МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ БЕЛАРУСЬ
БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет информационных технологий и робототехники (ФИТР)
Кафедра программного обеспечения информационных систем и технологий

КУРСОВОЙ ПРОЕКТ

По дисциплине: «**Языки программирования**»

на тему: «**Калькулятор массы тела**»

Исполнитель: студент группы 10701323
Шаплавский Никита Сергеевич

Преподаватель: доц. Сидорик Валерий Владимирович

Минск 2024

Шаплавский Никита Сергеевич, гр.10701323

Оглавление

Шаплавский Никита Сергеевич, гр.10701323

ВВЕДЕНИЕ

В современном информационном мире компьютерные программы играют важную роль в различных областях, включая анализ и моделирование различных систем. Одной из таких систем является анализ индекса массы тела (BMI), который находит применение в медицине, фитнесе и других областях, связанных со здоровьем. Определение и анализ BMI являются важными для оценки состояния здоровья и определения ключевых параметров тела человека.

Данный курсовой проект направлен на разработку программы на языке программирования Python с использованием библиотеки Tkinter для анализа и вычисления параметров BMI. Цель проекта заключается в создании удобного и функционального инструмента, который позволит пользователям проводить вычисления и оценивать характеристики BMI.

Программа будет охватывать широкий спектр функций, включая ввод параметров тела (рост, вес, возраст, пол), вычисление основных характеристик (таких как BMI, базальный метаболизм, идеальный вес), а также визуализацию результатов анализа. Использование библиотеки Tkinter позволит создать интуитивно понятный и удобный графический интерфейс для взаимодействия с программой.

Данный проект предполагает использование принципов программирования и анализа параметров тела для создания полезного инструмента, который будет не только обладать функциональностью для вычисления параметров BMI, но также обеспечит пользователей инструментами для лучшего понимания и визуализации основных аспектов состояния здоровья и физической формы.

Шаплавский Никита Сергеевич, гр.10701323

Постановка и описание задачи

Текст задания: Создать программу для определения оптимальной массы тела человека, занимающегося физическими упражнениями (с использованием формулы Креффа).

Цель программы: разработать приложение на языке программирования Python, которая позволяет пользователям просто и быстро вычислять параметры своего организма, а также предоставление удобного интерфейса для выполнения вычислений

Детализация задачи:

1. **Выбор языка интерфейса**:
   -Пользователи должны иметь возможность выбора языка, для того, чтобы программа не имела ограниченной локализации.

2. **Проверка вводимых значений**:
   -Обеспечение проверки корректности и правильности ввода данных перед расчетом параметров человека.

3. **Сохранение значений**:
   -Приложение должно сохранять параметры которые пользовотель в вводит, а так же получает в ходе расчета

4. **Пояснение к результатам**
   -Приложение должно комментировать телосложение человека

5. **Запись в файл и чтение из файла**:
   -Запись сохранённых значений и результатов.
   -Чтение значений и результатов из файла и вывод их.

6. **Графическое представление**:
   -Предоставление удобного и интуитивно понятного интерфейса

Исходные данные:
-Операционная система MsWindows
-Среда разработки PyCharm

Инструменты разработки:
-Язык программирования Python
-Библиотеки Python
-Формула Креффа

Шаплавский Никита Сергеевич, гр.10701323

Выходные данные:

-Идеальная масса для человека

-Индекса массы тела

-Индекса базального обмена веществ

Специальные требования:

-Отсутствуют.

Шаплавский Никита Сергеевич, гр.10701323

3 Обзор предметной области

Изучение предметной области для данной задачи начнётся с изучения информации о телосложении человека и его взаимосвязях. Это включает в себя понимание того, что такое индекс массы тела (BMI), базальный метаболизм (BMR) и идеальный вес.

Индекс массы тела (BMI) - это показатель, который использует вес и рост человека для оценки, находится ли его вес в пределах здорового диапазона. Он рассчитывается путём деления веса человека в килограммах на квадрат его роста в метрах. BMI широко используется как инструмент для скрининга, чтобы классифицировать людей в различные весовые категории, такие как недостаточный вес, нормальный вес, избыточный вес и ожирение.

Базальный метаболизм (BMR) - это количество калорий, необходимое для поддержания жизнедеятельности организма в состоянии покоя. BMR зависит от нескольких факторов, включая возраст, пол, вес и рост. Он представляет собой минимальное количество энергии, необходимое для поддержания жизненно важных функций организма, таких как дыхание, кровообращение и производство клеток.

Идеальный вес - это концепция, которая относится к оптимальному весу для человека, основанному на его росте, возрасте и поле. Он часто используется как цель для людей, стремящихся достичь здорового веса. Идеальный вес можно рассчитать с использованием различных формул, которые учитывают разные факторы для предоставления целевого диапазона веса, считающегося здоровым для человека.

Понимание этих концепций является ключевым для разработки программы, которая сможет точно анализировать и рассчитывать эти параметры, предоставляя пользователям ценные сведения о их здоровье и физической форме.

Шаплавский Никита Сергеевич, гр.10701323

4 Построение математической модели

1. Индекс массы тела (BMI)

Параметры для расчёта: вес и рост

Словесное описание: вес поделить на рост в квадрате, предварительно переведя рост в метры

Формула: $\frac{\text{вес}}{\text{рост}^2}$

2. Базальный метаболизм (BMR)

Для нахождения этого параметра использовалась формула из электронного источника calorizator - https://calorizator.ru/article/body/bmr-calculation

Дата доступа:02.12.2024

Параметры для расчёта: вес, рост, возраст, пол

Словесное описание: к параметру A прибавить произведение параметра B с весом, произведение C с ростом и отнять произведение D с возрастом

Значение параметров для мужского пола:

A=88.7 ; B=13.4; C=4.8; D=5.7;

Значение параметров для женского пола пола:

A=447.6 ; B=9.2; C=3.1; D=4.3;

Формула: $\text{A} + (\text{B} * \text{Вес}) + (\text{C} * \text{Рост}) - (D * \text{Возраст})$

Шаплавский Никита Сергеевич, гр.10701323

3. Идеальная масса тела (Формула Креффа)

Для нахождения этого параметра использовалась формула из электронного источника “Твое питание” - https://www.yournutrition.ru/weight/formula-kreffa/

Дата доступа:02.12.2024

Параметры для расчёта: рост, возраст, пол

Словесное описание: от роста отнять 100, затем умножить на 0.9 и прибавить возраст поделенный на 10 и умноженный на параметр А

Значение параметров для мужского пола:

А=1

Значение параметров для женского пола пола:

А=0.9

Формула: $(\text{Рост} - 100) * 0.9 + \left(\frac{\text{возраст}}{10}\right) * \text{А}$

Шаплавский Никита Сергеевич, гр.10701323

Диаграммы

Унифицированный язык моделирования (UML) представляет собой инструмент для определения, визуализации, разработки и документирования программных и не программных систем. Этот язык объединяет различные инженерные методы, успешно применяемые для моделирования разнообразных систем. Предварительное планирование и моделирование играют значительную роль в упрощении процесса разработки программного обеспечения для решения сложных задач. Кроме того, изменения в UML-диаграммах классов проще вносить, чем в исходный код.

UML-диаграмма классов используется для отображения структуры системы, показывая классы, их атрибуты и методы, а также взаимосвязи между классами. Это помогает моделировать объектно-ориентированный дизайн, иллюстрируя структуру и организацию классов в программе.

UML-диаграмма развёртывания описывает физическую структуру системы, включая компоненты, узлы (например, серверы) и связи между ними. Этот вид диаграммы позволяет визуализировать размещение компонентов и их взаимодействие в различных физических средах, таких как серверы и клиентские устройства.

UML-диаграмма активностей используется для моделирования последовательности действий или процессов в системе, отображая поток управления и взаимодействие между элементами. Это помогает визуализировать шаги выполнения процессов или сценариев в системе, включая ветвления, циклы и условия.

Эти диаграммы являются важными инструментами для анализа, проектирования и документирования систем, предоставляя наглядное представление различных аспектов программных проектов. Они способствуют лучшему пониманию структуры и функциональности системы, а также облегчают коммуникацию между разработчиками и заинтересованными сторонами. Следовательно, одним из этапов к реализации курсового проекта было создание диаграмм UML (рис. 5.1).

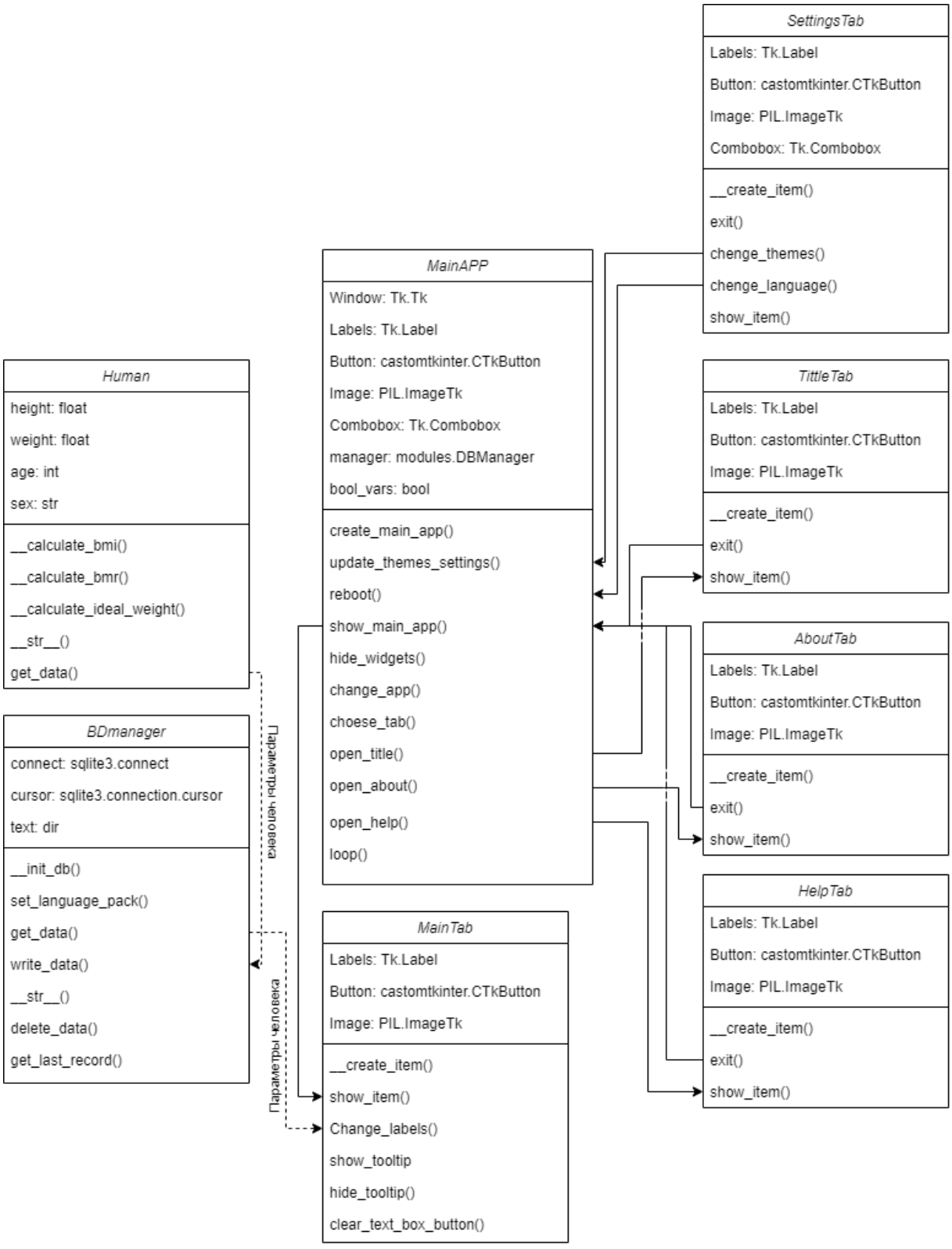

Рисунок 5.1 – UML диаграмма классов

Шаплавский Никита Сергеевич, гр.10701323

6 Алгоритм работы программы

Диаграмма блок-схемы - это графическое отображение структуры программы и последовательности ее выполнения. Этот инструмент визуализации позволяет наглядно представить логику алгоритма, выделяя ключевые этапы обработки данных и взаимодействия между различными компонентами системы. На рисунке 6.1 изображена блок-схема программы, посвященной "Калькулятор массы тела".

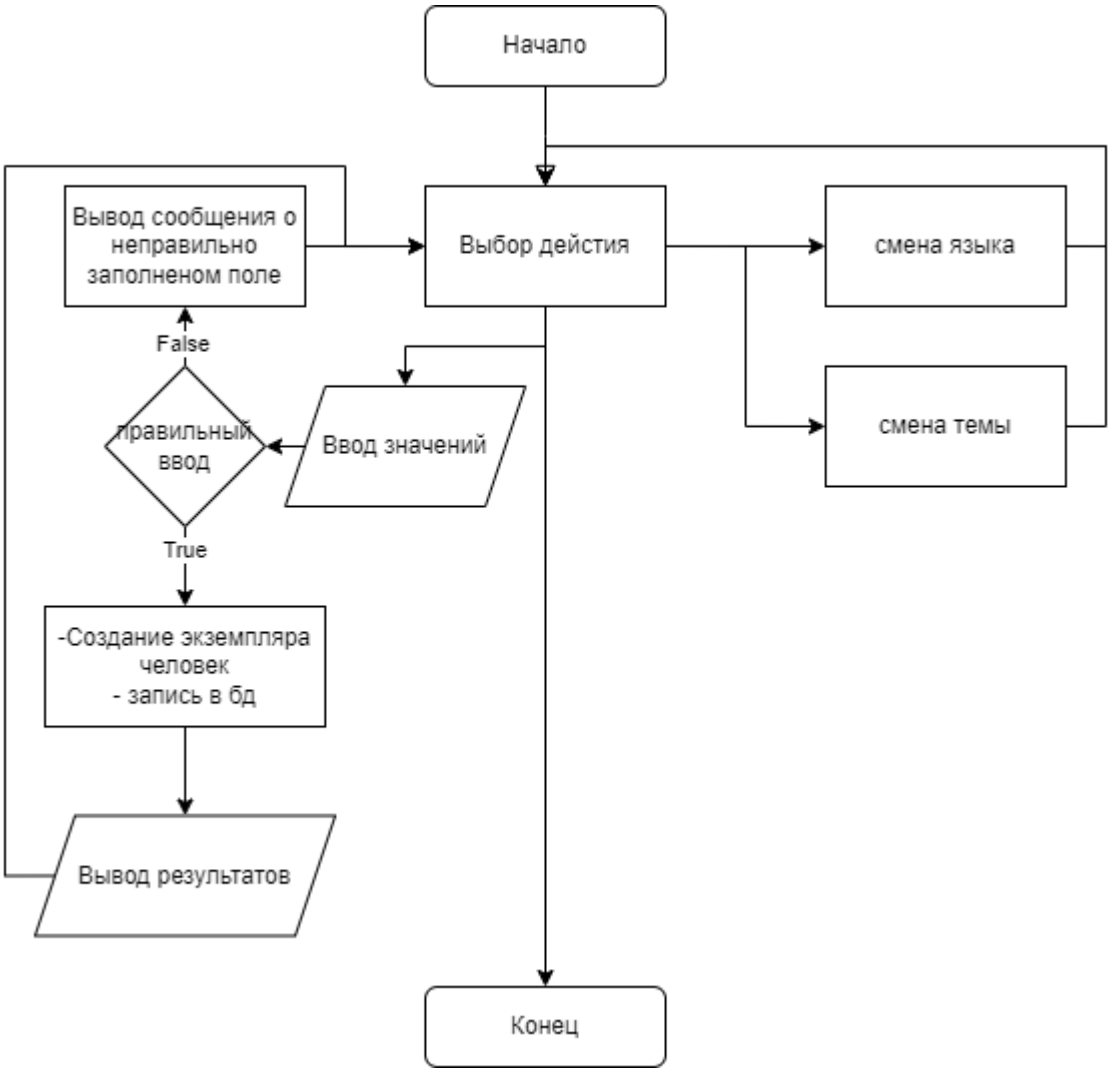

Рис. 6.1 Блок-схема работы программы

Проект представлен блок-схемой или, как ее по другому называют, бизнес-логикой переключения окон проекта, состоящая из 6 окон взаимосвязанных между собой: TittleTab (стартовое окно приложения), MainTab (основное окно для работы с приложением и логикой), HelpTab (о программе), AboutTab (об авторе), SettingsTab (окно с настройками), (рис. 6.2):

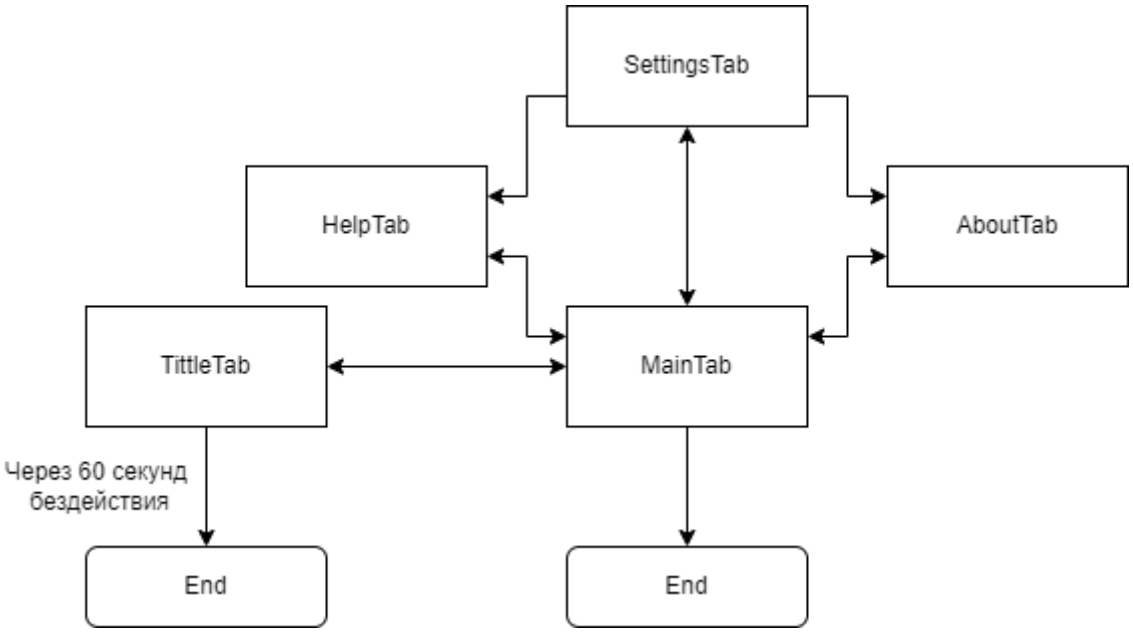

Рис. 6.2. Бизнес-логика переключения графических окон

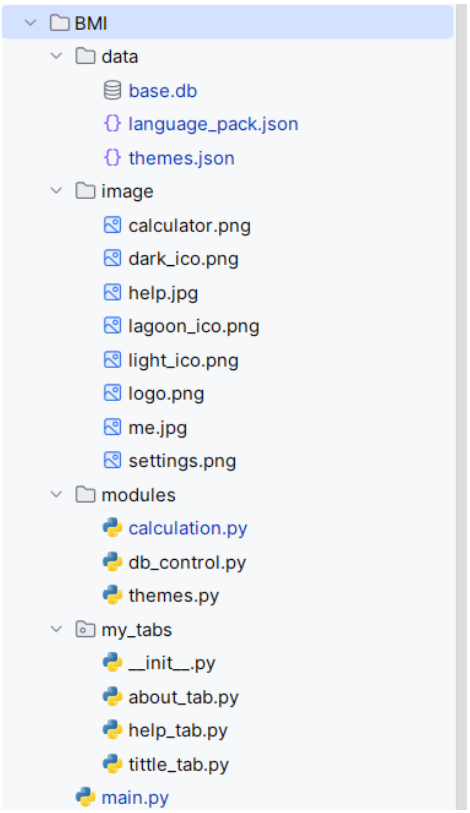

Рис. 6.3. Структура представления материала в электронном виде

Шаплавский Никита Сергеевич, гр.10701323

7 Описание работы графических окон

a. Стартовое окно TittleTab

Окно TittleTab является стартовым, содержит элементы управления для перехода в главное окно и выхода из программы. Оно отображается, пока пользователь не нажмет кнопку “Далее” в течении 60 секунд, иначе оно автоматически закроется. Пользователь может выйти из приложения при помози кнопки “Выход”. Стартовое окно представлено на рисунке 7.1.

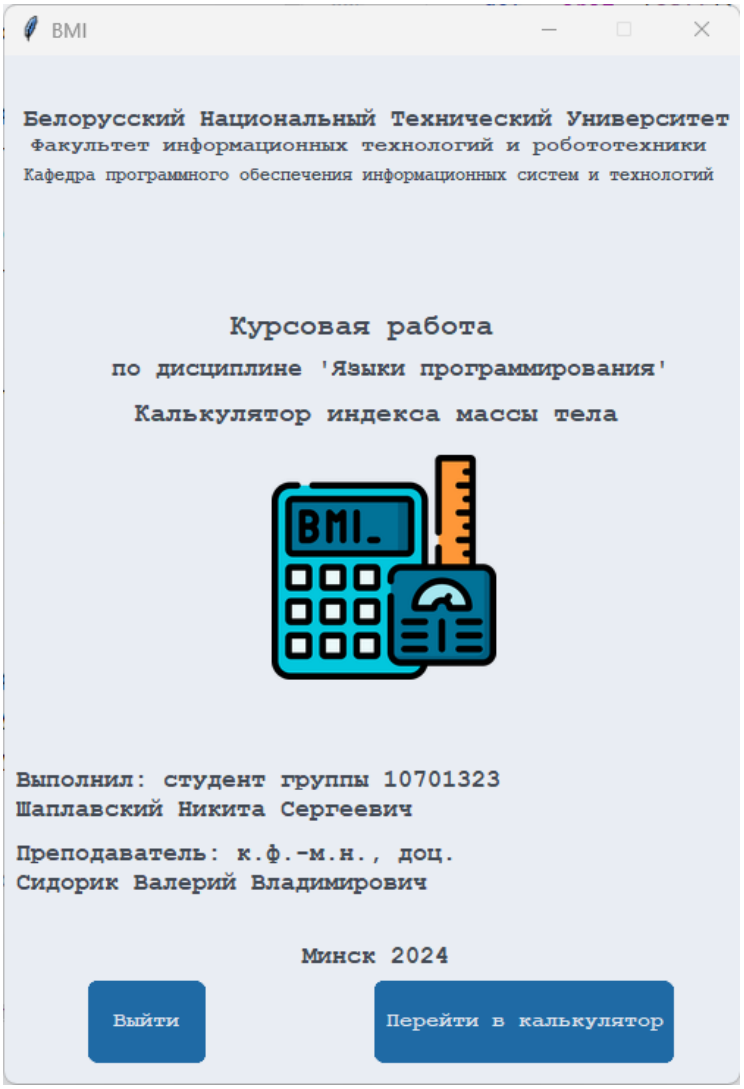

Рисунок 7.1 – Окно TittleTab

Шаплавский Никита Сергеевич, гр.10701323

b. Главное окно MainTab

Главное окно (см. рисунок 7.2.1) предоставляет возможность вводить параметры, рассчитывать BMI, BMR и идеальную массу тела, просматривать историю

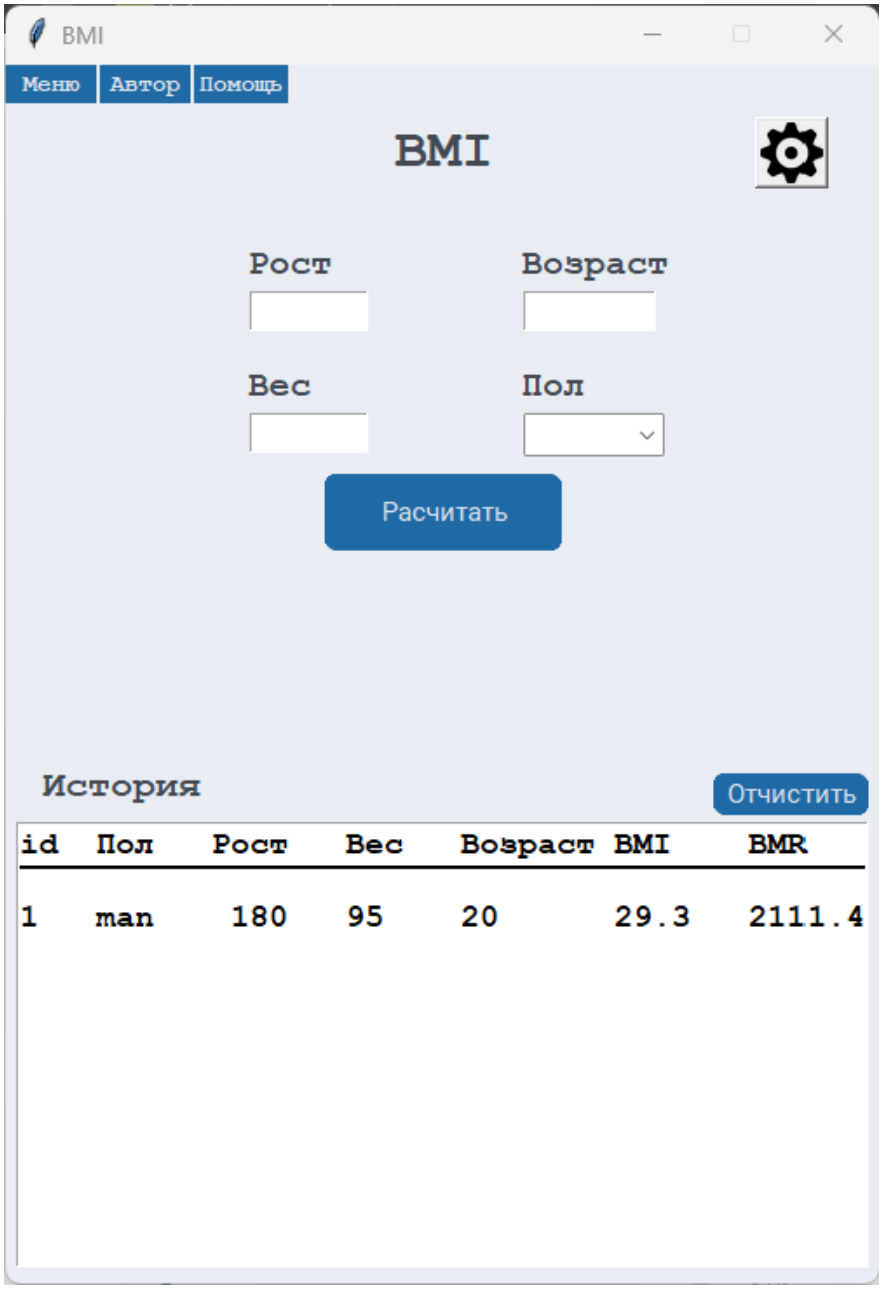

Рис 7.2.1 – Начальная форма MainWindow

После ввода параметров пользователем, и нажатия кнопки “Расчитать” в окне появятся следующие значение (см. рисунок 7.2.2):

1. Индекс массы тела BMI.
2. Индекс базального метаболизма(обмена веществ)BMR.
3. Индеальная масса.

4. Внесение этого расчета в БД и вывод его в окно “История”
5. Кнопка “Отчистить”

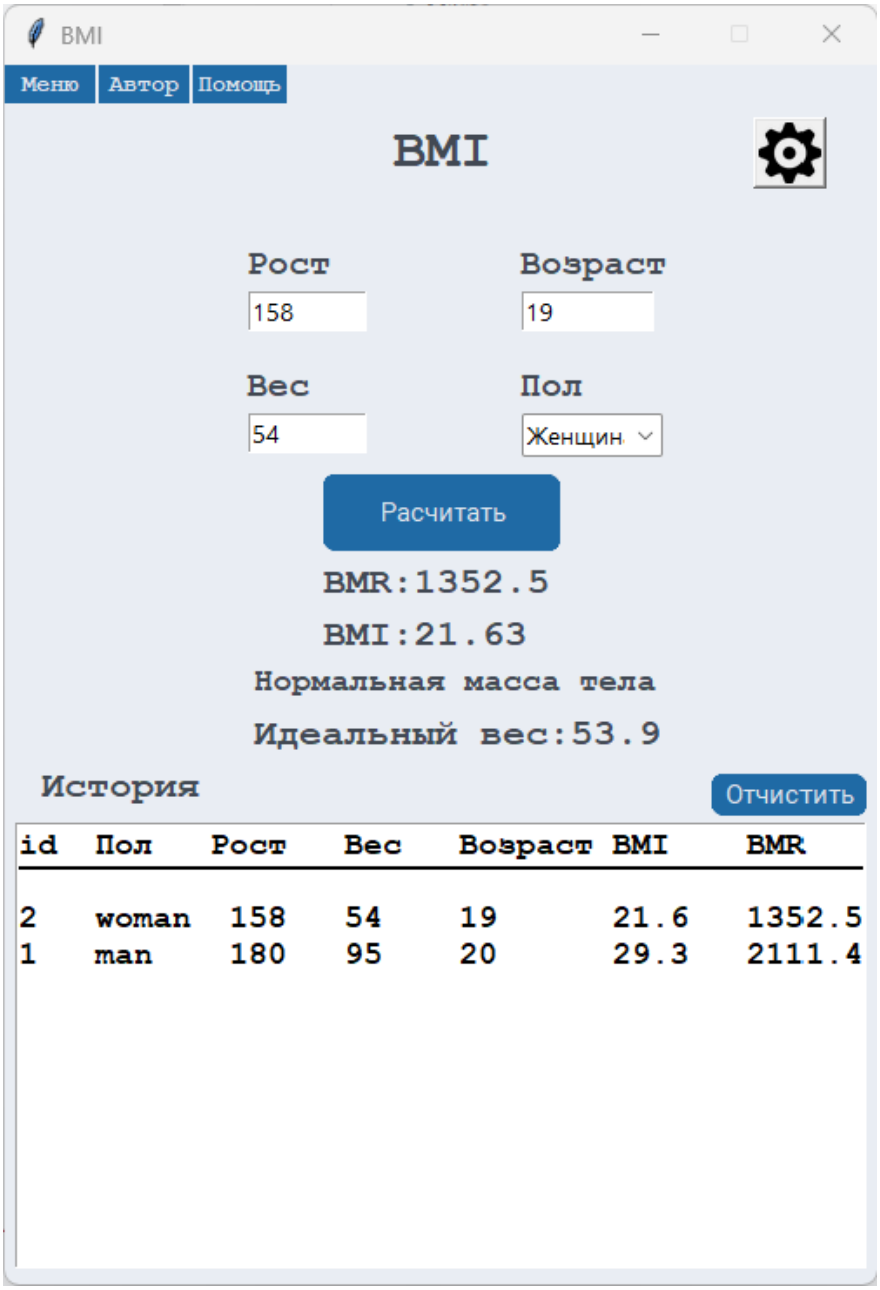

Рисунок 7.2.2 – Главное окно после ввода параметров

c. Окно настроек SettingsTab

После нажатия на кнопку “ ” появится окно (см. рис.7.3 с возможностями:

1. Изменение темы
2. Изменение языка

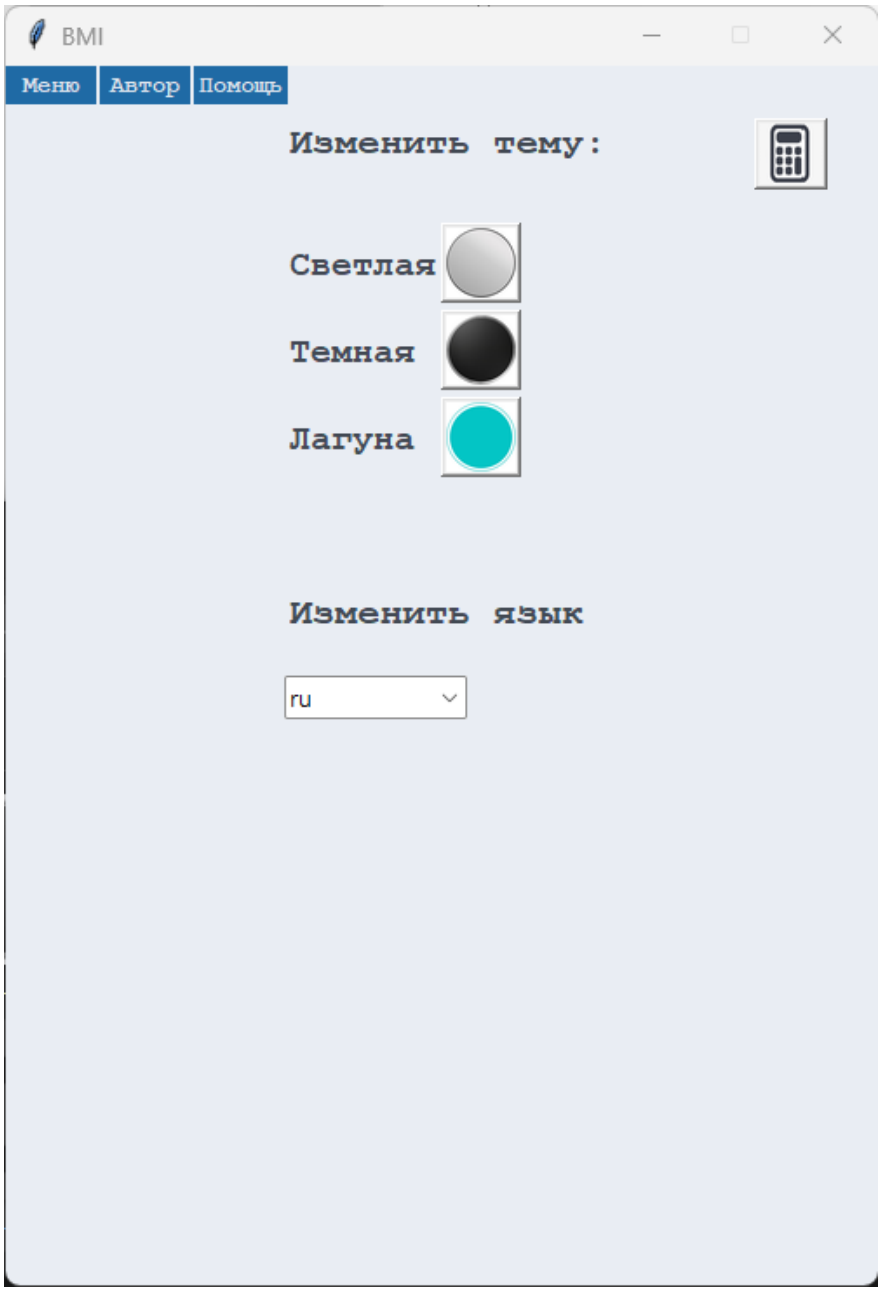

Рисунок 7.3 – Окно настроек SettingsTab

Шаплавский Никита Сергеевич, гр.10701323

d. Об авторе

Окно “Об Авторе” (см. рисунок 7.4) содержит основную информацию об авторе проекта: номер группы в университете, фамилию, имя, отчество, адрес электронной почты, фотографию.

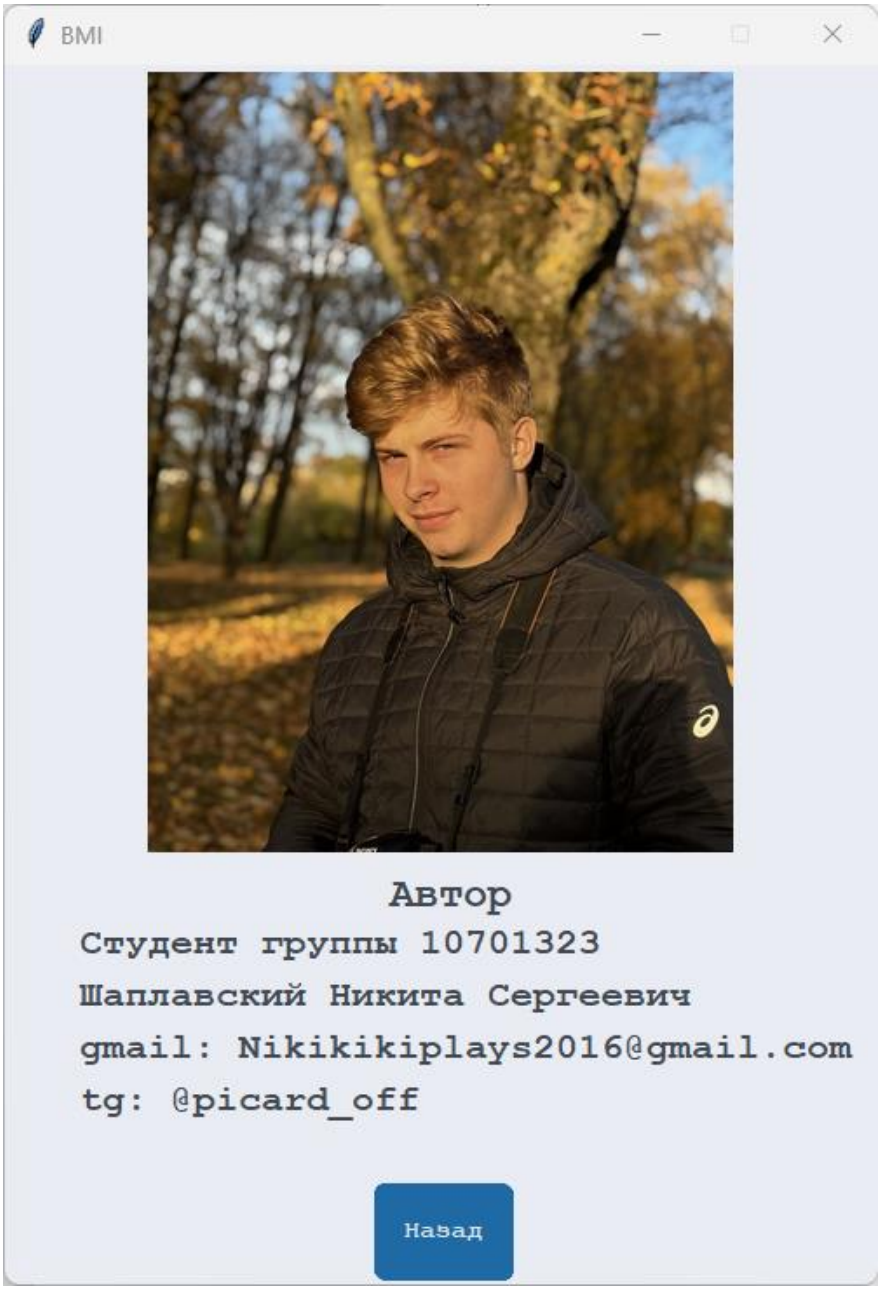

Рисунок 7.4 – Окно об авторе

е. **О программе**

Окно “О программе” (см. рисунок 7.5) отображает краткую информацию о приложении. Содержит главную информацию о программе.

Рисунок 7.5 – Окно “О программе”

Шаплавский Никита Сергеевич, гр.10701323

8 Дополнительные опции

1. Добавление своего языка в пакет

Языковой пакет в этой программе сост. из словарей в файле language_pack.json
Для добавление нового языка необходимо

1. Открыть файл language_pack.json в любом поддерживающем редакторе

2. Найти ключь language_list, она находится на 3-й строке и представляет собой список ключей для словарей с переводом

```
"language_list": [
    "ru",
    "en"
],
```

3. Добавить в него название своего будущего “языкового пакета”
   Добавим к примеру tr

```
"language_list": [
    "ru",
    "en",
    "tr",
],
```

4. Теперь копируем словарь с ключем “en”
   Он выглядит так:

```
"en": {
    "height": "height",
    "weight": "weight",
    "age": "age",
    "sex": "sex",
    "calculate": "Calculate",
    "change theme:": "Change theme: ",
    "light": "Light",
    "dark": "Dark",
    "lagoon": "Lagoon",
    "change language": "Change language",
    "Man": "man",
    "Woman": "woman",
    "bmi comment": [
        [
            16,
            "Severe body weight deficiency"
        ],
        [
            18.5,
            "Underweight"
        ],
        [
            25,
            "Normal body weight"
        ],
        [
            30,
            "pre-obesity"
        ],
        [
            35,
            "1st degree obesity"
        ],
        [
            40,
            "Obesity of the 2nd degree"
        ],
        [
            100,
            "Obesity of the 3rd degree"
```

```
        ]
    ],
    "man": "Man",
    "woman": "Woman",
    "History": "History",
    "Clear": "Clear",
    "ideal weight": "Ideal weight",
    "author": "Author",
    "group": "student group 10701323",
    "name": "Shaplauski Mikita Sergeevich",
    "back": "Back",
    "help": "Help",
    "about": "About",
    "menu": "Menu",
    "bmi": "Body mass index",
    "program_allows": "The program allows",
    "allows1": "1. Calculate body mass index",
    "allows2": "2. Calculate 'ideal' weight",
    "allows3": "3. View calculation history",
    "allows4": "4. Calculate basal metabolic rate",
    "cm_help": "Enter your height in cm",
    "age_help": "Enter your age (years)",
    "kg_help": "Enter your weight in kg",
    "ex num": "must be a number",
    "ex min": "must be greater than 0",
    "ex empty": "Fill in all fields"
}
```

5. Заменяем ключ “en” на добавленный нами в language_list, в моем слечае “tr”

6. Заменяем все значения ключей(в этом словаре) на наш перевод

7. Вставляем полученный словарь после последнего словаря, ставя запятаю

8. Сохраняем и заходим в программу

9. Переходим в окно настройки и в combobox “Изменения языка” выбираем ключь которым вы назвали свой “языковой пакет” после чего в приложении меняется язык (рис 8.1)

Шаплавский Никита Сергеевич, гр.10701323

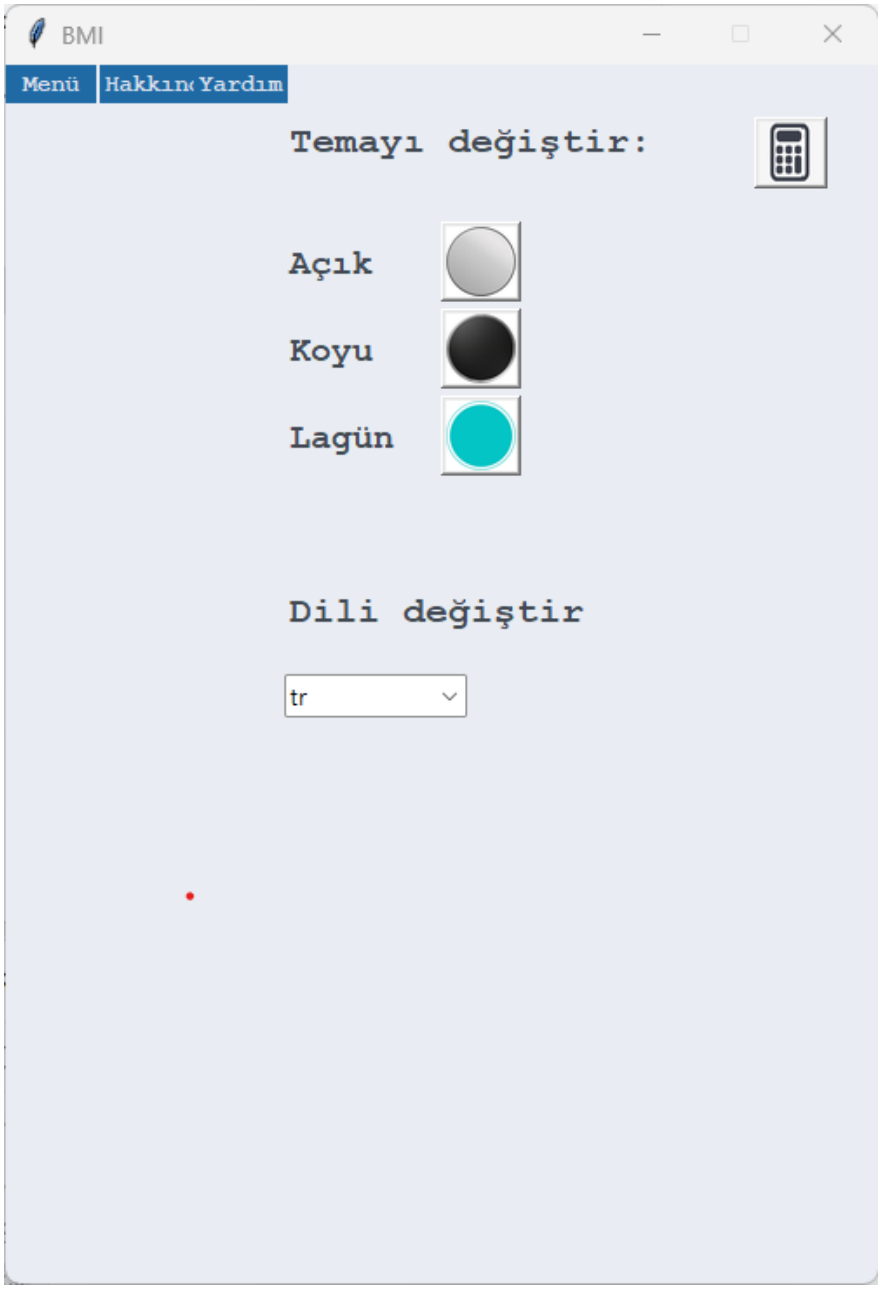

Рисунок 8.1 – Окно настроек с измененным языком

Шаплавский Никита Сергеевич, гр.10701323

Заключение

В ходе выполнения проекта, посвященного разработке приложения для расчета индекса массы тела (BMI), базального метаболизма (BMR) и идеального веса, было создано приложение на языке программирования Python, предназначенное для вычисления основных параметров здоровья человека.

Целью данной работы было создание инструмента, который позволял бы пользователям легко и быстро определять параметры здоровья и обладал удобным интерфейсом для выполнения вычислений. В результате были созданы функции для вычисления ключевых характеристик, таких как BMI, BMR и идеальный вес. Эти функции были интегрированы в графический интерфейс приложения, что позволило пользователям получить результаты вычислений с минимальными усилиями.

Были использованы различные библиотеки Python, такие как tkinter для создания интерфейса, а также другие модули для удобства и эффективности выполнения вычислений.

Этот проект не только помог в создании программного решения для вычисления параметров здоровья, но и дал возможность понять и применить теоретические концепции на практике. Полученное приложение может быть полезным инструментом для всех, кто интересуется здоровьем и физической формой.

Таким образом, выполнение этого проекта дало нам не только практические навыки программирования на Python, но и погрузило в тему здоровья и физической формы, позволив применить полученные знания для создания полезного программного продукта.

Шаплавский Никита Сергеевич, гр.10701323

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Python.org [Электронный ресурс] / Официальная документация по языку программирования Python. – Режим доступа: https://www.python.org/doc/.
Дата доступа: 02.12.2024

2. Хетланд, М. Л. Начинаем с Python: от новичка к профессионалу / М. Л. Хетланд. – Издательство Apress, 2014. – 559 с.

3. Stacoverflow [Электронный ресурс] – Режим доступа: https://stackoverflow.com/
Дата доступа: 02.12.2024

4. habr [Электронный ресурс] – Режим доступа: https://qna.habr.com/
Дата доступа: 02.12.2024

5. Calorizator [Электронный ресурс] –
Режим доступа: https://calorizator.ru/article/body/bmr-calculation
Дата доступа: 02.12.2024

6. Pypi [Электронный ресурс] – Режим доступа: https://pypi.org/
Дата доступа: 02.12.2024

7. Phythonist[Электронный ресурс] – Режим доступа: https://pythonist.ru/
Дата доступа: 02.12.2024

8. Git Hub[Электронный ресурс] – Режим доступа: https://github.com/
Дата доступа: 02.12.2024

9. Твое питание[Электронный ресурс] –
Режим доступа: https://www.yournutrition.ru/weight/formula-kreffa/
Дата доступа:02.12.2024

Приложение А

Код программы:

Основной модуль программы **main.py**:

```
from tkinter import *
from tkinter import ttk
from tkinter.messagebox import showinfo

from customtkinter import CTkButton

from sys import exit
import sqlite3
from json import load, dump
import os
from PIL import Image, ImageTk
import ctypes

from modules.calculation import Human
from modules.themes import get_themes_settings, change_themes
from modules.db_control import DBManager

from my_tabs.tittle_tab import TitleTab
from my_tabs.help_tab import HelpTab
from my_tabs.about_tab import AboutTab



# Установка фиксированного DPI для приложения
try:
    ctypes.windll.shcore.SetProcessDpiAwareness(1)  #
DPI_AWARENESS_CONTEXT_SYSTEM_AWARE
except Exception:
    pass

def get_language_pack():
    global language_pack
    global text
    with open("data/language_pack.json", "r", encoding="utf-8") as jsfile:
        language_pack = load(jsfile)
        text = language_pack[language_pack["language"]]

get_language_pack()

db_manager = DBManager("data/base.db", text)


def save_language_pack(lang_pack):
    with open("data/language_pack.json", "w", encoding="utf-8") as jsfile:
        dump(lang_pack, jsfile, indent=4)

def change_bmr_label(height: str, weight: str, age: str, sex: str, bmr_label:
Label,
                     bmi_label:Label, bmi_comment_label: Label, error_label: Label,
                     text_box: Text, ideal_weight_label: Label):
    bmi_label.config(text="")
    bmr_label.config(text="")
    bmi_comment_label.config(text="")
    error_label.config(text="")
    ideal_weight_label.config(text="")
    try:
        sex = text[sex]
    except KeyError:
        error_label.config(text=text["ex_empty"])
        return 0

    try:
```

```
            human = Human(height, weight, age, sex, text)
            bmr_label.config(text=f"BMR:{human.bmr}")
            bmi_label.config(text=f"BMI:{human.bmi}")
            check = True
            for i in text['bmi_comment']:
                if human.bmi <= i[0]:
                    bmi_comment_label.config(text=i[1].rjust(15))
                    check = False
                    break
            if check:
                bmi_comment_label.config(text=(text['bmi_comment'][6][1]).rjust(15))


ideal_weight_label.config(text=f"{(text['ideal_weight']).rjust(12)}:{human.ideal_we
ight}")
        except ValueError as ex:
            error_label.config(text=ex)
            return 0
        except TypeError as ex:
            error_label.config(text=ex)
            return 0
        db_manager.write_data((sex, height, weight, age, human.bmi, human.bmr))
        text_box.insert('3.0', db_manager.get_last_record())

class MainTab():
    def __init__(self, tab: Tk, font: dict) -> None:
        """
        Initialization main frame,
        """
        self.tab = tab
        self.font = font
        self.tooltip = None

        self.__create_item()

    def __create_item(self):
        self.label_arr = []        #Список для сбора всех Label
        self.label_place_arr = [] #Список для сбора расположения Label

        self.label_arr.append(Label(self.tab, text="BMI", font=self.font["h1"]))
        self.label_place_arr.append([220, 30])


        self.label_arr.append(Label(self.tab, text=(text["height"])[:6],
font=self.font["p"]))
        self.label_place_arr.append([136, 100])
        self.height_entry = Entry(self.tab, width=8)
        self.height_entry.bind("<Enter>", lambda e: self.show_tooltip(e,
text["cm_help"]))
        self.height_entry.bind("<Leave>", lambda e: self.hide_tooltip())

        self.label_arr.append(Label(self.tab, text=(text["age"])[:7],
font=self.font["p"]))
        self.label_place_arr.append([293,100])
        self.age_entry = Entry(self.tab, width=9)
        self.age_entry.bind("<Enter>", lambda e: self.show_tooltip(e,
text["age_help"]))
        self.age_entry.bind("<Leave>", lambda e: self.hide_tooltip())


        self.label_arr.append(Label(self.tab, text=(text["weight"])[:6],
font=self.font["p"]))
        self.label_place_arr.append([136, 170])
        self.weight_entry = Entry(self.tab, width=8)
        self.weight_entry.bind("<Enter>", lambda e: self.show_tooltip(e,
text["kg_help"]))
        self.weight_entry.bind("<Leave>", lambda e: self.hide_tooltip())
```

```
        self.label_arr.append(Label(self.tab, text=(text["sex"])[:6],
font=self.font["p"]))
        self.label_place_arr.append([293, 170])
        self.sex_combobox = ttk.Combobox(self.tab, values=[text["man"],
text["woman"]], state="readonly", width=7)

        self.bmr_label = Label(self.tab, font=self.font["h3"])
        self.bmi_label = Label(self.tab, font=self.font["h3"])
        self.bmi_comment_label = Label(self.tab, font=self.font["h4"])
        self.ideal_weight_label = Label(self.tab, font=self.font["h3"])

        self.error_label = Label(self.tab, font=self.font["h3"])

        self.calculate_button = CTkButton(
            self.tab,
            height=40,
            width=120,
            #fg_color="#dcdcdc",
            text=text["calculate"],
            command=lambda: change_bmr_label(
                self.height_entry.get(), self.weight_entry.get(),
                self.age_entry.get(), self.sex_combobox.get(),
                self.bmr_label, self.bmi_label, self.bmi_comment_label,
                self.error_label, self.text_box, self.ideal_weight_label
            ),
            corner_radius=6
        )


        self.label_arr.append(Label(self.tab, text=text["History"],
font=self.font["h3"]))
        self.label_place_arr.append([18, 400])

        self.scrollbar = Scrollbar(self.tab, width=18, orient=VERTICAL)
        self.text_box = Text(self.tab, font=self.font["h4"], width=44, height=12,
yscrollcommand=self.scrollbar.set)
        self.text_box.pack_propagate(False)

        self.clear_text_box_button = CTkButton(
            self.tab,
            height=20,
            width=60,
            #background="#cb8787",
            text=text["Clear"],
            command=self.clear_text_box_button,
            corner_radius=6
        )

        self.scrollbar.config(command=self.text_box.yview)

        self.text_box.insert('1.0', str(db_manager))

    def show_tooltip(self, event, text):
        self.tooltip = Toplevel(self.tab)
        self.tooltip.wm_overrideredirect(True)
        self.tooltip.wm_geometry(f"+{event.x_root + 10}+{event.y_root + 10}")
        label = Label(self.tooltip, text=text, relief="solid", borderwidth=1)
        label.pack()

    def hide_tooltip(self):
        if self.tooltip:
            self.tooltip.destroy()
            self.tooltip = None

    def show_items(self):
        for i, item in enumerate(self.label_arr):
            item.place(x=self.label_place_arr[i][0], y=self.label_place_arr[i][1])
```

```
        self.height_entry.place(x=140, y=130)
        self.age_entry.place(x=297, y=130)
        self.weight_entry.place(x=140, y=200)
        self.sex_combobox.place(x=297,y=200)

        self.bmr_label.place(x=180, y=283)
        self.bmi_label.place(x=180, y=313)
        self.bmi_comment_label.place(x=140, y=340)
        self.ideal_weight_label.place(x=140, y=370)

        self.error_label.place(x=140, y=285)
        self.calculate_button.place(x=160,y=205)#(x=200, y=380)

        #self.scrollbar.place(x=483, y=440.1)
        self.clear_text_box_button.place(x=354, y=355)
        self.text_box.place(x=6, y=435)

    def clear_text_box_button(self):
        self.text_box.delete("1.0", END)
        db_manager.delete_data()
        self.text_box.insert('1.0', str(db_manager))

class SettingsTab():
    def __init__(self, tab: Tk, font: dict, main_app) -> None:
        """
        Initialization settings frame
        """
        self.main_app = main_app
        self.tab = tab
        self.font = font

        light_img = Image.open("image/light_ico.png").convert("RGBA")
        dark_img = Image.open("image/dark_ico.png").convert("RGBA")
        lagoon_img = Image.open("image/lagoon_ico.png").convert("RGBA")

        light_img_resized = light_img.resize((40, 40), Image.Resampling.LANCZOS)
        dark_img_resized = dark_img.resize((40, 40), Image.Resampling.LANCZOS)
        lagoon_img_resized = lagoon_img.resize((40, 40), Image.Resampling.LANCZOS)

        self.light_img = ImageTk.PhotoImage(light_img_resized)
        self.dark_img = ImageTk.PhotoImage(dark_img_resized)
        self.lagoon_img = ImageTk.PhotoImage(lagoon_img_resized)

        self.__create_item()

    def __create_item(self) -> None:
        self.label_arr = [
            Label(self.tab, text=text["change theme:"], font=self.font['h3']),
            Label(self.tab, text=text["light"], font=self.font['p']),
            Label(self.tab, text=text["dark"], font=self.font['p']),
            Label(self.tab, text=text["lagoon"], font=self.font['p']),
            Label(self.tab, text=text["change language"], font=self.font["h3"])
        ]
        self.label_arr_place = [[160, 30], [160, 100], [160, 150], [160, 200],
[160, 300]]

        self.button_arr = [
            Button(
                self.tab,
                image=self.light_img,
                command=lambda: change_themes('light', self.tab)
            ),
            Button(
                self.tab,
                image=self.dark_img,
                command=lambda: change_themes('dark', self.tab)
            ),
            Button(
```

```
                self.tab,
                image=self.lagoon_img,
                command=lambda: change_themes('lagoon', self.tab)
            )
        ]

        self.button_arr_place = [[250, 90], [250, 140], [250, 190]]

        self.language_combobox = ttk.Combobox(self.tab,
values=language_pack["language_list"], state="readonly", width=10)
        self.language_combobox.set(language_pack["language"])
        self.language_combobox.bind("<<ComboboxSelected>>", self.chenge_language)

    def chenge_language(self, event=None):
        lang = self.language_combobox.get()
        language_pack["language"] = lang
        save_language_pack(language_pack)
        self.main_app.reboot()

    def show_items(self):
        for i, item in enumerate(self.label_arr):
            item.place(x=self.label_arr_place[i][0], y=self.label_arr_place[i][1])

        for i, item in enumerate(self.button_arr):
            item.place(x=self.button_arr_place[i][0],
y=self.button_arr_place[i][1])

        self.language_combobox.place(x=160, y=350)

class MainApp:
    def __init__(self):
        self.window = Tk()
        self.window.geometry("500x700")
        self.window.resizable(False, False)
        self.window.title("BMI")

        self.themes_settings = get_themes_settings()
        self.font = self.themes_settings['font']

        self.check_tittle = True

        calculator_img = Image.open("image/calculator.png").convert("RGBA")
        settings_img = Image.open("image/settings.png").convert("RGBA")

        calculator_img_resized = calculator_img.resize((35, 35),
Image.Resampling.LANCZOS)
        settings_img_resized = settings_img.resize((35, 35),
Image.Resampling.LANCZOS)

        image_calculator = ImageTk.PhotoImage(calculator_img_resized)
        image_settings = ImageTk.PhotoImage(settings_img_resized)

        self.font = get_themes_settings()['font']


        self.bool_check_app = True
        self.bool_dict_image = {
            True: image_settings,
            False: image_calculator
        }

        self.main_tab = MainTab(self.window, self.font)
        self.setting_tab = SettingsTab(self.window, self.font, self)

        self.title_tab = TitleTab(self.window, self.font, self)
        self.about_tab = AboutTab(self.window, self.font, text, self)
        self.help_tab = HelpTab(self.window, self.font, text, self)
```

```
        #self.main_tab.show_items()
        self.title_tab.show_items()
        self.create_main_app()

        change_themes(self.themes_settings['now_mode'], self.window)

    def create_main_app(self):
        self.button_change_app = Button(
            self.window,
            image=self.bool_dict_image[True],
            command=self.change_app
        )
        self.title_button = CTkButton(
            self.window,
            height=10,
            width=45,
            text=text["menu"],
            command=self.open_title,
            corner_radius=0,
            font=tuple(self.font["h4"])
        )
        self.about_autor_button = CTkButton(
            self.window,
            height=10,
            width=45,
            text=text["about"],
            command=self.open_about,
            corner_radius=0,
            font=tuple(self.font["h4"])
        )
        self.help_button = CTkButton(
            self.window,
            height=10,
            width=45,
            text=text["help"],
            command=self.open_help,
            corner_radius=0,
            font=tuple(self.font["h4"])
        )

    def update_themes_settings(self):
        self.themes_settings = get_themes_settings()

    def reboot(self):
        self.hide_widgets()
        del self.main_tab
        del self.setting_tab
        del self.about_tab
        del self.help_tab

        get_language_pack()
        db_manager.set_language_pack(text)
        self.main_tab = MainTab(self.window, self.font)
        self.setting_tab = SettingsTab(self.window, self.font, self)
        self.title_tab = TitleTab(self.window, self.font, self)
        self.about_tab = AboutTab(self.window, self.font, text, self)
        self.help_tab = HelpTab(self.window, self.font, text, self)

        self.create_main_app()

        self.setting_tab.show_items()
        self.show_main_app()
        self.update_themes_settings()
        change_themes(self.themes_settings['now_mode'], self.window)

    def show_main_app(self):
        self.button_change_app.place(x=430, y=30)
```

```
        self.title_button.place(x=0,y=0)
        self.about_autor_button.place(x=47, y=0)
        self.help_button.place(x=94, y=0)

    def hide_widgets(self):
        # Скрываем все виджеты в окне
        for widget in self.window.winfo_children():
            widget.place_forget()

    def change_app(self):
        self.hide_widgets()
        self.show_main_app()
        if self.bool_check_app:
            self.setting_tab.show_items()
            self.bool_check_app = False

self.button_change_app.config(image=self.bool_dict_image[self.bool_check_app])
        else:
            self.main_tab.show_items()
            self.bool_check_app = True

self.button_change_app.config(image=self.bool_dict_image[self.bool_check_app])

    def choese_tab(self, tab_name: str) -> None:
        self.hide_widgets()
        self.show_main_app()
        self.show_main_app()
        self.check_tittle = False
        if tab_name == "main":
            self.main_tab.show_items()
            self.bool_check_app = True

self.button_change_app.config(image=self.bool_dict_image[self.bool_check_app])
        elif tab_name == "settings":
            self.setting_tab.show_items()
            self.bool_check_app = False

self.button_change_app.config(image=self.bool_dict_image[self.bool_check_app])
        else:
            raise NameError

    def open_title(self):
        self.check_tittle = True
        self.hide_widgets()
        self.title_tab.show_items()

    def open_about(self):
        self.hide_widgets()
        self.about_tab.show_items()

    def open_help(self):
        self.hide_widgets()
        self.help_tab.show_items()

    def loop(self):
        self.window.mainloop()

main_app = MainApp()
main_app.loop()
```

модуль программы **about_tab.py**:

```
from tkinter import *
from customtkinter import CTkButton
from PIL import Image, ImageTk


class AboutTab:
```

```
def __init__(self, tab: Tk, font: dict, lp, main_app):
    self.tab = tab
    self.font = font
    self.main_app = main_app

    self.lp = lp

    img = Image.open("image/me.jpg").convert("RGBA")

    percent = 35
    width, height = img.size
    new_width = int(width * percent / 100)
    new_height = int(height * percent / 100)

    # Изменяем размер изображения
    resized_img = img.resize((new_width, new_height), Image.Resampling.LANCZOS)

    self.my_photo = ImageTk.PhotoImage(resized_img)
    self.__create_item()

def __create_item(self):
    self.calculator_tab_button = CTkButton(
        self.tab,
        width=70,
        height=50,
        text=self.lp["back"],
        command=lambda: self.main_app.choese_tab("main"),
        corner_radius=6,
        font=("Courier", 13, "bold")
    )
    self.my_photo_label = Label(image=self.my_photo)

    self.labels = [
        Label(
            self.tab,
            text=self.lp["author"],
            font=self.font["h2"]
        ),
        Label(
            self.tab,
            text=self.lp["group"],
            font=self.font["h3"]
        ),
        Label(
            self.tab,
            text=self.lp["name"],
            font=self.font["h3"]
        ),
        Label(
            self.tab,
            text="gmail: Nikikikiplays2016@gmail.com",
            font=self.font["h3"]
        ),
        Label(
            self.tab,
            text="tg: @picard_off",
            font=self.font["h3"]
        )
    ]
    self.places = [[218, 460], [40, 490], [40, 520], [40, 550], [40, 580]]
def show_items(self):
    self.my_photo_label.place(x=80, y=2)
    self.calculator_tab_button.place(x=185, y=560)

    for i,lab in enumerate(self.labels):
        lab.place(x=self.places[i][0], y=self.places[i][1])
```

модуль программы **help_tab.py**:

```
from tkinter import *
from customtkinter import CTkButton
from PIL import Image, ImageTk


class HelpTab:
    def __init__(self, tab: Tk, font: dict, lp, main_app):
        self.tab = tab
        self.font = font
        self.main_app = main_app

        self.lp = lp

        img = Image.open("image/help.jpg").convert("RGBA")

        percent = 50
        width, height = img.size
        new_width = int(width * percent / 100)
        new_height = int(height * percent / 100)

        # Изменяем размер изображения
        resized_img = img.resize((new_width, new_height), Image.Resampling.LANCZOS)

        self.help_photo = ImageTk.PhotoImage(resized_img)
        self.__create_item()

    def __create_item(self):

        self.help_photo_label = Label(image=self.help_photo)

        self.labels = [
            Label(
                self.tab,
                text=self.lp["bmi"],
                font=self.font["h2"]
            ),
            Label(
                self.tab,
                text=self.lp["program_allows"]+":",
                font=self.font["h2"]
            ),
            Label(
                self.tab,
                text=self.lp["allows1"],
                font=self.font["h3"]
            ),
            Label(
                self.tab,
                text=self.lp["allows2"],
                font=self.font["h3"]
            ),
            Label(
                self.tab,
                text=self.lp["allows3"],
                font=self.font["h3"]
            ),
            Label(
                self.tab,
                text=self.lp["allows4"],
                font=self.font["h3"]
            )
        ]
        self.places = [
            (20, 20),
            (20, 300),
            (20, 350),
            (20, 375),
```

```
            (20, 400),
            (20, 425)
        ]

        self.calculator_tab_button = CTkButton(
            self.tab,
            width=70,
            height=50,
            text=self.lp["back"],
            command=lambda: self.main_app.choese_tab("main"),
            corner_radius=6,
            font=("Courier", 13, "bold")
        )

    def show_items(self):
        self.help_photo_label.place(x=20, y=70)
        for i, label in enumerate(self.labels):
            label.place(x=self.places[i][0], y=self.places[i][1])
        self.calculator_tab_button.place(x=185, y=560)
```

модуль программы **tittle_tab.py**:

```
from tkinter import *
from customtkinter import CTkButton
from PIL import Image, ImageTk



class TitleTab:
    def __init__(self, tab: Tk, font: dict, main_app):
        self.tab = tab
        self.font = font
        self.main_app = main_app
        self.inactivity_time = 60000  # 60 секунд
        self.inactivity_timer = None
        self.reset_inactivity_timer()

        # Отслеживание активности пользователя
        self.tab.bind_all("<Any-KeyPress>", self.reset_inactivity_timer)
        self.tab.bind_all("<Motion>", self.reset_inactivity_timer)

        img = Image.open("image/logo.png").convert("RGBA")

        percent = 30
        width, height = img.size
        new_width = int(width * percent / 100)
        new_height = int(height * percent / 100)

        # Изменяем размер изображения
        resized_img = img.resize((new_width, new_height), Image.Resampling.LANCZOS)

        self.logo_photo = ImageTk.PhotoImage(resized_img)

        self.__create_item()

    def __create_item(self):
        self.plases = [[10, 30], [15, 50], [10, 70], [150, 170], [70, 200], [85,
230], [5, 480], [5, 500], [5, 530],
                       [5, 550], [200, 600]]
        self.labels = [Label(
            self.tab,
            text="Белорусский Национальный Технический Университет",
            font=("Courier", 11, "bold")
        ),

            Label(
                self.tab,
                text="Факультет информационных технологий и робототехники",
                font=("Courier", 10, "bold")
```

```
            ),

            Label(
                self.tab,
                text="Кафедра программного обеспечения информационных систем и
технологий",
                font=("Courier", 8, "bold")
            ),

            Label(
                self.tab,
                text="Курсовая работа",
                font=("Courier", 13, "bold")
            ),

            Label(
                self.tab,
                text="по дисциплине 'Языки программирования'",
                font=("Courier", 11, "bold")
            ),
            Label(
                self.tab,
                text="Калькулятор индекса массы тела",
                font=("Courier", 12, "bold")
            ),

            Label(
                self.tab,
                text="Выполнил: студент группы 10701323",
                font=("Courier", 11, "bold")
            ),

            Label(
                self.tab,
                text="Шаплавский Никита Сергеевич",
                font=("Courier", 11, "bold")
            ),

            Label(
                self.tab,
                text="Преподаватель: к.ф.-м.н., доц.",
                font=("Courier", 11, "bold")
            ),

            Label(
                self.tab,
                text="Сидорик Валерий Владимирович",
                font=("Courier", 11, "bold")
            ),

            Label(
                self.tab,
                text="Минск 2024",
                font=("Courier", 11, "bold")
            )]
        self.logo_photo_label = Label(image=self.logo_photo)
        self.calculator_tab_button = CTkButton(
            self.tab,
            width=70,
            height=50,
            text="Перейти в калькулятор",
            command=lambda: self.main_app.choese_tab("main"),
            corner_radius=6,
            font=("Courier", 13, "bold")
        )

        self.exit_button = CTkButton(
            self.tab,
```

```
            width=70,
            height=50,
            text="Выйти",
            command=exit,
            corner_radius=6,
            font=("Courier", 13, "bold")
        )

    def show_items(self):
        for i,label in enumerate(self.labels):
            label.place(x=self.plases[i][0], y=self.plases[i][1])

        self.calculator_tab_button.place(x=220, y=550)
        self.exit_button.place(x=50, y=550)
        self.reset_inactivity_timer()
        self.logo_photo_label.place(x=180, y=270)

    def reset_inactivity_timer(self, event=None):
        if self.inactivity_timer:
            self.tab.after_cancel(self.inactivity_timer)
        self.inactivity_timer = self.tab.after(self.inactivity_time,
self.close_app_due_to_inactivity)

    def close_app_due_to_inactivity(self):
        if self.main_app.check_tittle:  # Проверка, что активен title tab
            exit()
```

модуль программы **calculation.py**:

```
from tkinter import Entry, Label

class Human:
    def __init__(self, height: float, weight: float, age: int, sex: str, text: dir)
-> None:
        try:
            self.height = float(height)
        except:
            raise TypeError(f"{text['height']} {text['ex_num']}")
        if self.height <= 0:
           raise ValueError(f"{text['height']} {text['ex_min']}")


        try:
            self.weight = float(weight)

        except:
            raise TypeError(f"{text['weight']} {text['ex_num']}")
        if self.weight <= 0:
            raise ValueError(f"{text['weight']} {text['ex_min']}")


        try:
            self.age = int(age)
        except:
            raise TypeError(f"{text['age']} {text['ex_num']}")
        if self.age <= 0:
            raise ValueError(f"{text['age']} {text['ex_min']}")


        if sex == "man" or sex == "woman":
            self.sex = sex
        else:
            raise ValueError("Error in class human with 'sex'")

        self.bmr = self.__calculate_bmr()
        self.bmi = self.__calculate_bmi()
        self.ideal_weight = self.__calculate_ideal_weight()
```

```
    def __calculate_bmr(self):
        if self.sex == "man":
            bmr = 88.36 + (13.4 * self.weight) + (4.8 * self.height) - (5.7 *
self.age)
        else:
            bmr = 447.6 + (9.2 * self.weight) + (3.1 * self.height) - (4.3 *
self.age)
        return round(bmr, 2)

    def __calculate_bmi(self):
        return round(self.weight / ((self.height/100)**2), 2)

    def __calculate_ideal_weight(self):
        if self.sex == "man":
            return round(0.9 * (self.height - 100) + (self.age / 10), 1)
        else:
            return round(0.9 * (self.height - 100) + (self.age / 10) * 0.9, 1)

    def __str__(self):
        return f"sex: {self.sex}, height: {self.height}, weight: {self.weight},
age: {self.age}, BMI:{self.bmi}, BMR: {self.bmr}"

    def get_data(self):
        return (self.sex, self.height, self.weight, self.age, self.bmi, self.bmr)
```

модуль программы **db_control.py**:

```
import sqlite3


class DBManager:
    def __init__(self, url: str, language_pack: dir):
        self.conn = sqlite3.connect(url)
        self.cursor = self.conn.cursor()
        self.__init_db()
        self.text = language_pack
        self.max_index = 0

    def __init_db(self):
        create_table_query = """
        CREATE TABLE IF NOT EXISTS health_data (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            sex TEXT NOT NULL,
            height INTEGER NOT NULL,
            weight INTEGER NOT NULL,
            age INTEGER NOT NULL,
            BMI REAL,
            BMR REAL
        );
        """
        self.cursor.execute(create_table_query)
        self.conn.commit()

    def set_language_pack(self, language_pack: dir):
        self.text = language_pack

    def get_data(self) -> tuple:
        self.cursor.execute("SELECT sex, height, weight, age, BMI, BMR FROM
health_data")
        data = self.cursor.fetchall()
        return data

    def write_data(self, data: tuple) -> None:
        insert_query = """
        INSERT INTO health_data (sex, height, weight, age, BMI, BMR)
        VALUES (?, ?, ?, ?, ?, ?);
        """
```

```
        self.cursor.execute(insert_query, data)
        self.conn.commit()


    def __str__(self):
        data = self.get_data()

        string = "id  " + self.text['sex'].ljust(6)[:6] +
self.text['height'].ljust(7)[:7]
        string += self.text['weight'].ljust(6)[:6] + self.text['age'].ljust(8)[:8]
+ "BMI".ljust(7) + "BMR" + "\n"
        string += "\u0305"*44 + "\n"
        self.max_index= len(data)
        for indx, row in enumerate(reversed(data)):
            string += str(self.max_index - indx).ljust(4)

            string += row[0].ljust(7)[:7]
            string += str(row[1]).ljust(6)
            string += str(row[2]).ljust(6)
            string += str(row[3]).ljust(8)
            string += str(round(row[4],1)).ljust(7)[:7]
            string += str(round(row[5],1)).ljust(6)[:6]
            """for i in row:
                try:
                    string += str(round(i,2)).ljust(6)[:7]
                except TypeError:
                    string += i.ljust(7)[:7]"""

            string += "\n"


        return string


    def delete_data(self) -> None:
        delete_query = "DELETE FROM health_data;"
        self.cursor.execute(delete_query)
        self.__init_db()


    def get_last_record(self) -> str:
        self.cursor.execute("SELECT sex, height, weight, age, BMI, BMR FROM
health_data ORDER BY id DESC LIMIT 1;")
        row = self.cursor.fetchone()

        string = ""
        string += str(self.max_index + 1).ljust(4)
        string += row[0].ljust(7)[:7]
        string += str(row[1]).ljust(6)
        string += str(row[2]).ljust(6)
        string += str(row[3]).ljust(8)
        string += str(round(row[4], 1)).ljust(7)[:7]
        string += str(round(row[5], 1)).ljust(6)[:6]
        string += "\n"
        self.max_index += 1
        return string
```

модуль программы **themes.py**:

```
import json
from tkinter import *
from tkinter import Label
from json import load, dump
```

```python
def get_themes_settings() -> dict:
    with open("data/themes.json", "r") as jsfile:
        themes_settings = load(jsfile)
    return themes_settings

def save_themes_settings(settings: dict) -> None:
    with open("data/themes.json", "w") as jsfile:
        dump(settings, jsfile, indent=4)


def change_themes(mode: str, window: Tk) -> None:
    themes_settings = get_themes_settings()

    text_color = themes_settings['mode'][mode]['text-color']
    window_color = themes_settings['mode'][mode]['window-color']

    for widget in window.winfo_children():
        if isinstance(widget, Label):
            widget.config(fg=text_color, bg=window_color)

    window.config(bg=window_color)
    themes_settings['now_mode'] = mode
    save_themes_settings(themes_settings)
```