполнил:

Рябов Тимофей Сергеевич,

школа 131

ВВЕДЕНИЕ

Проблема: данные об участнике соревнований заносятся вручную, из-за чего в общей таблице встречаются *ошибки и дубликаты*

Цель исследования: возможности языка **Python** и **СУБД SQL** для решения проблемы внесения и хранения сведений об участниках в базу данных с проверкой их дублирования

Предмет исследования: самостоятельное создание оконного приложения, работающего с **СУБД SQL**

Объект исследования: когнитивные возможности ученика 9 класса в освоении **ООП** и работы с **СУБД**

ВВЕДЕНИЕ

Методы исследования:

- Чтение специальной литературы по **Python** и принципам создания приложений.
- Изучение принципов **ООП** в Python.
- Изучение документации библиотеки **Qt5** для Python.
- Изучение работы **СУБД SQL**, языка запросов СУБД.
- Изучение принципов правильного **построения баз данных**.
- Изучение библиотеки **sqlite3** для связи Python с СУБД.

ПЛАН РЕАЛИЗАЦИИ

Этапы работы:

- Изучение основ **ООП** языка Python
- Изучение основ **СУБД**
- Создание **формы** для приложения
- Изучение языка запросов **SQL** и тренировка в написании запросов к СУБД через библиотеку **SQLite3**
- Связывания графического интерфейса с кодом на Python средствами библиотеки **Qt5**
- Создание **приложения** для работы с БД.

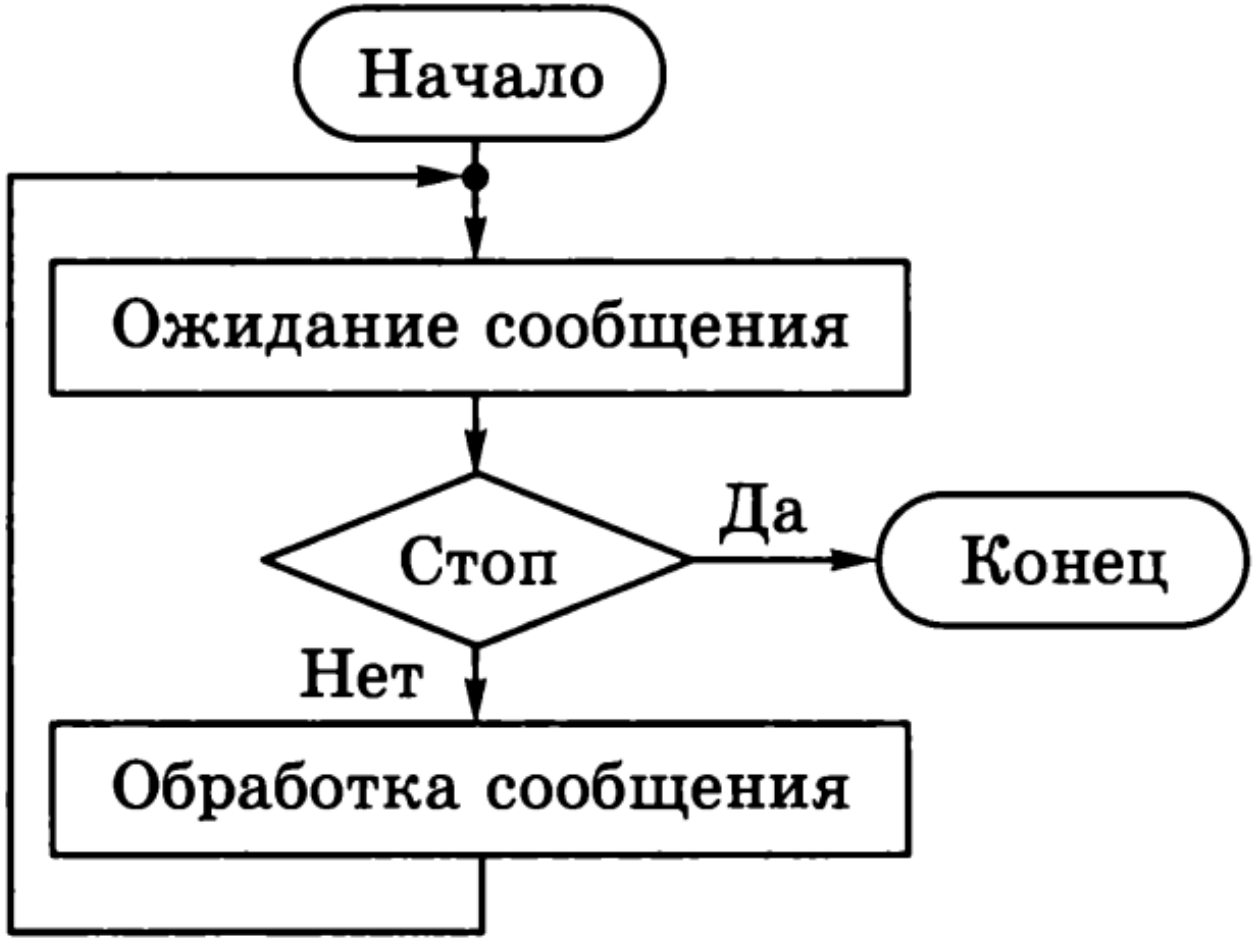
PYTHON ООП

Объект — это данные и методы их обработки.
Класс — это описание множества объектов, имеющих общий набор свойств (данных) и общее поведение (методы).
Интерфейс — это открытые для других объектов свойства и методы.

К трем китам ООП относятся:

- инкапсуляция,
- полиморфизм,
- наследование.

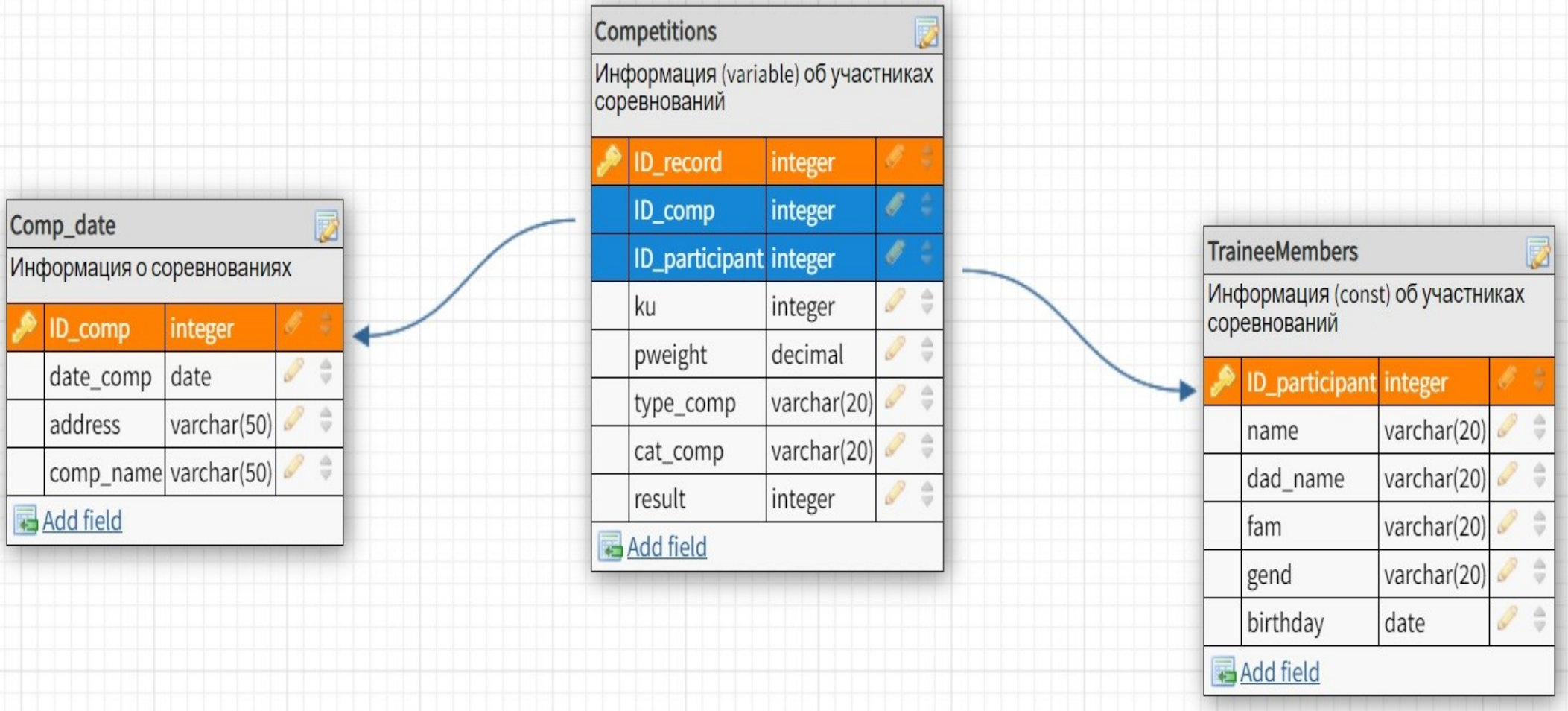
PYTHON ООП


Начало
Ожидание сообщения
Стоп
Да
Конец
Нет
Обработка сообщения

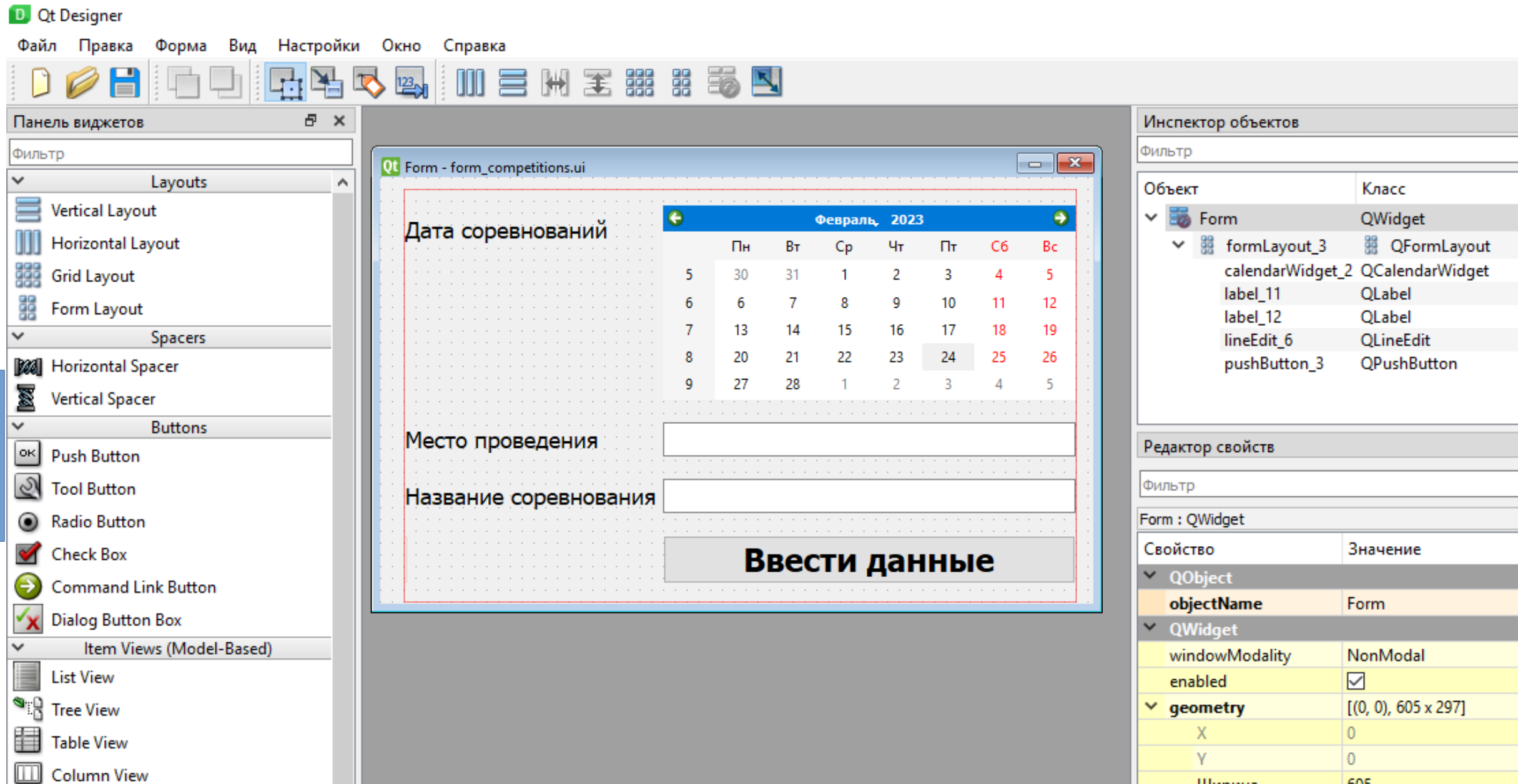
SQL

ER-диаграмма — это схема сущностей (E — entity) и их связей (R – relationship) по первичным и внешним ключам.
Сущность (entity) представляет тип объектов, которые должны храниться в БД.
Атрибут представляет свойство, которое описывает некоторую характеристику объекта.
Первичный ключ — это комбинация атрибутов (столбцов), которые уникально идентифицируют каждую строку таблицы.
Связь (отношение, relationship) представляет ассоциацию между сущностями разных типов

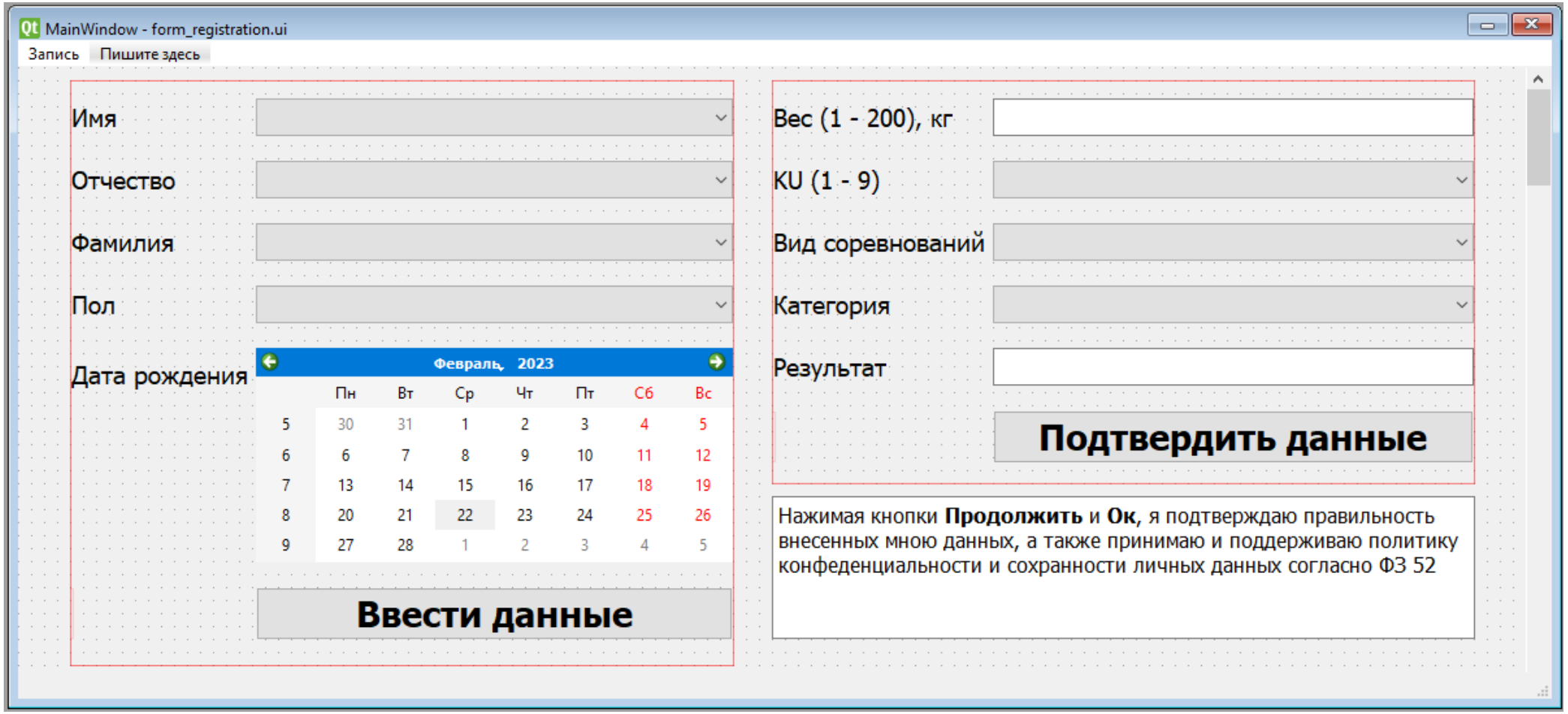
ER-ДИАГРАММА



СОЗДАНИЕ ОКОННОГО ИНТЕРФЕЙСА



СОЗДАНИЕ ОКОННОГО ИНТЕРФЕЙСА



План создания приложения

- Перевод GUI-форм на Python
- Написание главной программы, запускающей первый модуль и далее очередь обработки сообщений.
- Подключение модулей обработчиков событий для каждой формы.
- Внутри каждой формы подключаемся к базе данных.

План создания приложения

- Создаем функцию для внесения новых данных в каждую таблицу.
- Создаем функцию для изменения уже имеющихся данных в каждой таблице.
- Создаем функцию для маркировки дублей для каждой таблицы.
- Создаем интерактивные диалоги с пользователем посредством стандартных окон.

File Edit View Navigate Code Refactor Run Tools VCS Window Help Form_competition_Ryabov_131 - form_registration.py - Administra

Form_competition_Ryabov_131 〉 form_registration.py

Project

Pull Requests

main.py | first_window.py | second_window.py | form_registration.py

```
10
11  from PyQt5 import QtCore, QtGui, QtWidgets
12
13
14  class Ui_MainWindow(object):
15      def setupUi(self, MainWindow):
16          MainWindow.setObjectName("MainWindow")
17          MainWindow.resize(1180, 543)
18          MainWindow.setLayoutDirection(QtCore.Qt.LeftToRight)
19          self.centralwidget = QtWidgets.QWidget(MainWindow)
20          self.centralwidget.setObjectName("centralwidget")
21          self.formLayoutWidget = QtWidgets.QWidget(self.centralwidget)
22          self.formLayoutWidget.setGeometry(QtCore.QRect(40, 10, 511, 451))
23          self.formLayoutWidget.setObjectName("formLayoutWidget")
24          self.formLayout = QtWidgets.QFormLayout(self.formLayoutWidget)
25          self.formLayout.setContentsMargins(0, 0, 0, 0)
```

main.py | first_window.py | second_window.py | form_registration.py

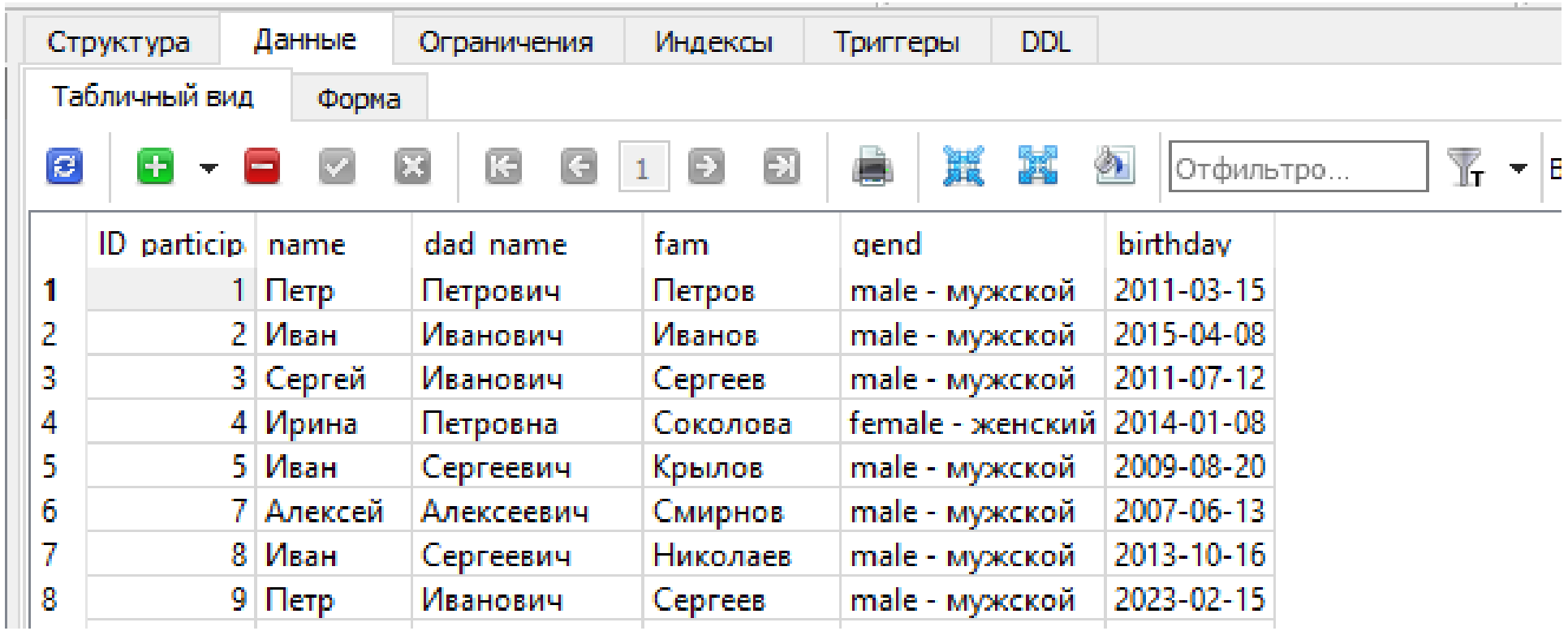
```python
import sqlite3
import sys

import second_window

from PyQt5.QtWidgets import QApplication, QMessageBox, QWidget

from form_competitions import Ui_Form


# Наследуемся от виджета из PyQt5.QtWidgets и от класса с интерфейсом
class MyForm(QWidget, Ui_Form):
    def __init__(self):
        """
        Конструктор класса - инициализация всех свойств класса
        """
        # наследование свойств родителей - классов QMainWindow и Ui_MainW
        super().__init__()
```

main.py | first_window.py | second_window.py | form_registration.py

```python
import sqlite3
import sys

from PyQt5.QtWidgets import QApplication, QMainWindow, QMessageBox, QInputDialog

from form_registration import Ui_MainWindow


# Наследуемся от виджета из PyQt5.QtWidgets и от класса с интерфейсом
class MyWidget(QMainWindow, Ui_MainWindow):
    def __init__(self, id_comp, parent=None):
        """
        Конструктор класса - инициализация всех свойств класса
        """
        # прием данных - ID соревнования
        self.id_comp = id_comp
        # наследование свойств родителей - классов QMainWindow и Ui_MainWindow
        super().__init__()
```

Подключение к СУБД

```
def db_connection(self):
    self.con = sqlite3.connect('db\\new_db.db')
    self.cur = self.con.cursor()
```



	ID particip	name	dad name	fam	gend	birthday
1	1	Петр	Петрович	Петров	male - мужской	2011-03-15
2	2	Иван	Иванович	Иванов	male - мужской	2015-04-08
3	3	Сергей	Иванович	Сергеев	male - мужской	2011-07-12
4	4	Ирина	Петровна	Соколова	female - женский	2014-01-08
5	5	Иван	Сергеевич	Крылов	male - мужской	2009-08-20
6	7	Алексей	Алексеевич	Смирнов	male - мужской	2007-06-13
7	8	Иван	Сергеевич	Николаев	male - мужской	2013-10-16
8	9	Петр	Иванович	Сергеев	male - мужской	2023-02-15

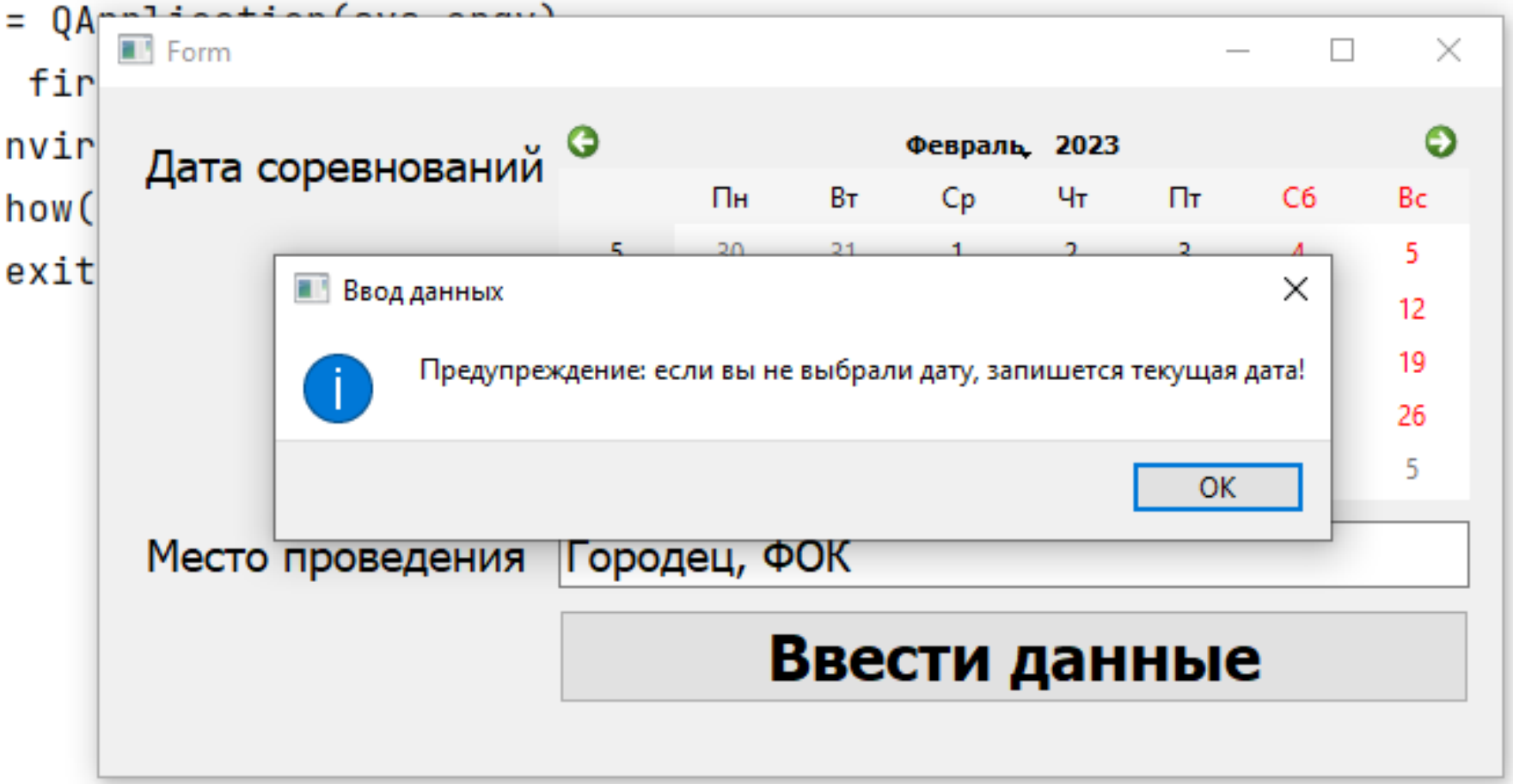
Функция создания новой записи в БД

```
def new_record(self, name=None, dad_name=None, fam=None, gend=None, birthday=None,
               ku=None, pweight=None, type_comp=None, cat_comp=None, result=None, n_table=1, id_comp=None, id=None):
    # метод ввода данных в БД
    # print(name, dad_name, fam, gend, birthday, ku, pweight, type_comp, cat_comp, result, n_table, id_comp, id)
    if all([name, dad_name, fam, gend, birthday]) and n_table == 1:
        self.con.execute("""INSERT INTO TraineeMembers(name, dad_name, fam, gend, birthday)
                            VALUES (?, ?, ?, ?, ?)""",
                         (name, dad_name, fam, gend, birthday))
        self.con.commit()
        res = self.check_record(name=name, dad_name=dad_name, fam=fam, n_table=1)
    elif all([ku, pweight, type_comp, cat_comp, result]) and n_table == 2:
        self.con.execute("""INSERT INTO Competitions(ID_comp, ID_participant, ku, pweight, type_comp, cat_comp, resul
                            VALUES (?, ?, ?, ?, ?, ?, ?)""",
                         (id_comp, id, ku, pweight, type_comp, cat_comp, result))
        self.con.commit()
        res = self.check_record(id_comp=id_comp, id=id, type_comp=type_comp, cat_comp=cat_comp, n_table=2)
    else:
        res = None
        self.mess = 'Упс... Что-то пошло не так...'
        self.message()
    return res
```

Функция для проверки наличия записи

```
def check_record(self, name=None, dad_name=None, fam=None, gend=None, birthday=None, pweight=None, ku=None,
                 type_comp=None, cat_comp=None, result=None, n_table=1, id=None, id_comp=None):
    # метод проверки наличия данных в БД
    # print(name, dad_name, fam, gend, birthday, pweight, ku, type_comp, cat_comp, result, n_table, id, id_comp
    # поиск участника по базе - если есть, то фиксируем ID, если нет, то добавляем в базу новый ID и данные
    if all([name, dad_name, fam]) and n_table == 1:
        res = self.cur.execute("""SELECT ID_participant
                                  FROM TraineeMembers
                                  WHERE fam = ? and name = ? and dad_name = ?""",
                               (fam, name, dad_name)).fetchone()
    elif all([type_comp, cat_comp, id, id_comp]) and n_table == 2:
        res = self.cur.execute("""SELECT ID_record
                                  FROM Competitions
                                  WHERE ID_comp = ? and ID_participant = ? and type_comp = ? and cat_comp = ?"
                               (id_comp, id, type_comp, cat_comp)).fetchone()
    else:
        self.mess = 'Упс... Что-то пошло не так...'
        self.message()
        res = None
    return res
```

Создание интерактива



Заключение

Мы изучили и успешно применили следующие важные разделы Computer Science:

- Теория и практика по проектированию **БД**
- Теория и практика по основам **ООП**, в частности в **Python**
- Изучили документацию библиотек **PyQt5** и **sqlite3**
- Создали на их основе **работающее приложение** по конкретной задаче — учете спортсменов-участников соревнований по каратэ

Заключение

При выполнении данной работы мы смогли на практике доказать нашу гипотезу:

ученик 9 класса имеет **достаточные когнитивные способности** для самостоятельного освоения материала уровня промышленного программирования,

а именно — написания с нуля работающего оконного приложения, в котором реализуются запросы к самостоятельно спроектированной БД.

Заключение

Наше приложение будет **развиваться дальше**. Есть запрос тренера на форму для **аналитики** по хранимым данным.

В качестве продолжения нужно

- изучить тонкости запросов SELECT, группировки, соединения таблиц и отборы по разным условиям,
- разработать универсальный дизайн окна для возможности ввода разнообразных запросов к БД.

Спасибо за внимание!