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

**Оглавление**

[Введение…………………………………………………………………...……3](#_Toc1)

2. [Постановка и описание задачи………………………………………………4](#_Toc152585790)

3. Обзор предметной области…………………………………………………..6

4. Построение математической модели……………………………………….7

5. UML Диаграммы……………………………………………………………..9

6. Алгоритм работы программы……………………………………………...11

7. Описание работы графических окон………………………………………13

7.1 Стартовое окно TittleTab………………………………………….13

7.2 Главное окно MainTab…………………………………………….14

7.3 Окно настроек ……………………………………………………..16

7.4 Об авторе…………………………………………………………..17

7.5 О программе……………………………………………………….18

8. Дополнительные опции…………………………………………………….19

ЗАКЛЮЧЕНИЕ……………………………………………………………… .22

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ………………………… 23

Приложение А…………………………………………………………………24

# 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.

2 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.

1. **Проверка вводимых значений**:

-Обеспечение проверки корректности и правильности ввода данных перед расчетом параметров человека.

1. **Сохранение значений**:

-Приложение должно сохранять параметры которые пользовотель в вводит, а так же получает в ходе расчета

1. **Пояснение к результатам**

-Приложение должно комментировать телосложение человека

1. **Запись в файл и чтение из файла**:

-Запись сохранённых значений и результатов.

-Чтение значений и результатов из файла и вывод их.

1. **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

*Выходные данные*:

-Идеальная масса для человека

-Индекса массы тела

-Индекса базального обмена веществ

*Специальные требования*:

-Отсутствуют.

1. 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.

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:

1. **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 А прибавить произведение параметра B c весом, произведение С 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:

1. **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:

A=1

Значение параметров для женского пола пола:

A=0.9

Формула:

5 UML 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).

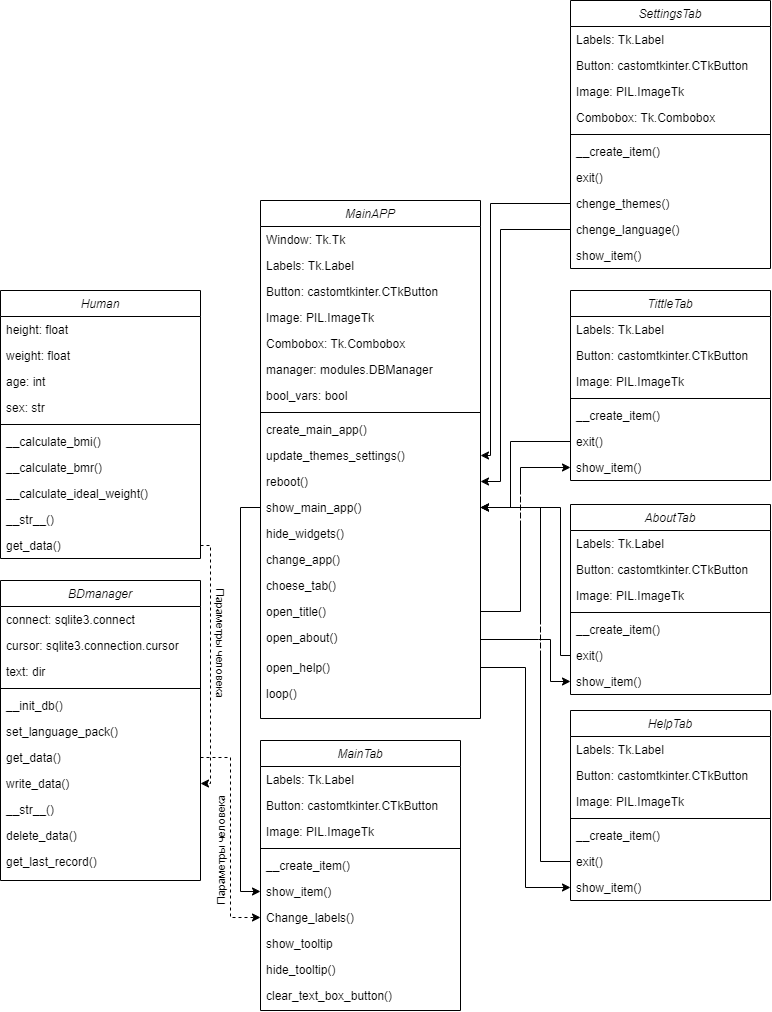


Рисунок 5.1 – UML диаграмма классов

**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".

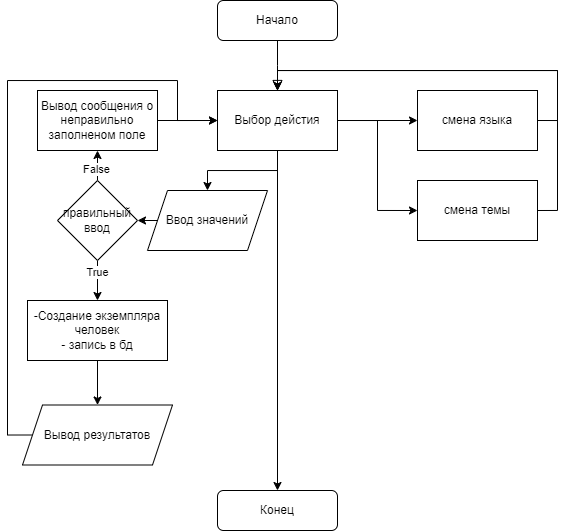


Рис. 6.1 Блок-схема работы программы

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):

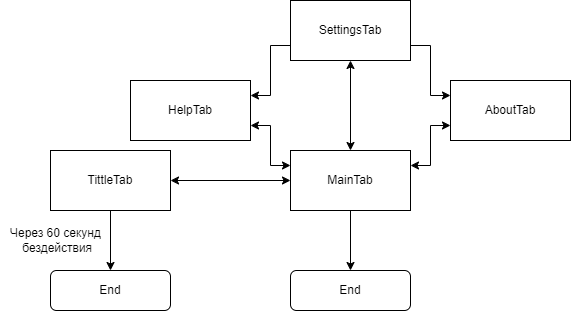


Рис. 6.2. Бизнес-логика переключения графических окон

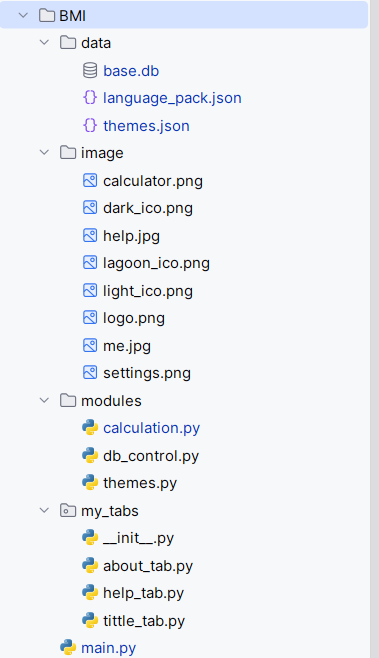


Рис. 6.3. Структура представления материала в электронном виде

1. **Описание работы графических окон**
   1. 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.

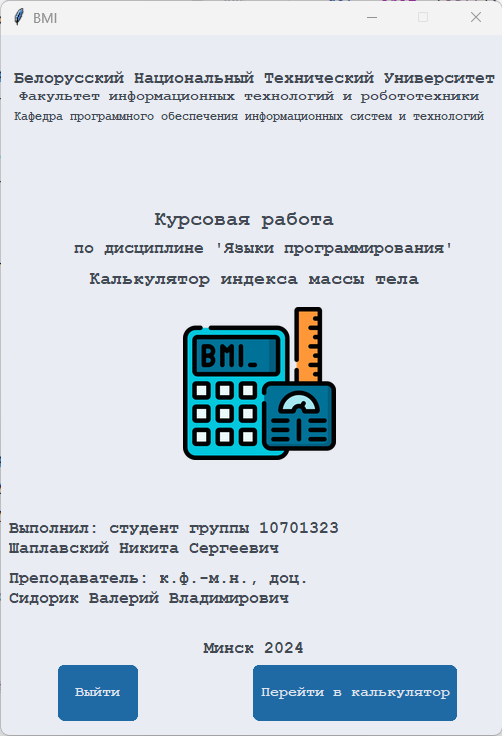
****

Рисунок 7.1 – Окно TittleTab

* 1. **Главное окно MainTab**

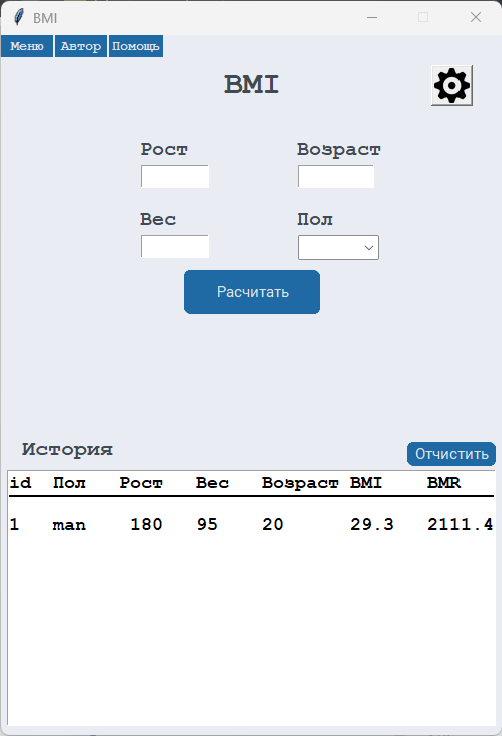
 Главное окно (см. рисунок 7.2.1) предоставляет возможность вводить параметры, рассчитывать BMI, BMR и идеальную массу тела, просматривать историю

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”

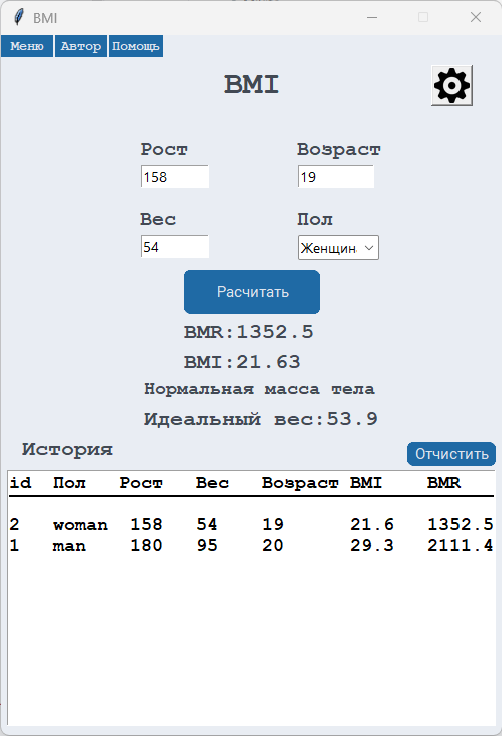
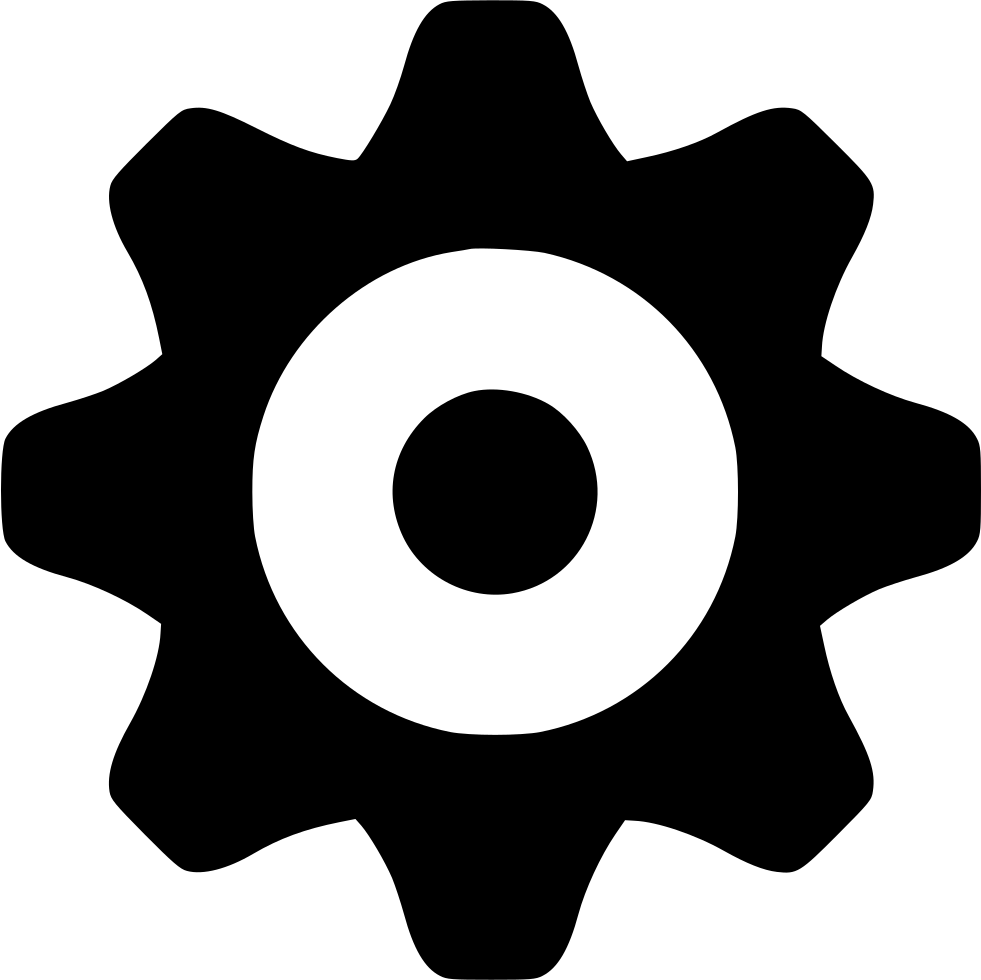


Рисунок 7.2.2 – Главное окно после ввода параметров

* 1. **Окно настроек SettingsTab**



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

1. Изменение темы

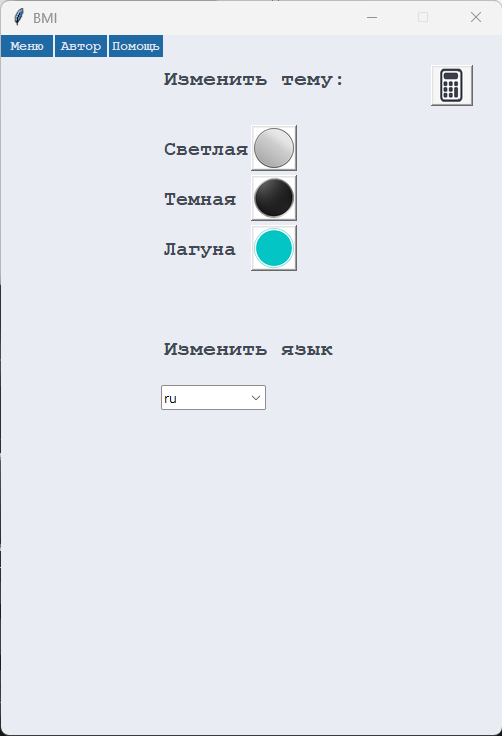
2. Изменение языка

Рисунок 7.3 – Окно настроек SettingsTab

* 1. **Об авторе**

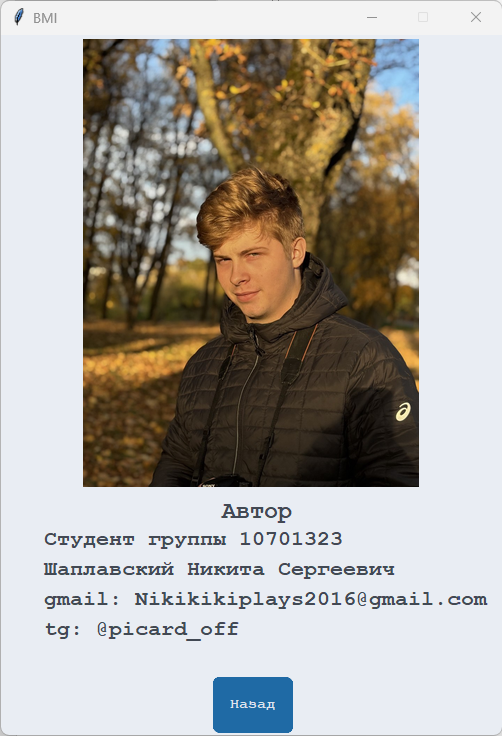
Окно “Об Авторе” (см. рисунок 7.4) содержит основную информацию об авторе проекта: номер группы в университете, фамилию, имя, отчество, адрес электронной почты, фотографию.

Рисунок 7.4 – Окно об авторе

* 1. **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”

1. **Дополнительные опции**
2. **Добавление своего языка в пакет**

Языковой пакет в этой программе сост. из словарей в файлe 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"  
],

1. Добавить в него название своего будущего “языкового пакета”

Добавим к примеру tr

"language\_list": [  
 "ru",  
 "en",

"tr",   
],

1. 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"  
}

1. Заменяем ключ “en” на добавленный нами в language\_list, в моем слечае “tr”
2. Заменяем все значения ключей(в этом словаре) на наш перевод
3. Вставляем полученный словарь после последнего словаря, ставя запятаю
4. Сохраняем и заходим в программу
5. 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)

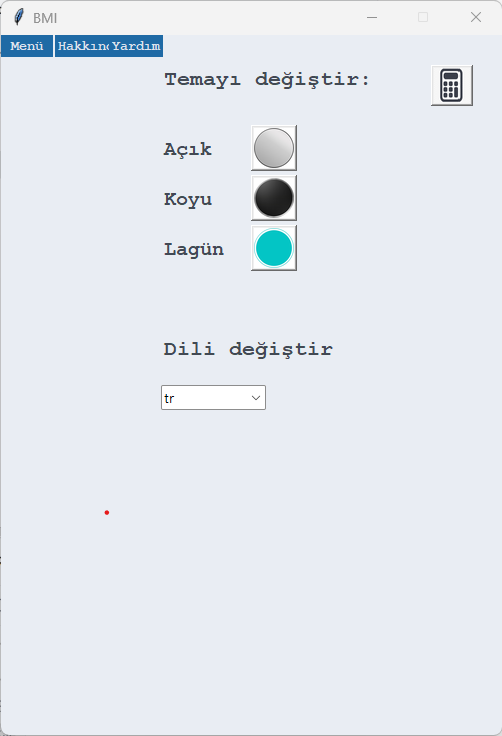


Рисунок 8.1 – Окно настроек с измененным языком

**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.

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

1. 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/](%20https://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 A

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\_weight}")  
 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  
  
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)