<table>
<tr><td rowspan="2"> 
Politechnika
Wrocławska</td><td rowspan="2">Technologie Internetowe

Laboratorium</td><td>Poniedziałek 17.05-19.40</td></tr>
<tr><td>Wydział Informatyki i Zarządzania

Kierunek Inżynieria Systemów</td></tr>
<tr><td>Prowadzący:

Mgr inż. Mateusz Magda</td><td>MINI-PROJEKT

„Boulderowe zawody wspinaczkowe”</td><td>Rok studiów III

Semestr V

Rok Akademicki 2017/2018</td></tr>
<tr><td colspan="2"></td><td>Wykonujący zadanie:

Kevin Nikolas

Adrian Podkalicki

Michał Kuriata

Bartek Bojanowski</td></tr>
</table>

1. Opis zadania.

Boulderowe zawody wspinaczkowe – mistrzostwa Polski, składają się z dwóch etapów: eliminacje i finał. Mamy 6 kategorii: kobiety, mężczyźni, juniorki, juniorzy, mastersi, masterki. Wspinaczkowa droga „baldowa" może być zaliczona albo w połowie (wspinacz doszedł na niej do chwytu „bonus", mniej więcej w połowie) albo w całości (top). Jeśli wspinacz dojdzie do końca drogi automatycznie zaliczamy mu zarówno top jak i bonus (bo przechodząc do końca poszedł przez środek).

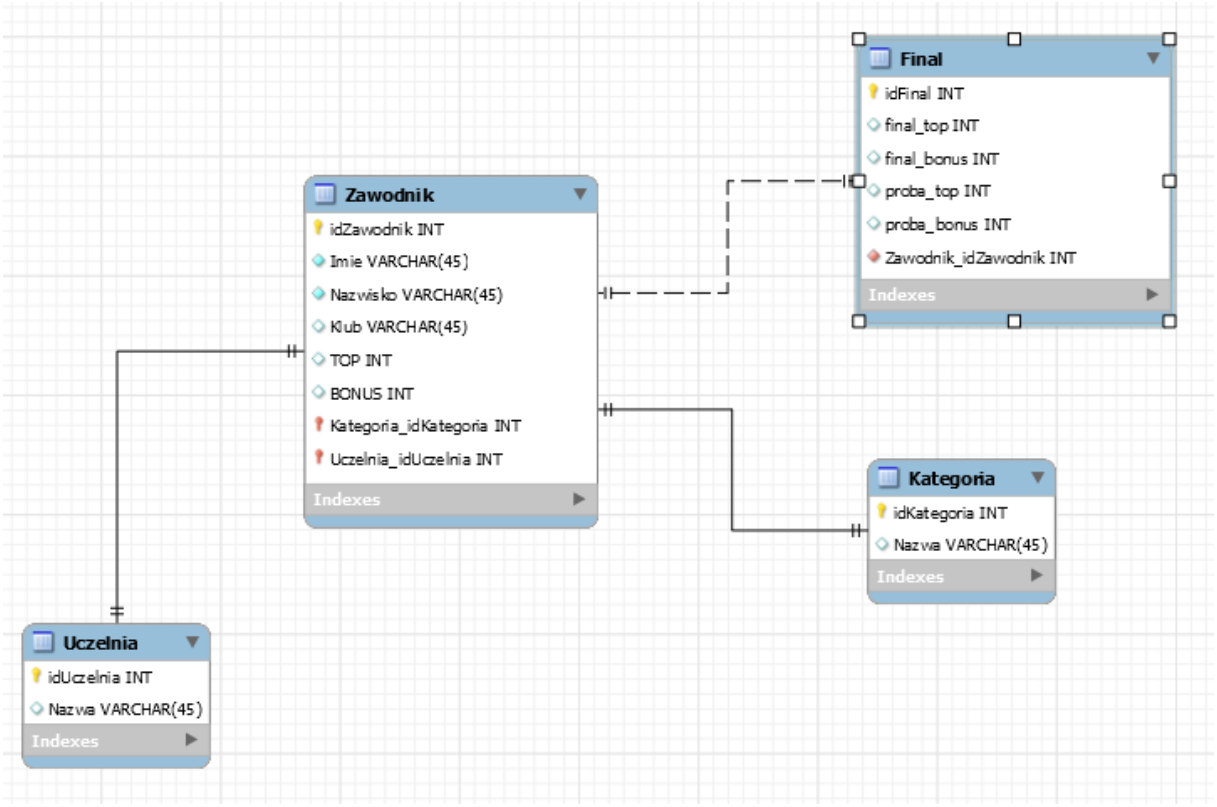
Zaprojektować, stworzyć i przygotować bazę danych oraz wypełnić ją danymi na podstawie załączonego pliku. Przypisując losową liczbę punktów każdemu zawodnikowi, wykonać symulację zawodów i pokazać pełne wyniki (raport miejsc wszystkich zawodników).

Eliminacje – zawodnicy z każdej kategorii startują w czterech grupach (od godziny 8:00 co 2 godziny). Mają 2 godziny na wykonanie jak największej ilości dróg. Pod koniec czasu ostatniej grupy zawodnicy z wszystkich grup są łączeni, dzieleni na kategorie (kobiety/mężczyźni/…) i liczone jest kto dostaje się do finału. Finał rozgrywany jest w kategoriach kobiet i mężczyzn, w pozostałych kategoriach miejsca na podium przydzielane są na podstawie punktów osiągniętych w eliminacjach. Najważniejszym kryterium jest kto ile zrobił dróg (osiągnął topów), a drugim kryterium – kto ile osiągnął bonusów. Do finału dostaje się 6 kobiet i 6 mężczyzn. Dopuszczalna jest sytuacja remisowa – gdy ostatnia osoba ze wszystkich, które dostają się do finału remisuje punktami z poprzednikiem. Wtedy do finału dostaje się tyle osób, ile remisuje na ostatnim miejscu

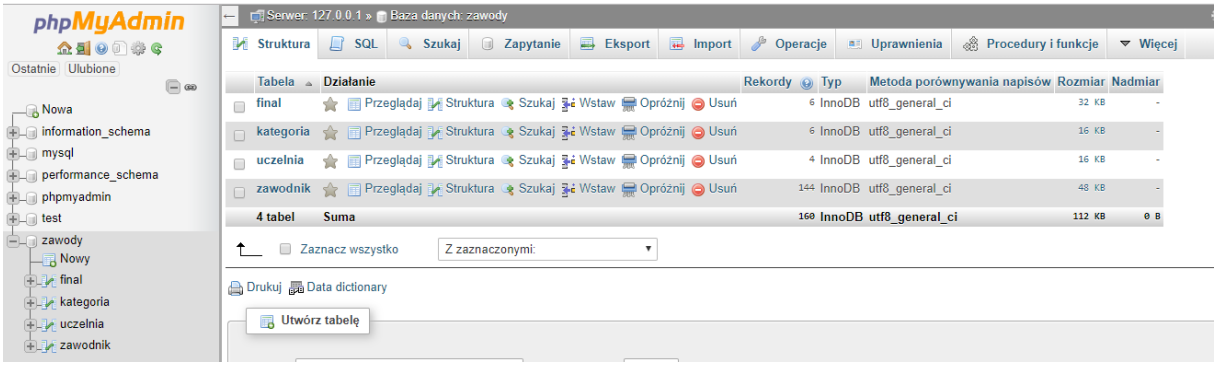
Finał – w finale zawodnicy mają do pokonania N dróg finałowych (możemy przyjąć że N = 4). W finale oprócz topów i bonusów wykonanych dróg zliczamy także próby – w ilu podejściach wykonano daną drogę (przykładowo Krzysiek osiągnął bonus pierwszej finałowej drogi po 4 nieudanych próbach oraz top tej samej drogi po 6 nieudanych próbach, co jest lepszym wynikiem od Kuby, który wykonał tę samą drogę w 4 próbach do bonusu i w 8 próbach do topu). Jeśli zawodnicy na podium mają sytuację remisową – rozstrzyga liczba punktów osiągniętych w eliminacjach.

2. Przebieg mini-projektu.

Zaprojektowanie diagramu ERD w środowisku MySQL Workbench.



Połączenie z localhostem i implementacja pustych tabel z relacjami (na podstawie zaprojektowanego diagramu ERD).



Pobranie danych o zawodnikach z pliku CSV (załączonego z treścią zadania) oraz wypełnienie tabel w środowisku Python.

```
import MySQLdb
import random
import csv

db = MySQLdb.connect(host='localhost', user='root', passwd='', db='zawody')
myCursor = db.cursor()
```

```
filename = 'poprawione_dane.csv'
with open(filename) as csvfile:
    readCSV = csv.reader(csvfile, delimiter=',')
    imiona=[]
    nazwiska=[]
    kluby=[]

    for row in readCSV:
        nazwiska.append(row[0])
        imiona.append(row[1])
        kluby.append(row[2])
```

Zaprojektowanie systemu pozwalającego na wyłonienie zwycięzcy w kategoriach: JUNIORKI, JUNIORZY, MASTERKI, MASTERSI.

```
# -*- coding: utf-8 -*-
import MySQLdb
import random

db = MySQLdb.connect(host='localhost', user='root', passwd='', db='zawody')
myCursor = db.cursor()


myCursor.execute("""select idZawodnik, Imie, Nazwisko, TOP, BONUS
            from kategoria inner join zawodnik
            on kategoria.idKategoria = zawodnik.Kategoria_idKategoria
            where Nazwa = 'Juniorki'""")

result = myCursor.fetchall()
juniorki=[]
for i in range(0,len(result)):
    juniorki.append(result[i])

#sortowanie po losowo przydzielonych punktach:
juniorki.sort(key=lambda tup: tup[3]+(tup[4]*0.01))
juniorki.reverse()
```

Wygenerowanie pliku pdf za pomocą wbudowanej biblioteki Python'a „reportlab.pdfgen".

```
# -*- coding: utf-8 -*-
from mini_projekt_reszta import juniorki
from reportlab.pdfbase.ttfonts import *
from reportlab.pdfgen import canvas
from reportlab.pdfbase import pdfmetrics



pdfmetrics.registerFont(TTFont('Hebrew','Arial.ttf'))
c=canvas.Canvas('juniorki.pdf')
c.drawString(50, 800, 'KATEGORIA: JUNIORKI')
z=1
k=780
for i in range(0,len(juniorki)):
    c.setFont('Hebrew',14)
    c.drawString(50, k, str(z))
    c.drawString(80, k, (juniorki[i][1]+' '+juniorki[i][2]).decode('utf-
8'))
    c.drawString(250, k, ('TOPÓW: ' + str(juniorki[i][3])).decode('utf-8'))
```

```
    c.drawString(350, k, ('BONUSÓW: ' + str(juniorki[i][4])).decode('utf-
8'))
    k = k - 15
    z = z + 1
c.save()
```

Końcowy raport w kategorii „JUNIORKI”.

KATEGORIA: JUNIORKI

1	Bianka Barabaś	TOPÓW: 16	BONUSÓW: 18
2	Kasia Karkola	TOPÓW: 10	BONUSÓW: 10
3	Bianka Janecka	TOPÓW: 8	BONUSÓW: 8
4	Dominika Herman	TOPÓW: 6	BONUSÓW: 7
5	Iga Kubicz	TOPÓW: 5	BONUSÓW: 5
6	Julia Połeć	TOPÓW: 1	BONUSÓW: 2

Zaprojektowanie systemu pozwalającego na wyłonienie zwycięzcy w kategoriach: KOBIETY, MĘŻCZYŹNI. Tutaj dzielimy zawody na eliminacje i finały, a wyłonienie finalistów odbywa się w ten sam sposób co sortowanie po punktach w czterech pozostałych kategoriach. Planowana liczba osób, która przejdzie do finału to 6, chyba, że osoba, która jest na 7 miejscu (wzwyż) ma tyle samo punktów co osoba na 6 miejscu - wtedy również ta osoba przechodzi do finału.

Eliminacje w kategorii „KOBIETY” i „MĘŻCZYŹNI”.

```
# -*- coding: utf-8 -*-
import MySQLdb
from reportlab.pdfbase.ttfonts import *
from reportlab.pdfgen import canvas
from reportlab.pdfbase import pdfmetrics

db = MySQLdb.connect(host='localhost', user='root', passwd='', db='zawody')
myCursor = db.cursor()


x=raw_input('podaj kategorie: ')
myCursor.execute("""select idZawodnik, Imie, Nazwisko, TOP, BONUS
            from kategoria inner join zawodnik
            on kategoria.idKategoria = zawodnik.Kategoria_idKategoria
            where Nazwa = %s""", (x))

result = myCursor.fetchall()
kategoria=[]
for i in range(0,len(result)):
    kategoria.append(result[i])

kategoria.sort(key=lambda tup: tup[3]+(tup[4]*0.01))
kategoria.reverse()


pdfmetrics.registerFont(TTFont('Hebrew','Arial.ttf'))
c=canvas.Canvas(str(x)+'-eliminacje.pdf')
c.drawString(50, 800, 'KATEGORIA: '+str(x))
```

```
z=1
k=780
for i in range(0,len(kategoria)):
    c.setFont('Hebrew',14)
    c.drawString(50, k, str(z))
    c.drawString(80, k, (kategoria[i][1]+' '+kategoria[i][2]).decode('utf-
8'))
    c.drawString(250, k, ('TOPÓW: ' + str(kategoria[i][3])).decode('utf-
8'))
    c.drawString(350, k, ('BONUSÓW: ' + str(kategoria[i][4])).decode('utf-
8'))
    k = k - 15
    z = z + 1
c.save()
```

Końcowy raport z **eliminacji** w kategorii „KOBIETY”.

KATEGORIA: kobiety

1	Dorota Halusiak	TOPÓW: 18	BONUSÓW: 21
2	Justyna Miklas	TOPÓW: 17	BONUSÓW: 20
3	Justyna Szczygieł	TOPÓW: 17	BONUSÓW: 20
4	Magdalena Piosek	TOPÓW: 17	BONUSÓW: 18
5	EWA GRZESZCZAK	TOPÓW: 16	BONUSÓW: 18
6	Gosia Bobnis	TOPÓW: 16	BONUSÓW: 16
7	Marysia Łaszkiewicz	TOPÓW: 14	BONUSÓW: 15
8	Majka Miezianko	TOPÓW: 12	BONUSÓW: 13
9	Zuzanna Banaś	TOPÓW: 11	BONUSÓW: 14
10	Katarzyna Mirosław	TOPÓW: 11	BONUSÓW: 14
11	Olga Medvedieva	TOPÓW: 11	BONUSÓW: 11
12	Anna Przeliorz-Pyszczek	TOPÓW: 9	BONUSÓW: 12
13	Ania Sroka	TOPÓW: 9	BONUSÓW: 10
14	Katarzyna Mikuś	TOPÓW: 8	BONUSÓW: 10
15	Justyna Baranowska	TOPÓW: 8	BONUSÓW: 8
16	Ania Grabińska	TOPÓW: 6	BONUSÓW: 7
17	Agnieszka Szczepaniec	TOPÓW: 5	BONUSÓW: 7
18	Izabela Juszczak	TOPÓW: 4	BONUSÓW: 4
19	Małgorzata Smełka	TOPÓW: 2	BONUSÓW: 3
20	Natalia Sidorowicz	TOPÓW: 2	BONUSÓW: 2
21	Zolak Czemarmazowicz	TOPÓW: 0	BONUSÓW: 3
22	Marianna Piwońska	TOPÓW: 0	BONUSÓW: 2
23	Marta Kostrzewska	TOPÓW: 0	BONUSÓW: 0
24	Elena □Yarova	TOPÓW: 0	BONUSÓW: 0

Opracowanie algorytmu pozwalającego na znalezienie ile osób przejdzie do finału.

```
k=6
for i in range(6,len(kategoria)):

    if (kategoria[5][3] + (kategoria[5][4] * 0.001)) == (kategoria[k][3] +
(kategoria[k][4] * 0.001)):
        k = k + 1
```

Po wyłonieniu finalistów, wykonujemy ponowną symulację, tym razem na nowych, bardziej rozbudowanych warunkach (warunki zostały opisane w treści zadania). Oprócz topów i bonusów pojawiają się jeszcze próby i drogi oznaczone przez zmienną „N”.

Finał w kategorii „KOBIETY” i „MĘŻCZYŹNI”.

```
# -*- coding: utf-8 -*-
import MySQLdb
from mini_projekt_elim import final
import random

db = MySQLdb.connect(host='localhost', user='root', passwd='', db='zawody')
myCursor = db.cursor()

myCursor.execute("""DELETE FROM final""")


#N liczba dróg
N=8
for i in range(0,len(final)):
    top=random.randint(0,N)
    bonus=top+random.randint(0,2)
    proba_top=top+random.randint(0,2)
    proba_bonus=bonus+random.randint(0,4)
    myCursor.execute("""INSERT INTO
final(idFinal,final_top,final_bonus,proba_top,proba_bonus,Zawodnik_idZawodn
ik)
    VALUES
(%s,%s,%s,%s,%s,%s);""",(i+1,top,bonus,proba_top,proba_bonus,final[i][0]))


myCursor.execute("""select Imie, Nazwisko, final_top, final_bonus,
proba_top, proba_bonus, Zawodnik_idZawodnik
            from zawodnik inner join final
            on zawodnik.idZawodnik = final.Zawodnik_idZawodnik""")

result = myCursor.fetchall()
final=[]
for i in range(0,len(result)):
    final.append(result[i])


final.sort(key=lambda tup: (tup[2]+tup[3]*0.1)-((tup[4]*0.01)+tup[5]*0.01))
final.reverse()

for osoba in final:
    print osoba

myCursor.execute("""DELETE FROM final""")
for i in range(len(final)):
    myCursor.execute("""INSERT INTO
final(idFinal,final_top,final_bonus,proba_top,proba_bonus,Zawodnik_idZawodn
ik)
    VALUES
(%s,%s,%s,%s,%s,%s);""",(i+1,final[i][2],final[i][3],final[i][4],final[i][5
],final[i][6]))

db.commit()
db.close()
```

Opracowanie algorytmu umożliwiającego poprawne dobieranie wyrazów do liczby prób jaka została przypisana do danego zawodnika, np.:

„TOPÓW: 0 W 0 PRÓBACH, BONUSÓW: 0 W 1 PRÓBIE

```
# -*- coding: utf-8 -*-
import MySQLdb
from mini projekt elim import x
from reportlab.pdfbase.ttfonts import *
from reportlab.pdfgen import canvas
from reportlab.pdfbase import pdfmetrics

db = MySQLdb.connect(host='localhost', user='root', passwd='', db='zawody')
myCursor = db.cursor()
pdfmetrics.registerFont(TTFont('Hebrew','Arial.ttf'))
c=canvas.Canvas(str(x)+'-final.pdf')


myCursor.execute("""SELECT imie, nazwisko, final_top, proba_top, final_bonus, proba_bonus
FROM final INNER JOIN zawodnik
ON final.zawodnik_idzawodnik=zawodnik.idzawodnik""")
result = myCursor.fetchall()


z=0
final=[]
for i in result:
    final.append(i)
k=780
szerokosc_t=220
szerokosc_b=380
for i in range(0,len(final)):
    c.setFont('Hebrew', 12)
    if final[i][3]==1:
        print i+1,final[i][0],final[i][1]
        print 'topów:',final[i][2],'w',final[i][3],'próbie'
        c.drawString(50, k, str(i + 1))
        c.drawString(80, k, (final[i][0] + ' ' + final[i][1]).decode('utf-8'))
        c.drawString(szerokosc_t, k, ('TOPÓW: ' + str(final[i][2]) + ' w ' + str(final[i][3])
+ ' PRÓBIE').decode('utf-8'))

        if final[i][5]==1:
            print 'bonusów:',final[i][4],'w',final[i][5],'próbie'
            c.drawString(szerokosc_b, k, ('BONUSÓW: ' + str(final[i][4]) + ' w ' +
str(final[i][5]) + ' PRÓBIE').decode('utf-8'))
        else:
            print 'bonusów:',final[i][4],'w',final[i][5],'próbach'
            c.drawString(szerokosc_b, k, ('BONUSÓW: ' + str(final[i][4]) + ' w ' +
str(final[i][5]) + ' PRÓBACH').decode('utf-8'))
    elif final[i][5]==1:
        print i+1,final[i][0],final[i][1]
        print 'topów:',final[i][2],'w',final[i][3],'próbach'
        print 'bonusów:',final[i][4],'w',final[i][5],'probie'
        c.drawString(50, k, str(i + 1))
        c.drawString(80, k, (final[i][0] + ' ' + final[i][1]).decode('utf-8'))
        c.drawString(szerokosc_t, k, ('TOPÓW: ' + str(final[i][2]) + ' w ' + str(final[i][3])
+ ' PRÓBACH').decode('utf-8'))
        c.drawString(szerokosc_b, k, ('BONUSÓW: ' + str(final[i][4]) + ' w ' +
str(final[i][5]) + ' PRÓBIE').decode('utf-8'))
    else:
        print i+1,final[i][0],final[i][1]
        print 'topów:',final[i][2],'w',final[i][3],'próbach'
        print 'bonusów:',final[i][4],'w',final[i][5],'próbach'
        c.drawString(50, k, str(i + 1))
        c.drawString(80, k, (final[i][0] + ' ' + final[i][1]).decode('utf-8'))
        c.drawString(szerokosc_t, k, ('TOPÓW: ' + str(final[i][2]) + ' w ' + str(final[i][3])
+ ' PRÓBACH').decode('utf-8'))
        c.drawString(szerokosc_b, k, ('BONUSÓW: ' + str(final[i][4]) + ' w ' +
str(final[i][5]) + ' PRÓBACH').decode('utf-8'))
    k-=15

c.save()
db.commit()
db.close()
```

Końcowy raport z **finału** w kategorii „KOBIETY”.

1	Justyna Szczygieł	TOPÓW: 4 w 6 PRÓBACH	BONUSÓW: 6 w 10 PRÓBACH
2	Justyna Miklas	TOPÓW: 2 w 4 PRÓBACH	BONUSÓW: 4 w 5 PRÓBACH
3	Dorota Halusiak	TOPÓW: 2 w 3 PRÓBACH	BONUSÓW: 3 w 5 PRÓBACH
4	Magdalena Piosek	TOPÓW: 1 w 3 PRÓBACH	BONUSÓW: 2 w 2 PRÓBACH
5	EWA GRZESZCZAK	TOPÓW: 1 w 2 PRÓBACH	BONUSÓW: 1 w 2 PRÓBACH
6	Gosia Bobnis	TOPÓW: 0 w 1 PRÓBIE	BONUSÓW: 0 w 1 PRÓBIE

DODATKOWO:

Opracowanie interface’u pozwalającego na ręczne wprowadzenie danych o zawodnikach. Poszczególne funkcje zostały opisane bezpośrednio w kodzie.

```
# -*- coding: utf-8 -*-
from PyQt5.QtWidgets import QApplication, QWidget
from PyQt5.QtGui import QIcon
from PyQt5.QtWidgets import QLabel, QGridLayout
from PyQt5.QtWidgets import QLineEdit, QPushButton, QHBoxLayout
from PyQt5.QtWidgets import QMessageBox
from PyQt5.QtCore import Qt
import sys
import xlwt
i=0
lista=[]

class Zawody(QWidget):

    def __init__(self, parent=None):
        super().__init__(parent)

        self.interfejs()

    def interfejs(self):

        # etykiety
        etykieta1 = QLabel("Id:", self)
        etykieta2 = QLabel("Imie:", self)
        etykieta3 = QLabel("Nazwisko:", self)
        etykieta4 = QLabel("Top: ",self)
        etykieta5 = QLabel("Bonus: ",self)
        etykieta6 = QLabel("Klub zawodnika",self)
        # przypisanie widgetów do układu tabelarycznego
        ukladT = QGridLayout()
        ukladT.addWidget(etykieta1, 0, 0)
        ukladT.addWidget(etykieta2, 0, 1)
        ukladT.addWidget(etykieta3, 0, 2)
        ukladT.addWidget(etykieta4,0,3)
        ukladT.addWidget(etykieta5,0,4)
        ukladT.addWidget(etykieta6,0,5)

        # 1-liniowe pola edycyjne
        self.idEdt = QLineEdit()
        self.imieEdt = QLineEdit()
        self.nazwiskoEdt =QLineEdit()
        self.topEdt = QLineEdit()
```

```
        self.bonusEdt = QLineEdit()
        self.klubzawEdt = QLineEdit()

        ukladT.addWidget(self.idEdt, 1, 0)
        ukladT.addWidget(self.imieEdt, 1, 1)
        ukladT.addWidget(self.nazwiskoEdt, 1, 2)
        ukladT.addWidget(self.topEdt,1,3)
        ukladT.addWidget(self.bonusEdt,1,4)
        ukladT.addWidget(self.klubzawEdt,1,5)
        # przyciski
        zapiszBtn = QPushButton("&Zapisz zawodnika", self)
        koniecBtn = QPushButton("&Koniec", self)
        koniecBtn.resize(koniecBtn.sizeHint())

        ukladH = QHBoxLayout()
        ukladH.addWidget(zapiszBtn)


        ukladT.addLayout(ukladH, 2, 0, 1, 3)
        ukladT.addWidget(koniecBtn, 3, 0, 1, 3)

        # przypisanie utworzonego układu do okna
        self.setLayout(ukladT)
        koniecBtn.clicked.connect(self.koniec)
        zapiszBtn.clicked.connect(self.dzialanie)



        self.idEdt.setFocus()
        self.setGeometry(20, 20, 400, 100)
        self.setWindowTitle("Zawody")
        self.show()

    def koniec(self):
        self.close()

    def closeEvent(self, event):

        odp = QMessageBox.question(
            self, 'Komunikat',
            "Czy na pewno koniec?",
            QMessageBox.Yes | QMessageBox.No, QMessageBox.No)

        if odp == QMessageBox.Yes:
            event.accept()
        else:
            event.ignore()

    def keyPressEvent(self, e):
        if e.key() == Qt.Key_Escape:
            self.close()

    def dzialanie(self):
        global i,lista
        nadawca = self.sender()

        try:
            id = float(self.idEdt.text())
            imie = (self.imieEdt.text())
            nazwisko =(self.nazwiskoEdt.text())
            top =float(self.topEdt.text())
```

```
            bonus = float(self.bonusEdt.text())
            klubzaw = (self.klubzawEdt.text())
            if nadawca.text() == "&Zapisz zawodnika" :
                book = xlwt.Workbook(encoding="utf-8")
                sheet1 = book.add_sheet("Sheet 1")
                sheet1.write(i, 0, id)  # id
                sheet1.write(i, 1, imie)  # imie
                sheet1.write(i,2,nazwisko)
                sheet1.write(i,3,top)
                sheet1.write(i,4,bonus)
                sheet1.write(i,5,klubzaw)
                i=i+1
                lista.append(id)


        except ValueError:
            QMessageBox.warning(self, "Błąd", "Błędne dane",
QMessageBox.Ok)
        book.save("trial.xls")
if __name__ == '__main__':

    app = QApplication(sys.argv)
    okno = Zawody()
    sys.exit(app.exec_())
```

Interface – finalna postać.