ДЕПАРТАМЕНТ ОБРАЗОВАНИЯ И НАУКИ ГОРОДА МОСКВЫ

Государственное бюджетное общеобразовательное учреждение

Города Москвы

Школа № 1502 «Энергия»

**Проект**

«От абака до суперкомпьютера»

Исполнитель: ученик 10Т класса

Светлов Александр

Руководитель: Кузьмина Е. Ф.

Москва, 2023

Оглавление

[**ВВЕДЕНИЕ** 3](#_Toc125351884)

[**1.** **Теоретическая часть** 4](#_Toc125351885)

[**1.1. Сравнение популярных языков программирования** 4](#_Toc125351886)

[**1.2. Преимущество создания приложений на Python** 5](#_Toc125351887)

[**1.3. Основные библиотеки для написания интерфейса** 5](#_Toc125351888)

[**1.4. Преимущества библиотеки PyQt5** 6](#_Toc125351889)

[**2. Технологический раздел** 7](#_Toc125351890)

[**2.1. Методы, библиотеки, классы** 7](#_Toc125351891)

[**2.2. Структура** 8](#_Toc125351892)

[**2.3. Руководство пользователя** 8](#_Toc125351893)

[**Заключение** 16](#_Toc125351894)

[Список литературы 17](#_Toc125351895)

[Приложение 18](#_Toc125351896)

# **ВВЕДЕНИЕ**

В настоящее время сложно представить себе повседневность без различных гаджетов, таких как смартфоны, планшеты, компьютеры и т.д., но далеко не все знают, как разрабатывались эти гаджеты, каковы были их предки. Хотя эта информация и доступна, но она разбросана по разным источникам и не скомпонована в легко доступной форме.

У современной молодежи наблюдается тенденция к вытеснению мышления так называемым клиповым мышлением. Клиповое мышление характеризуется тем, что человек не может длительно поддерживать сосредоточение на какой-либо мысли, а перескакивает с одного информационного повода на следующий. У подростков, да и во многом у остальной части населения, в качестве информационных поводов преимущественно выступают зрительные образы.

При наличии у человека клипового мышления усвоение им новой информации легче всего происходит в игровой форме, сопровождающейся большим массивом красочных изображений. Это правило касается любой информации. В особенности касается обучения в школе.

Далеко не каждый школьник, который заинтересуется историей создания компьютерной вычислительной техники пойдёт в магазин или в библиотеку, чтобы купить либо взять книгу, по интересующей его теме. Наверняка все сразу начнут вводить поисковые запросы в браузере и пытаться найти нужную информацию в красочной (с картинками) форме.

В помощь таким школьникам, а также в помощь учителям, можно было бы создать подборку информации по истории ЭВМ и в игровой форме в сопровождении графического материала подать её в виде сайта или в виде автономной программы.

Поскольку история создания ЭВМ включена в школьную программу, то ресурс, который поможет заинтересовавшимся школьникам изучить глубже информацию тем более актуален.

**Цель: со**здание компьютерной программы, в которой пользователю демонстрируется в игровой и графически привлекательной форме актуальная и правдивая информация по истории создания ЭВМ. Передача информации сопровождается элементами тестирования. Так же предоставить простой способ редактирования информации.

Для реализации цели необходимо решить следующие задачи:

* выбрать тему проекта;
* изучить материал;
* выбрать язык программирования;
* выбрать библиотеки для создания интерфейса;
* изучить выбранную библиотеку;
* написать прототип программы;
* произвести тестирование;
* исправить ошибки и недочеты;
* выпустить в релиз.

# **Теоретическая часть**

## **1.1. Сравнение популярных языков программирования**

|  |  |
| --- | --- |
| *Язык* | *Краткое описание* |
| *Python* | высокоуровневый язык программирования общего назначения с динамической строгой типизацией и автоматическим управлением памятью, ориентированный на повышение производительности разработчика, читаемости кода и его качества, а также на обеспечение переносимости написанных на нём программ. Язык является полностью объектно-ориентированным в том плане, что всё является объектами. |
| *C++* | компилируемый, статически типизированный язык программирования общего назначения. Поддерживает такие парадигмы программирования, как процедурное программирование, объектно-ориентированное программирование, обобщённое программирование. Язык имеет богатую стандартную библиотеку, которая включает в себя распространённые контейнеры и алгоритмы, ввод-вывод, регулярные выражения, поддержку многопоточности и другие возможности. C++ сочетает свойства как высокоуровневых, так и низкоуровневых языков. В сравнении с его предшественником — языком C — наибольшее внимание уделено поддержке объектно-ориентированного и обобщённого программирования. |
| *C#* | объектно-ориентированный язык программирования общего назначения. В основном используется для создания приложений на базе WPF и Windows Forms. Так же для сетевого программирования. |
| *Go* | компилируемый многопоточный язык программирования, разработанный внутри компании Google. Альтернатива C++, но является более простым и используется в основном на серверах и в веб -программировании. |
| *C* | компилируемый статически типизированный язык программирования общего назначения. Довольно старый язык, который использует почти машинные программы. Похож на *Ассемблер*. |
| *Rust* | мультипарадигменный компилируемый язык программирования общего назначения, сочетающий парадигмы функционального и процедурного программирования с объектной системой, основанной на типажах. |

## **1.2. Преимущество создания приложений на Python**

Основным преимуществом языка *Python* перед другими, представленными ранее является высокая скорость написания кода и его простота. Так же у языка Python есть огромное количество библиотек (полный список которых можно посмотреть на сайте <https://pypi.org/>), что поможет в написании моего приложения. Так же Python обладает высокой читабельностью кода, что поможет при дальнейшей поддержке проекта. На Python можно быстро написать почти любой проект. Так же, содержит множество инструментов для работы в разных сферах ­⸻ начиная созданием простых консольных программ, заканчивая огромными сайтами, обладает простым и понятным синтаксисом, приближенным к английскому языку, позволяет создавать легко читаемый код, не перегруженный знаками препинания, бесплатен, так как это свободно распространяемое программное обеспечение с открытым исходным кодом, интерпретируемый. Перед запуском программы компиляция не требуется, можно запускать в большинстве операционных систем, в том числе в Windows, Linux, macOS, язык программирования. С его помощью можно создавать графические приложения, простые игры, программы для анализа больших данных., интенсивно развивается и совершенствуется и наконец ⸻ он входит в ТОП-5 самых популярных языков программирования.

## **1.3. Основные библиотеки для написания интерфейса**

|  |  |
| --- | --- |
| *Библиотеки* | Описание |
| *PySimpleGUI* | библиотека для быстрого написания примитивного интерфейса с минимальными возможностями. |
| *Tkinter* | кросс-платформенная событийно-ориентированная графическая библиотека на основе средств Tk. |
| *PyQt* | набор расширений графического фреймворка Qt для языка программирования Python, выполненный в виде расширения Python. |
| *Flexx* | строит интерфейс на базе веб-технологий. |
| *PyGUI* | простой фреймворк для создания графических интерфейсов пользователя с помощью скриптов на языке Python. Он рисует через GPU и работает на Windows 10, macOS и Linux, содержит функции графики, темы, API для пользовательских отрисовок и инструменты для разработки приложений. |
| *kivi* | среда Python с открытым исходным кодом для разработки мобильных приложений и другого программного обеспечения с поддержкой мультитач с естественным пользовательским интерфейсом. |

## **1.4. Преимущества библиотеки PyQt5**

*PyQt5* включает набор классов управления компоновкой, которые используются для описания компоновки виджетов в пользовательском интерфейсе приложения. Эти layouts автоматически позиционируют и изменяют размер виджетов, когда объем доступного для них места изменяется, гарантируя, что они единообразно расположены и пользовательский интерфейс в целом остается пригодным для использования, простой графический интерфейс для компоновки объектов в окне, Система компоновки, предоставляет простой и мощный способ автоматической организации дочерних виджетов внутри виджета, чтобы гарантировать, что они эффективно используют доступное пространство, включает набор классов управления компоновкой, которые используются для описания компоновки виджетов в пользовательском интерфейсе приложения. Эти layouts автоматически позиционируют и изменяют размер виджетов, когда объем доступного для них места изменяется, гарантируя, что они единообразно расположены и пользовательский интерфейс в целом остается пригодным для использования, Все подклассы QWidget могут использовать менеджеры компоновки для управления своими дочерними элементами.

# **2. Технологический раздел**

## **2.1. Методы, библиотеки, классы**

Используемые библиотеки:

* *json*
* *math*
* *sys*
* *sqlite3*
* *csv*
* *hashlib*
* *PyQt5.QtWidgets.QApplication*
* *PyQt5.QtWidgets.QMainWindow*
* *PyQt5.QtWidgets*.*QMessageBox*
* *PyQt5.QtWidgets.QInputDialog*
* *PyQt5.QtCore*
* *PyQt5.QtGui*
* *PyQt5.QtWidgets*
* *PyQt5.QtGui.QPixmap*

*Используемые классы:*

* *Ui\_MainWindow*
* *DB*
* *Main\_Window*

*Использованные методы:*

* *setupUi*
* *retranslateUi*
* *add\_value*
* *create\_table*
* *read\_all\_from\_table*
* *get\_lines*
* *get\_line*
* *delete\_line*
* *change\_value*
* *m\_change*
* *m\_page\_change*
* *theme\_button*
* *view\_theory*
* *theme\_change*
* *test\_wigit\_controller*
* *accaunt\_button*
* *display\_questions*
* *check\_answer*
* *end\_test*
* *test\_buttons*
* *login*
* *registred*
* *export\_results*
* *img\_to\_label*

## **2.2. Структура**

Для создания интерфейса используется класс *Ui\_MainWindow(object)* содержащий в себе методы *setupUi* и *retranslateUi.*

В методе *setupUi* происходит инициализация расположения элементов в интерфейсе программы. *retranslateUi* применяет текст к объектам и к названию окна.

## **2.3. Руководство пользователя**

Для корректной работы приложения требуется:

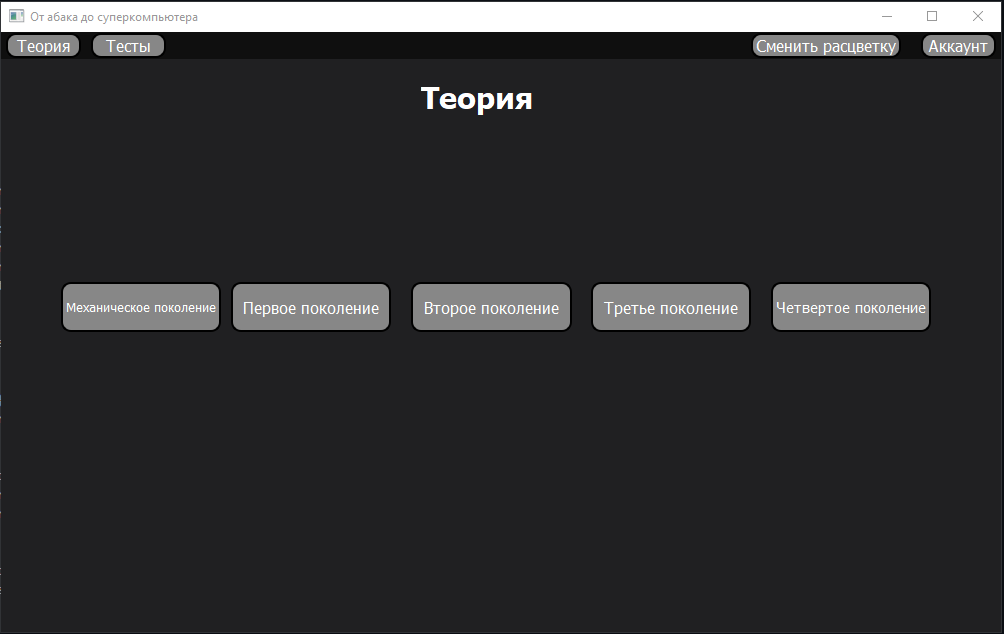
* экран минимальным размером по сторонам 1000X560 пикс.;
* операционная система «Windows 7/8/10»;
* место на диске 91 МБ;
* 64-разрядный процессор и операционная система;
* 1 гб. ОЗУ или больше;
* подключение интернета для скачивания**;**

Программа передается в виде папки, в которой содержится исполняемый файл и файлы библиотеки программы. Для запуска программы требуется запустить на исполнение файл *«From abacus to supercomputer.exe».* (рис. 1)



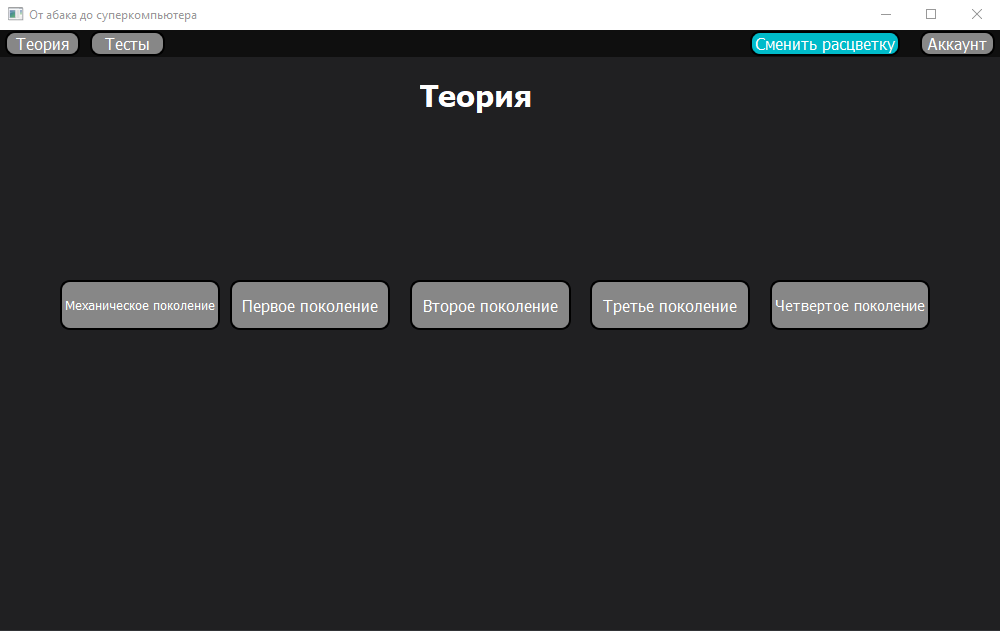
*Рисунок 1. Иконка приложения*

После запуска вас встретит главное меню, в котором вы сможете сразу приступить к изучению тем (рис. 2).



*Рисунок 2. Начальный экран, теория*

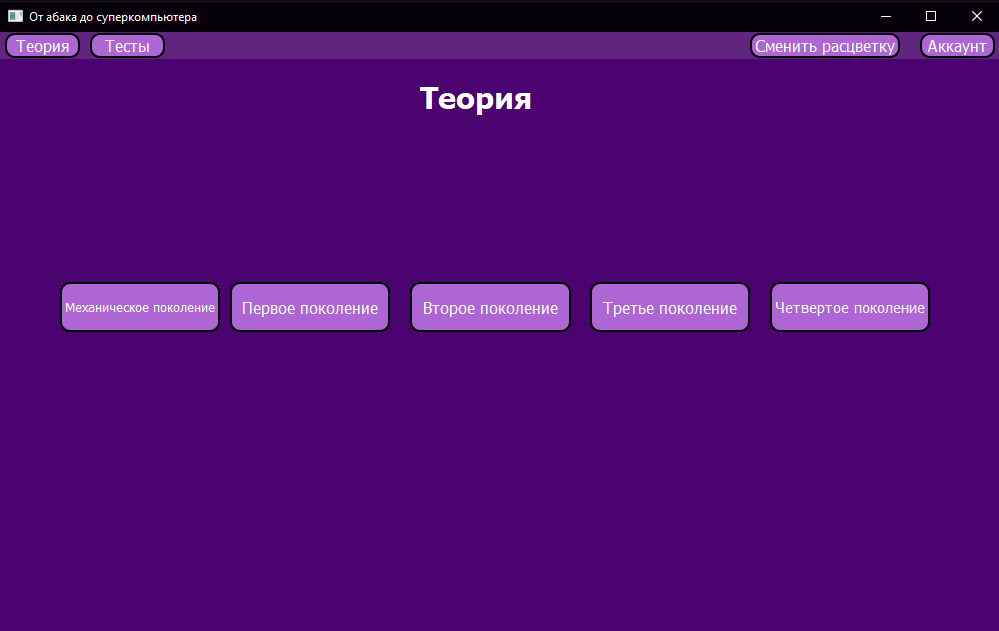
У программы есть несколько предустановленных тем, которые меняются по нажатию кнопки справа вверху (рис. 3-6)



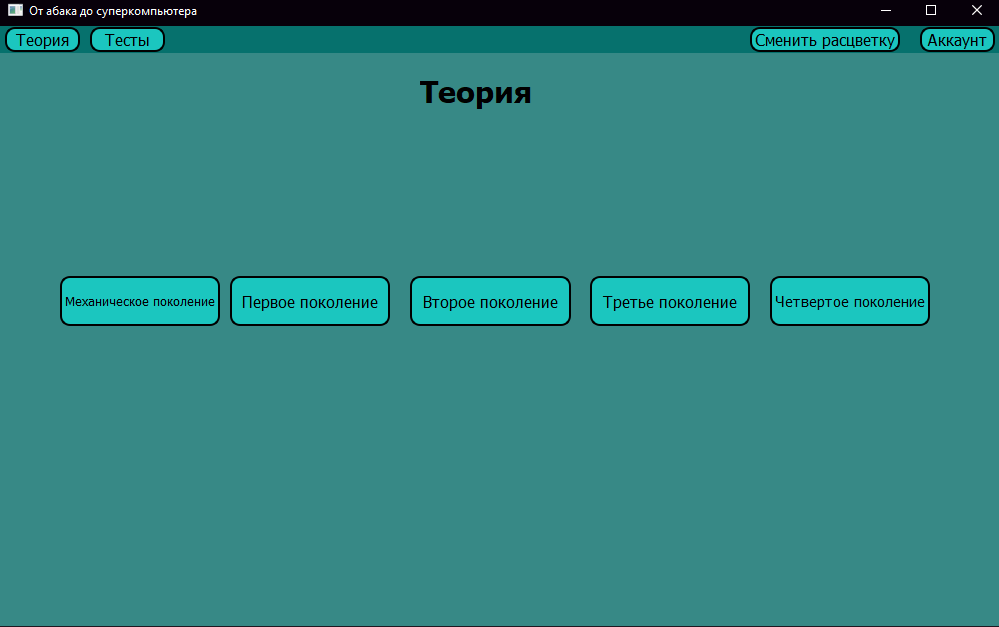
*Рисунок 3. Кнопка* «Сменить расцветку», *первая стандартная расцветка*



*Рисунок 4. Вторая стандартная расцветка*



*Рисунок 5. Третья стандартная расцветка*

**

*Рисунок 6. Четвертая стандартная расцветка*

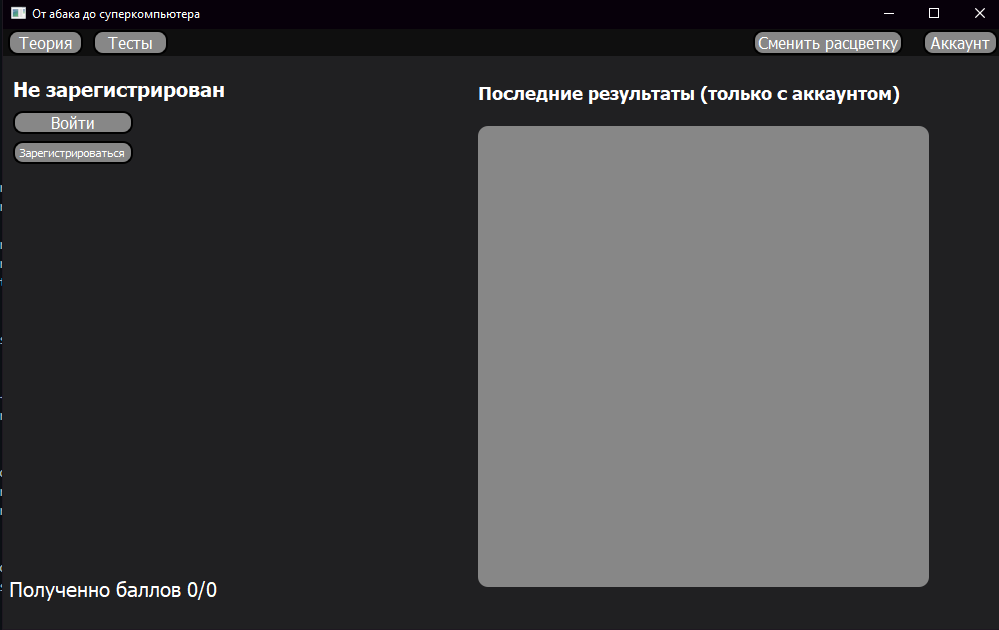
Далее я буду все демонстрировать на первой стандартной расцветке (рис. 3).

На начальном экране мы видим верхнюю часть программы, которая будет сопровождать на протяжении выполнения этой программы (рис. 7).



*Рисунок 7. Верхняя часть программы*

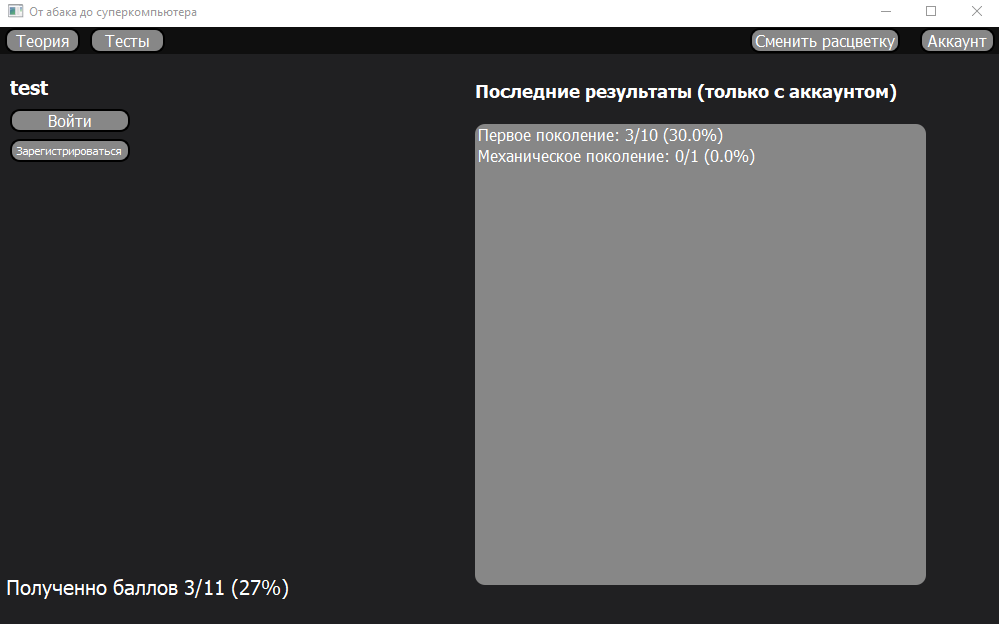
Две левые кнопки отвечают за выбор того, хотим ли мы изучать теорию, либо проходить тесты. Далее идут еще 2 кнопки. Про первую уже написано выше. Кнопка «*Аккаунт*» переносит нас на страницу с нашим аккаунтом (рис. 7)



*Рисунок 7. Вкладка* «*Аккаунт*»

Слева вверху мы видим сообщение о том, что мы не вошли/зарегистрировались в системе. Ниже мы можем совершить 2 действия ⸻ войти в аккаунт (если у вас есть учетные данные какого-то аккаунта), либо зарегистрироваться. При нажатии на одну из кнопок ⸻ будут возникать диалоговые окна, в которые нужно будет вводить то, что у вас попросят.

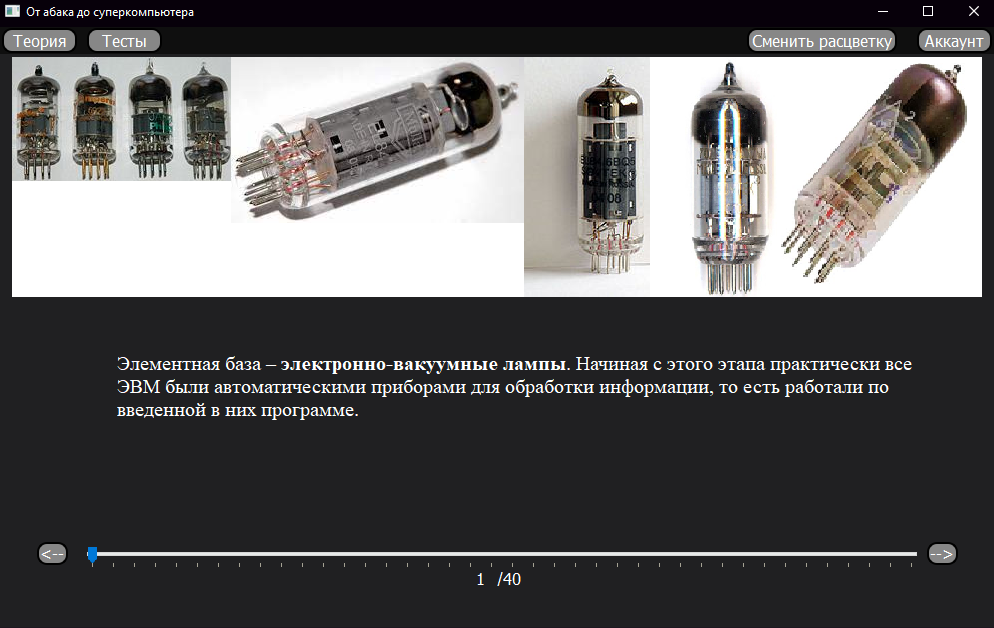
Вот так выглядит это окошко, если войти в аккаунт (рис. 2).



*Рисунок 8. Вкладка «Аккаунт» при вошедшем пользователе*

Под надписью *“Последние результаты”* мы видим список результатов этого аккаунта. Если вы решите несколько раз один и тот-же тест, то все ваши попытки будут отображены здесь. Так же, снизу слева, есть надпись, говорящая о том, сколько вы за все попытки решения тестов набрали баллов, и сколько могли набрать.

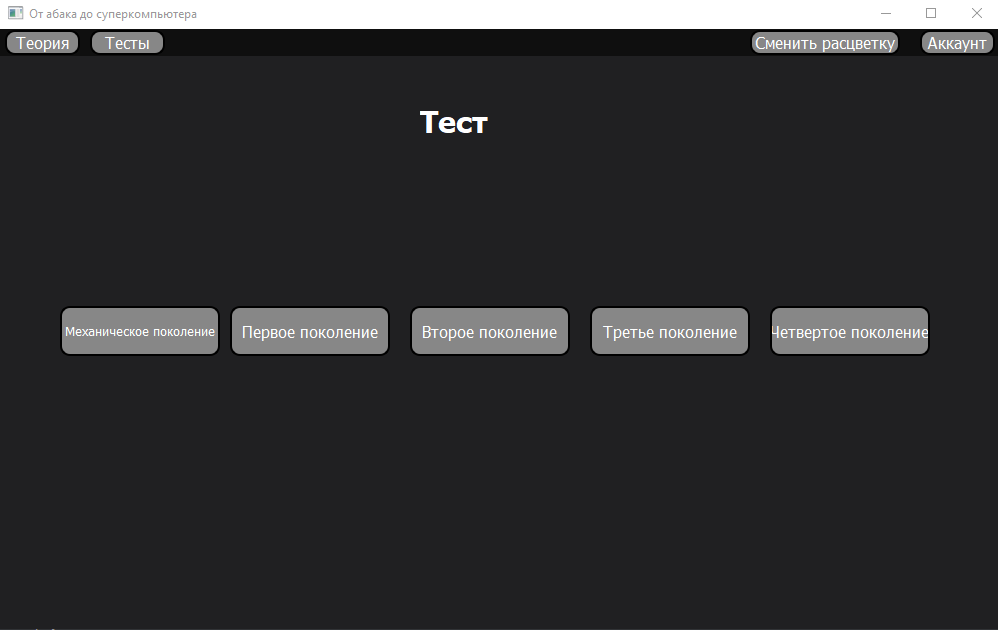
Теперь вернемся к вкладке “Теория” (рис. 2). Если перейти в любую из вкладок (в примере скриншот вкладки первого поколения), то мы увидим следующее *(рис. 9).*



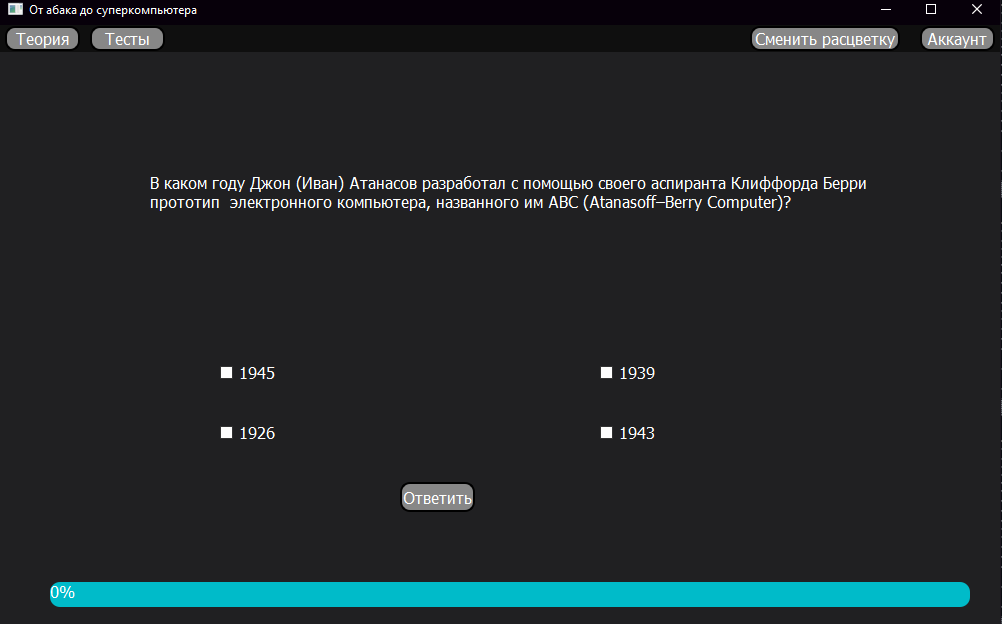
*Рисунок 9. Теория (выбранная)*

В верхней части мы наблюдаем картинку, которая меняется в зависимости от слайда. Ее так же может и не быть. Под картинкой находится текст. Он тоже меняется в зависимости от слайда. Ниже находятся элементы управления. Посередине находится слайдер, с помощью которого можно перемещаться по слайдам. Так же по его бокам есть кнопки, которые перелистывают слайд назад, или вперед (в зависимости от нажатой кнопки). Прямо под слайдером находится значение, которое показывает, какой сейчас открыт слайд и сколько их всего в этом слайде. Так же можно вводя числа переключать слайды. Ввод чего-то помимо цифр заблокирован.

Теперь перейдем к тестам *(рис. 10).* Чтобы перейти к тестам, сначала надо войти/зарегистрироваться. Можно так же, как и при выборе темы для изучения, выбрать и тему теста. В примере я буду показывать на примере теста на первое поколение.



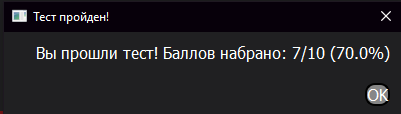
*Рисунок 10. Выбор теста*

­­­ 

*Рисунок 11. Вариант теста*

При выборе одной из тем, появится подобное окно *(рис. 11).* Сверху текст вопроса, ниже варианты ответа, кнопка сохранения ответов и шкала прогресса решения теста (если правильно ответить, то шкала будет прибавляться). В вариантах ответа, нужно выбрать один из вариантов. После того, как вы выберите ответ, то надо нажать на кнопку “Ответить”. Если ответить правильно, то ответ будет отмечен зеленым цветом, в противном случае ⸻ неправильные ответы будут выделены красным, а правильный ⸻ зеленым. Если вы не знаете ответ на вопрос, то можно просто нажать на кнопку “Ответить” и подсветится правильный ответ, но баллов за пропущенный вопрос не получить. Во время выполнения тестов, верхние кнопки становятся неактивными, кроме “Сменить расцветку”.

После завершения теста, будет выведено уведомление, подобное этому *(рис. 12).*, где будут показаны результаты пройденного теста.



*Рисунок 12. Уведомление об окончании теста*

После закрытия уведомления, будет открыта вкладка “Аккаунт” и в истории решения тестов будет отображен и только что пройденный тест.

# **Заключение**

В процессе проделанной работы было создано обучающее приложение «От абака до суперкомпьютера», была изучена литература по данной теме, литература и документация по языку программирования Python, изучена библиотека PyQT5, изучена программа QT Designer, изучены СУБД и sqlite3, разработан простой, но доступный интерфейс приложения, созданы шаблоны для возможности дальнейшего расширения программы, проведено тестирование и исправление ошибок. Программу можно расширять, добавлять в нее больше информации и новые темы.

# Список литературы

1. "Поколение Python": курс для начинающих [электронный ресурс]. Режим доступа URL: <https://stepik.org/course/58852/promo> – уровень доступа: открытый.
2. "Поколение Python": курс для продвинутых [электронный ресурс]. Режим оступа URL: <https://stepik.org/course/68343/promo> – уровень доступа: открытый.
3. Python 3.9.14 documentation [электронный ресурс]. Режим доступа URL: <https://docs.python.org/3.9/index.html> – уровень доступа: открытый.
4. QT Designer [электронный ресурс]. Режим доступа URL:  
   <https://doc.qt.io/qt-5/qtdesigner-manual.html> – уровень доступа: открытый.
5. QT Widgets [электронный ресурс]. Режим доступа URL:  
   <https://doc.qt.io/qt-5/qtwidgets-module.html> – уровень доступа: открытый.
6. Альбов А. Рей Томлинсон: QWERTYOP // Магия ПК. - 2001. - N 10.
7. Апокин И.А., Майстров Л.Е. Развитие вычислительных машин. - М.: Наука, 1974.
8. Балашов Е.П., Частиков А.П. Эволюция вычислительных систем. - М.: Знание, 1981
9. Балашов Е.П., Частиков А.П. Эволюция мини- и микроЭВМ. Малые вычислительные машины. - М.: Знание, 1983.
10. Грофф, Джеймс Р., Вайнберг, Пол Н., Оппелъ, Эндрю Дж. Г89 SQL: полное руководство, 3‑е изд.: Пер. с англ. ‑М.: ООО "И.Д. Вильяме", 2015. -960 с.: ил. – Парал. тит. англ.
11. Мюллер, Джон Пол. Python для чайников, 2-е изд.: Пер. с англ. – СПб.: ООО "Диалектика", 2019. -416 с.: ил. – Парал. тит. англ.
12. Персональный компьютер. С. В. Глушаков, А. С. Сурядный, Т. С. Хачиров. – 6-е изд., доп. и перераб. - Москва: АСТ, 2008. - 475 с.: ил.
13. Петров, Владимир Николаевич. Информационные системы. – Санкт‑Петербург [и др.]: Питер, 2002. - 687 с.: ил.; 24 см. – 2003.
14. Поляков, Еремин, Информатика. 7 класс. Учебник. В 2-х частях. ФП. – 2016.
15. Сексенбаев, Курманбек. Информационные технологии в развитии современного информационного общества. – 2015.

# Приложение

*From abacus to supercomputer.py*

import json

import math

import sys

import sqlite3

import threading

import time

from PyQt5.QtGui import QFontDatabase, QFont

import hashlib

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

from PyQt5 import QtCore, QtGui, QtWidgets

from PyQt5.QtGui import QPixmap

ch\_flag = False

app = None

class Ui\_MainWindow(object):

def setupUi(self, MainWindow):

self.MainWindow = MainWindow

MainWindow.setObjectName("MainWindow")

MainWindow.resize(1000, 600)

#theory\_wigit

self.main\_wiget\_theory = QtWidgets.QFrame(MainWindow)

self.main\_wiget\_theory.setGeometry(QtCore.QRect(0, 20, 1000, 580))

self.main\_wiget\_theory.setFrameShape(QtWidgets.QFrame.StyledPanel)

self.main\_wiget\_theory.setFrameShadow(QtWidgets.QFrame.Raised)

self.main\_wiget\_theory.setObjectName("main\_wiget\_theory")

self.m\_text\_theory = QtWidgets.QLabel(self.main\_wiget\_theory)

self.m\_text\_theory.setGeometry(QtCore.QRect(120, 270, 820, 140))

self.m\_text\_theory.setObjectName("m\_text\_theory")

self.m\_img = QtWidgets.QLabel(self.main\_wiget\_theory)

self.m\_img.setGeometry(QtCore.QRect(15, 9, 970, 240))

self.m\_img.setObjectName("m\_img")

self.m\_split\_slider = QtWidgets.QSlider(self.main\_wiget\_theory)

self.m\_split\_slider.setGeometry(QtCore.QRect(90, 500, 830, 20))

self.m\_split\_slider.setOrientation(QtCore.Qt.Horizontal)

self.m\_split\_slider.setObjectName("m\_split\_slider")

#m

self.m\_page = QtWidgets.QLineEdit(self.main\_wiget\_theory)

self.m\_page.setGeometry(QtCore.QRect(477, 523, 25, 15))

self.m\_page.setObjectName("m\_page")

self.m\_l\_change = QtWidgets.QPushButton(self.main\_wiget\_theory)

self.m\_l\_change.setGeometry(QtCore.QRect(40, 495, 31, 23))

self.m\_l\_change.setObjectName("m\_right\_change")

self.m\_r\_change = QtWidgets.QPushButton(self.main\_wiget\_theory)

self.m\_r\_change.setGeometry(QtCore.QRect(930, 495, 31, 23))

self.m\_r\_change.setObjectName("m\_left\_change")

self.m\_all\_page = QtWidgets.QLabel(self.main\_wiget\_theory)

self.m\_all\_page.setGeometry(QtCore.QRect(500, 523, 47, 16))

self.m\_all\_page.setObjectName("m\_all\_page")

#theme\_buttons

self.theme\_buttons = QtWidgets.QFrame(MainWindow)

self.theme\_buttons.setGeometry(QtCore.QRect(0, 27, 1000, 600))

self.theme\_buttons.setFrameShape(QtWidgets.QFrame.StyledPanel)

self.theme\_buttons.setFrameShadow(QtWidgets.QFrame.Raised)

self.theme\_buttons.setObjectName("theme\_buttons")

self.th\_zero = QtWidgets.QPushButton(self.theme\_buttons)

self.th\_zero.setGeometry(QtCore.QRect(60, 250, 160 , 50))

self.th\_zero.setObjectName("th\_zero")

self.th\_first = QtWidgets.QPushButton(self.theme\_buttons)

self.th\_first.setGeometry(QtCore.QRect(230, 250, 160 , 50))

self.th\_first.setObjectName("th\_first")

self.th\_second = QtWidgets.QPushButton(self.theme\_buttons)

self.th\_second.setGeometry(QtCore.QRect(410, 250, 161, 50))

self.th\_second.setObjectName("th\_second")

self.th\_third = QtWidgets.QPushButton(self.theme\_buttons)

self.th\_third.setGeometry(QtCore.QRect(590, 250, 160, 50))

self.th\_third.setObjectName("th\_third")

self.th\_forth = QtWidgets.QPushButton(self.theme\_buttons)

self.th\_forth.setGeometry(QtCore.QRect(770, 250, 160, 50))

self.th\_forth.setObjectName("th\_forth")

self.th\_name = QtWidgets.QLabel(self.theme\_buttons)

self.th\_name.setGeometry(QtCore.QRect(420, 40, 150, 50))

self.th\_name.setObjectName("th\_name")

MainWindow.setCentralWidget(self.theme\_buttons)

#blind

self.blind = QtWidgets.QWidget(MainWindow)

self.blind.setGeometry(QtCore.QRect(0, 0, 1000, 27))

self.blind.setObjectName("blind")

self.b\_theory\_button = QtWidgets.QPushButton(self.blind)

self.b\_theory\_button.setGeometry(QtCore.QRect(5, 1, 75, 25))

self.b\_theory\_button.setObjectName("b\_theory\_button")

self.b\_accaunt\_button = QtWidgets.QPushButton(self.blind)

self.b\_accaunt\_button.setGeometry(QtCore.QRect(920, 1, 75, 25))

self.b\_accaunt\_button.setObjectName("b\_accaunt\_button")

self.b\_test\_button = QtWidgets.QPushButton(self.blind)

self.b\_test\_button.setGeometry(QtCore.QRect(90, 1, 75, 25))

self.b\_test\_button.setObjectName("b\_test\_button")

self.b\_change\_color = QtWidgets.QPushButton(self.blind)

self.b\_change\_color.setGeometry(QtCore.QRect(750, 1, 150, 25))

self.b\_change\_color.setObjectName("b\_change\_color")

#accaunt\_wigit

self.accaunt\_wigit = QtWidgets.QFrame(MainWindow)

self.accaunt\_wigit.setGeometry(QtCore.QRect(0, 27, 1000, 600))

self.accaunt\_wigit.setFrameShape(QtWidgets.QFrame.StyledPanel)

self.accaunt\_wigit.setFrameShadow(QtWidgets.QFrame.Raised)

self.accaunt\_wigit.setObjectName("accaunt\_wigit")

self.ac\_accaunt\_name = QtWidgets.QLabel(self.accaunt\_wigit)

self.ac\_accaunt\_name.setGeometry(QtCore.QRect(10, 17, 991, 31))

self.ac\_accaunt\_name.setObjectName("ac\_accaunt\_name")

self.ac\_history\_label = QtWidgets.QLabel(self.accaunt\_wigit)

self.ac\_history\_label.setGeometry(QtCore.QRect(475, 17, 440, 40))

self.ac\_history\_label.setObjectName("ac\_history\_label")

self.ac\_login = QtWidgets.QPushButton(self.accaunt\_wigit)

self.ac\_login.setGeometry(QtCore.QRect(10, 55, 120, 23))

self.ac\_login.setObjectName("ac\_login")

self.ac\_registr = QtWidgets.QPushButton(self.accaunt\_wigit)

self.ac\_registr.setGeometry(QtCore.QRect(10, 85, 120, 23))

self.ac\_registr.setObjectName("ac\_registr")

self.ac\_view\_balls = QtWidgets.QLabel(self.accaunt\_wigit)

self.ac\_view\_balls.setGeometry(QtCore.QRect(6, 502, 601, 61))

self.ac\_view\_balls.setObjectName("ac\_view\_balls")

self.ac\_histiry\_results = QtWidgets.QListWidget(self.accaunt\_wigit)

self.ac\_histiry\_results.setGeometry(QtCore.QRect(475, 70, 451, 461))

self.ac\_histiry\_results.setObjectName("ac\_histiry\_results")

#test\_wigit\_theme

self.test\_theme\_get = QtWidgets.QFrame(MainWindow)

self.test\_theme\_get.setGeometry(QtCore.QRect(0, 27, 1000, 600))

self.test\_theme\_get.setFrameShape(QtWidgets.QFrame.StyledPanel)

self.test\_theme\_get.setFrameShadow(QtWidgets.QFrame.Raised)

self.test\_theme\_get.setObjectName("test\_theme\_get")

self.te\_zero = QtWidgets.QPushButton(self.test\_theme\_get)

self.te\_zero.setGeometry(QtCore.QRect(60, 250, 160 , 50))

self.te\_zero.setObjectName("te\_zero")

self.te\_first = QtWidgets.QPushButton(self.test\_theme\_get)

self.te\_first.setGeometry(QtCore.QRect(230, 250, 160, 50))

self.te\_first.setObjectName("te\_first")

self.te\_second = QtWidgets.QPushButton(self.test\_theme\_get)

self.te\_second.setGeometry(QtCore.QRect(410, 250, 160, 50))

self.te\_second.setObjectName("te\_second")

self.te\_third = QtWidgets.QPushButton(self.test\_theme\_get)

self.te\_third.setGeometry(QtCore.QRect(590, 250, 160, 50))

self.te\_third.setObjectName("te\_third")

self.te\_forth = QtWidgets.QPushButton(self.test\_theme\_get)

self.te\_forth.setGeometry(QtCore.QRect(770, 250, 160, 50))

self.te\_forth.setObjectName("te\_forth")

self.te\_name = QtWidgets.QLabel(self.test\_theme\_get)

self.te\_name.setGeometry(QtCore.QRect(420, 40, 150, 50))

self.te\_name.setObjectName('te\_name')

#test\_wigit\_answers

self.test\_wigit\_answers = QtWidgets.QFrame(MainWindow)

self.test\_wigit\_answers.setGeometry(QtCore.QRect(0, 27, 1000, 600))

self.test\_wigit\_answers.setFrameShape(QtWidgets.QFrame.StyledPanel)

self.test\_wigit\_answers.setFrameShadow(QtWidgets.QFrame.Raised)

self.test\_wigit\_answers.setObjectName("test\_wigit\_answers")

self.ta\_question = QtWidgets.QLabel(self.test\_wigit\_answers)

self.ta\_question.setGeometry(QtCore.QRect(150, 30, 730, 220))

self.ta\_question.setObjectName("ta\_answer")

self.ta\_progress = QtWidgets.QProgressBar(self.test\_wigit\_answers)

self.ta\_progress.setGeometry(QtCore.QRect(50, 530, 920, 25))

self.ta\_progress.setProperty("value", 0)

self.ta\_progress.setObjectName("ta\_progress")

self.ta\_fst\_answ = QtWidgets.QCheckBox(self.test\_wigit\_answers)

self.ta\_fst\_answ.setGeometry(QtCore.QRect(220, 310, 380, 20))

self.ta\_fst\_answ.setObjectName("ta\_fst\_answ")

self.ta\_sec\_answ = QtWidgets.QCheckBox(self.test\_wigit\_answers)

self.ta\_sec\_answ.setGeometry(QtCore.QRect(220, 370, 380, 20))

self.ta\_sec\_answ.setObjectName("ta\_sec\_answ")

self.ta\_third\_answ = QtWidgets.QCheckBox(self.test\_wigit\_answers)

self.ta\_third\_answ.setGeometry(QtCore.QRect(600, 310, 400, 20))

self.ta\_third\_answ.setObjectName("ta\_third\_answ")

self.ta\_forth\_answ = QtWidgets.QCheckBox(self.test\_wigit\_answers)

self.ta\_forth\_answ.setGeometry(QtCore.QRect(600, 370, 400, 20))

self.ta\_forth\_answ.setObjectName("ta\_forth\_answ")

self.ta\_answ = QtWidgets.QPushButton(self.test\_wigit\_answers)

self.ta\_answ.setGeometry(QtCore.QRect(400, 430, 75, 30))

self.ta\_answ.setObjectName("ta\_answ")

#-----------------------------------------------------------------------------------

self.retranslateUi(MainWindow)

QtCore.QMetaObject.connectSlotsByName(MainWindow)

def retranslateUi(self, MainWindow):

\_translate = QtCore.QCoreApplication.translate

MainWindow.setWindowTitle(\_translate("MainWindow", "От абака до суперкомпьютера"))

#theme\_buttons

self.th\_zero.setText(\_translate("MainWindow", "Механическое поколение"))

self.th\_first.setText(\_translate("MainWindow", "Первое поколение"))

self.th\_second.setText(\_translate("MainWindow", "Второе поколение"))

self.th\_third.setText(\_translate("MainWindow", "Третье поколение"))

self.th\_forth.setText(\_translate("MainWindow", "Четвертое поколение"))

self.th\_name.setText(\_translate("MainWindow", "<b>Теория</b>"))

self.m\_all\_page.setText(\_translate("MainWindow", "TextLabel"))

self.m\_l\_change.setText(\_translate("MainWindow", "<--"))

self.m\_r\_change.setText(\_translate("MainWindow", "-->"))

#blind

self.b\_theory\_button.setText(\_translate("MainWindow", "Теория"))

self.b\_accaunt\_button.setText(\_translate("MainWindow", "Аккаунт"))

self.b\_test\_button.setText(\_translate("MainWindow", "Тесты"))

self.b\_change\_color.setText(\_translate("MainWindow", "Сменить расцветку"))

#theory\_wigit

self.m\_text\_theory.setText(\_translate("MainWindow", "Теория"))

#accaunt\_wigit

self.ac\_accaunt\_name.setText(\_translate("MainWindow", "Не зарегистрирован"))

self.ac\_login.setText(\_translate("MainWindow", "Войти"))

self.ac\_registr.setText(\_translate("MainWindow", "Зарегистрироваться"))

self.ac\_view\_balls.setText(\_translate("MainWindow", "Полученно баллов 10/1000"))

self.ac\_history\_label.setText(\_translate("MainWindow", "Последние результаты (только с аккаунтом)"))

#test\_wigit\_theme

self.te\_zero.setText(\_translate("MainWindow", "Механическое поколение"))

self.te\_first.setText(\_translate("MainWindow", "Первое поколение"))

self.te\_second.setText(\_translate("MainWindow", "Второе поколение"))

self.te\_third.setText(\_translate("MainWindow", "Третье поколение"))

self.te\_forth.setText(\_translate("MainWindow", "Четвертое поколение"))

self.te\_name.setText(\_translate("MainWindow", "<b>Тест</b>"))

#test\_wigit\_answers

self.ta\_question.setText(\_translate("MainWindow", "TextLabel"))

self.ta\_fst\_answ.setText(\_translate("MainWindow", "CheckBox"))

self.ta\_sec\_answ.setText(\_translate("MainWindow", "CheckBox"))

self.ta\_third\_answ.setText(\_translate("MainWindow", "CheckBox"))

self.ta\_forth\_answ.setText(\_translate("MainWindow", "CheckBox"))

self.ta\_answ.setText(\_translate("MainWindow", "Ответить"))

class DB:

def \_\_init\_\_(self, base\_name: str) -> None:

self.base = sqlite3.connect(base\_name, check\_same\_thread=False)

self.cur = self.base.cursor()

if not self.base:

raise ConnectionError('Error to connection DB')

def add\_value(self, tabl\_n: str, \*args) -> None:

self.cur.execute('INSERT INTO {} VALUES({})'.format(tabl\_n, ', '.join(['?' for \_ in range(len(args))])), tuple(args))

self.base.commit()

def create\_table(self, name: str, \*arg) -> None:

self.base.execute('CREATE TABLE IF NOT EXISTS {}({})'.format(name, ', '.join(arg)))

self.base.commit()

def read\_all\_from\_table(self, tabl\_n: str) -> list:

return self.base.execute(f"SELECT \* FROM {tabl\_n}").fetchall()

def get\_lines(self, tabl\_n: str, colum\_name: str, value: str):

return self.base.execute("SELECT \* FROM {} WHERE {} == ?".format(tabl\_n, colum\_name), (value,)).fetchall()

def get\_line(self, tabl\_n: str, colum\_name: str, value: str):

return self.base.execute("SELECT \* FROM {} WHERE {} == ?".format(tabl\_n, colum\_name), (value,)).fetchone()

def delete\_line(self, tabl\_n: str, fl\_colum\_name: str, fl\_value: str) -> None:

self.cur.execute('DELETE FROM {} WHERE {} == ?'.format(tabl\_n, fl\_colum\_name), (fl\_value,))

self.base.commit()

def change\_value(self, tabl\_n: str, fl\_value: str, colum\_fl\_name: str, new\_value: str, colum\_new\_name: str) -> None:

self.cur.execute('UPDATE {} SET {} == ? WHERE {} == ?'.format(tabl\_n, colum\_new\_name, colum\_fl\_name), (new\_value, fl\_value))

self.base.commit()

class Main\_Window(QMainWindow, Ui\_MainWindow):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.setupUi(self)

self.accaunt = None

self.db = DB('db.db')

self.db.create\_table('accaunts', 'id INTENGER PRIMARY KEY', 'name', 'pass', 'balls', 'all\_balls')

self.db.create\_table('results', 'id INTENGER PRIMARY KEY', 'name', 'type', 'balls', 'max\_balls')

#perem

self.c\_page\_chage\_code = False

self.theme\_button()

#Коннект кнопок

#m

self.m\_r\_change.clicked.connect(self.m\_change)

self.m\_l\_change.clicked.connect(self.m\_change)

self.m\_page.setText('1')

self.m\_page.textChanged.connect(self.m\_page\_change)

#theme\_buttons

self.th\_zero.clicked.connect(self.theme\_change)

self.th\_first.clicked.connect(self.theme\_change)

self.th\_second.clicked.connect(self.theme\_change)

self.th\_third.clicked.connect(self.theme\_change)

self.th\_forth.clicked.connect(self.theme\_change)

#blind

self.b\_theory\_button.clicked.connect(self.theme\_button)

self.b\_test\_button.clicked.connect(self.test\_wigit\_controller)

self.b\_accaunt\_button.clicked.connect(self.accaunt\_button)

self.b\_change\_color.clicked.connect(self.change\_color)

#test\_select\_theme

self.te\_zero.clicked.connect(self.test\_buttons)

self.te\_first.clicked.connect(self.test\_buttons)

self.te\_second.clicked.connect(self.test\_buttons)

self.te\_third.clicked.connect(self.test\_buttons)

self.te\_forth.clicked.connect(self.test\_buttons)

#accaunts

self.ac\_login.clicked.connect(self.login)

self.ac\_registr.clicked.connect(self.registred)

# self.ac\_export\_csv.clicked.connect(self.export\_results)

#Fonts init

fontId = QFontDatabase.addApplicationFont("fonts\RedOctober-Fat.ttf")

if fontId == 0:

fontName = QFontDatabase.applicationFontFamilies(fontId)[0]

self.font = QFont(fontName, 30)

else:

self.font = QFont()

def m\_change(self):

self.c\_page\_chage\_code = True

if self.sender() == self.m\_r\_change:

self.m\_split\_slider.setValue(self.m\_split\_slider.value() + 1)

self.m\_page.setText(str(self.m\_split\_slider.value() + 1))

elif self.sender() == self.m\_l\_change:

self.m\_split\_slider.setValue(self.m\_split\_slider.value() - 1)

self.m\_page.setText(str(self.m\_split\_slider.value() + 1))

def m\_page\_change(self):

if self.c\_page\_chage\_code:

self.c\_page\_chage\_code = False

return

if self.m\_page.text() == '':

return

if len([i for i in self.m\_page.text() if i not in '0123456789']) != 0:

self.m\_page.setText(''.join([i for i in self.m\_page.text() if i in '0123456789']))

return

elif int(self.m\_page.text()) > self.m\_split\_slider.maximum() + 1:

self.m\_page.setText(str(self.m\_split\_slider.maximum() + 1))

elif int(self.m\_page.text()) <= 0:

self.m\_page.setText('1')

self.m\_split\_slider.setValue(int(self.m\_page.text()) - 1)

def theme\_button(self):

self.main\_wiget\_theory.hide()

self.accaunt\_wigit.hide()

self.test\_wigit\_answers.hide()

self.test\_theme\_get.hide()

self.theme\_buttons.show()

self.blind.show()

def view\_theory(self):

# self.\_dashes\_under\_slider()

self.m\_img.setGeometry(QtCore.QRect(15, 10, 970, 240))

self.m\_img.setPixmap(QPixmap())

self.m\_text\_theory.setGeometry(QtCore.QRect(120, 270, 820, 140))

a = str(self.m\_split\_slider.value() + 1)

json\_ = None

with open(f"json\\theme{self.theme}.json", "r", encoding='utf-8') as file:

json\_ = json.loads(file.read())

try:

pixmap = QPixmap(json\_[a]['img'])

if pixmap.width() > self.m\_img.width() and pixmap.height() <= self.m\_img.height(): # Подгон размеров картинки под label

otn = pixmap.width() / pixmap.height()

pixmap = pixmap.scaled(self.m\_img.width(), math.ceil(self.m\_img.width() / otn))

elif pixmap.width() <= self.m\_img.width() and pixmap.height() > self.m\_img.height():

otn = pixmap.width() / pixmap.height()

pixmap = pixmap.scaled(int(self.m\_img.height() \* otn), self.m\_img.height())

elif pixmap.width() > self.m\_img.width() and pixmap.height() > self.m\_img.height():

otn = pixmap.width() / pixmap.height()

pixmap = pixmap.scaled(self.m\_img.width(), math.ceil(self.m\_img.width() / otn))

if pixmap.width() <= self.m\_img.width() and pixmap.height() > self.m\_img.height():

otn = pixmap.width() / pixmap.height()

pixmap = pixmap.scaled( math.ceil(self.m\_img.height() \* otn), self.m\_img.height())

if pixmap.width() < self.m\_img.width() and pixmap.height() < self.m\_img.height():

otn = pixmap.width() / pixmap.height()

pixmap = pixmap.scaled(int(self.m\_img.height() \* otn), self.m\_img.height())

if pixmap.width() > self.m\_img.width() and pixmap.height() <= self.m\_img.height():

otn = pixmap.width() / pixmap.height()

pixmap = pixmap.scaled(self.m\_img.width(), math.ceil(self.m\_img.width() / otn))

# Задавание пиксмапа

self.m\_img.setPixmap(pixmap)

x = int((970 / 2) - (pixmap.width() / 2)) # Чтобы картинка была посередине

y = int((240 / 2) - (pixmap.height() / 2))

self.m\_img.setGeometry(QtCore.QRect(15 + x, 10 + y, pixmap.width(), pixmap.height()))

except (TypeError, ZeroDivisionError):

self.m\_text\_theory.setGeometry(QtCore.QRect(30, 20, 930, 410))

self.m\_img.setGeometry(QtCore.QRect(0, 0, 0, 0))

self.m\_split\_slider.setTickPosition(QtWidgets.QSlider.TickPosition.TicksBelow)

self.m\_split\_slider.setTickInterval(1)

# self.c\_page\_chage\_code = True

self.m\_page.setText(str(self.m\_split\_slider.value() + 1))

self.m\_all\_page.setText('/' + str(self.m\_split\_slider.maximum() + 1))

self.m\_text\_theory.setText(str(json\_[a]['text']))

self.m\_text\_theory.setFont(QtGui.QFont('Times New Roman', 15))

self.m\_text\_theory.setWordWrap(True) # Чтобы текст был с переносами

def theme\_change(self):

self.theme\_buttons.hide()

self.accaunt\_wigit.hide()

self.test\_wigit\_answers.hide()

self.test\_theme\_get.hide()

self.main\_wiget\_theory.show()

self.blind.show()

if self.sender() == self.th\_zero:

self.theme = '0'

elif self.sender() == self.th\_first:

self.theme = '1'

elif self.sender() == self.th\_second:

self.theme = '2'

elif self.sender() == self.th\_third:

self.theme = '3'

elif self.sender() == self.th\_forth:

self.theme = '4'

json\_ = None

with open(f"json\\theme{self.theme}.json", "r") as file:

json\_ = json.loads(file.read())

self.m\_split\_slider.setValue(0)

self.m\_split\_slider.setMaximum(len(json\_) - 1)

self.m\_split\_slider.setSingleStep(1)

self.m\_split\_slider.valueChanged.connect(self.view\_theory)

self.view\_theory()

def test\_wigit\_controller(self):

if not self.accaunt:

dlg = QMessageBox(self)

dlg.setWindowTitle("Упс...")

dlg.setText(f"Вам нужно войти в свой аккаунт, чтобы решать тесты!")

dlg.exec()

return

self.main\_wiget\_theory.hide()

self.accaunt\_wigit.hide()

self.test\_wigit\_answers.hide()

self.test\_theme\_get.show()

self.blind.show()

def accaunt\_button(self):

self.theme\_buttons.hide()

self.main\_wiget\_theory.hide()

self.test\_theme\_get.hide()

self.accaunt\_wigit.show()

self.blind.show()

self.ac\_view\_balls.clear()

self.ac\_histiry\_results.clear()

if self.accaunt:

self.ac\_accaunt\_name.setText(self.accaunt[1])

results = self.db.get\_lines('results', 'name', self.accaunt[1])

all\_balls, your\_balls = 0, 0

for i in results:

self.ac\_histiry\_results.addItem(f"{i[2]}: {i[3]}/{i[4]} ({i[3] / i[4] \* 100}%)")

all\_balls += i[4]

your\_balls += i[3]

self.ac\_view\_balls.setText(f"Полученно баллов {your\_balls}/{all\_balls} ({int(your\_balls / all\_balls \* 100) if all\_balls != 0 else 0}%)")

self.db.change\_value('accaunts', self.accaunt[1], 'name', your\_balls, 'balls')

self.db.change\_value('accaunts', self.accaunt[1], 'name', all\_balls, 'all\_balls')

# self.ac\_export\_csv.setEnabled(True)

else:

# self.ac\_export\_csv.setEnabled(False)

self.ac\_accaunt\_name.setText("Не зарегистрирован")

self.ac\_view\_balls.setText(f"Полученно баллов 0/0")

def display\_questions(self):

self.ta\_answ.setText('Ответить')

questions = None

with open(f"json\\test{self.test\_theme}.json", 'r', encoding='utf-8') as file:

questions = json.loads(file.read())

self.max\_balls = len(questions)

if int(self.question\_now) > len(questions):

self.end\_test()

return

self.ta\_question.setWordWrap(True)

self.ta\_question.setText(questions[self.question\_now]['text'])

self.ta\_fst\_answ.setText(questions[self.question\_now]['var'].split('\r')[0])

self.ta\_sec\_answ.setText(questions[self.question\_now]['var'].split('\r')[1])

self.ta\_third\_answ.setText(questions[self.question\_now]['var'].split('\r')[2])

self.ta\_forth\_answ.setText(questions[self.question\_now]['var'].split('\r')[3])

self.ta\_fst\_answ.setStyleSheet('')

self.ta\_sec\_answ.setStyleSheet('')

self.ta\_third\_answ.setStyleSheet('')

self.ta\_forth\_answ.setStyleSheet('')

self.ta\_fst\_answ.setChecked(False)

self.ta\_sec\_answ.setChecked(False)

self.ta\_third\_answ.setChecked(False)

self.ta\_forth\_answ.setChecked(False)

self.theme\_buttons.hide()

self.main\_wiget\_theory.hide()

self.accaunt\_wigit.hide()

self.test\_theme\_get.hide()

self.test\_wigit\_answers.show()

self.blind.show()

try:

self.ta\_answ.clicked.disconnect()

except TypeError:

pass

self.ta\_answ.clicked.connect(self.check\_answer)

def check\_answer(self):

question = None

with open(f"json\\test{self.test\_theme}.json", 'r', encoding='utf-8') as file:

question = json.loads(file.read())[self.question\_now]

check\_boxes = (self.ta\_fst\_answ, self.ta\_sec\_answ,

self.ta\_third\_answ, self.ta\_forth\_answ)

answers = (self.ta\_fst\_answ.isChecked(), self.ta\_sec\_answ.isChecked(),

self.ta\_third\_answ.isChecked(), self.ta\_forth\_answ.isChecked())

corr\_num = None

for i in range(len(check\_boxes)):

if check\_boxes[i].text() == question["corr"]:

corr\_num = i

break

if answers[corr\_num] == True and answers.count(True) == 1:

self.balls += 1

check\_boxes[corr\_num].setStyleSheet("background-color: rgba(0, 255, 0, 100);")

else:

for i, j in [(answers[x], check\_boxes[x]) for x in range(4)]:

if i == True and j != check\_boxes[corr\_num]:

j.setStyleSheet("background-color: rgba(255, 0, 0, 100);")

check\_boxes[corr\_num].setStyleSheet("background-color: rgba(0, 255, 0, 100);")

self.ta\_answ.setText('Продолжить')

self.ta\_answ.clicked.disconnect()

self.ta\_answ.clicked.connect(self.display\_questions)

self.ta\_progress.setProperty("value", int(self.balls) \* 100 / self.max\_balls)

self.question\_now = str(int(self.question\_now) + 1)

def end\_test(self):

self.b\_accaunt\_button.setEnabled(True)

self.b\_theory\_button.setEnabled(True)

self.b\_test\_button.setEnabled(True)

dlg = QMessageBox(self)

dlg.setWindowTitle("Тест пройден!")

dlg.setText(f"Вы прошли тест! Баллов набрано: {self.balls}/{self.max\_balls} ({self.balls / self.max\_balls \* 100}%)")

dlg.exec()

thems = {

'0': 'Механическое поколение',

'1': 'Первое поколение',

'2': 'Второе поколение',

'3': 'Третье поколение',

'4': 'Четвертое поколение'

}

self.db.add\_value('results', None, self.accaunt[1], thems[self.test\_theme], self.balls, self.max\_balls)

self.test\_wigit\_answers.hide()

self.main\_wiget\_theory.hide()

self.test\_theme\_get.hide()

self.theme\_buttons.hide()

self.accaunt\_wigit.show()

self.blind.show()

self.accaunt\_button()

def test\_buttons(self):

self.balls = 0

self.ta\_progress.setProperty("value", 0)

self.test\_theme = ''

if self.sender() == self.te\_zero:

self.test\_theme = '0'

elif self.sender() == self.te\_first:

self.test\_theme = '1'

elif self.sender() == self.te\_second:

self.test\_theme = '2'

elif self.sender() == self.te\_third:

self.test\_theme = '3'

elif self.sender() == self.te\_forth:

self.test\_theme = '4'

self.question\_now = '1'

self.ta\_question

self.display\_questions()

self.b\_accaunt\_button.setEnabled(False)

self.b\_test\_button.setEnabled(False)

self.b\_theory\_button.setEnabled(False)

def login(self):

a = QInputDialog()

b = a.getText(self.accaunt\_wigit, 'Login', 'Введите имя вашего аккаунта:')

if b[1] is False:

return

accaunt = self.db.get\_line('accaunts', 'name', b[0])

if accaunt:

a = QInputDialog()

b = a.getText(self.accaunt\_wigit, 'Login', 'Введите пароль от вашего аккаунта:')

if b[1]:

hash\_object = hashlib.sha256(b[0].encode('UTF-8'))

hex\_dig = hash\_object.hexdigest()

if accaunt[2] == hex\_dig:

self.accaunt = accaunt

self.accaunt\_button()

dlg = QMessageBox(self)

dlg.setWindowTitle("Успех!")

dlg.setText(f"Вы зашли в аккаунт {accaunt[1]}")

dlg.exec()

else:

return

else: # "Упс...", "Аккаунта с таким именем нет в базе."

dlg = QMessageBox(self)

dlg.setWindowTitle("Упс...")

dlg.setText("Аккаунта с таким именем нет в базе...")

dlg.exec()

def registred(self):

a = QInputDialog()

name = a.getText(self.accaunt\_wigit, 'Registr', 'Введите имя вашего аккаунта:')

accaunt = self.db.get\_line('accaunts', 'name', name[0])

if accaunt:

dlg = QMessageBox(self)

dlg.setWindowTitle("Упс...")

dlg.setText("В базе уже есть такой аккаунт...")

dlg.exec()

self.registred()

return

a = QInputDialog()

passwd = a.getText(self.accaunt\_wigit, 'Registr', 'Введите пароль от вашего аккаунта:')

if passwd[1]:

hash\_object = hashlib.sha256(passwd[0].encode('UTF-8'))

hex\_dig = hash\_object.hexdigest()

self.db.add\_value('accaunts', None, name[0], hex\_dig, 0, 0)

accaunt = self.db.get\_line('accaunts', 'name', name[0])

self.accaunt = accaunt

self.accaunt\_button()

dlg = QMessageBox(self)

dlg.setWindowTitle("Успех!")

dlg.setText(f"Вы создали аккаунт {accaunt[1]}")

dlg.exec()

else:

return

# def export\_results(self):

# writer = None

# with open('results.csv', 'w', encoding='utf-8') as file:

# writer = csv.DictWriter(file, ['theme\_name', 'your\_balls', 'max\_balls'])

# results = self.db.get\_lines('results', 'name', self.accaunt[1])

# results\_csv = []

# for i in results:

# results\_csv.append({'theme\_name': i[2], 'your\_balls': i[3], 'max\_balls': i[4]})

# writer.writerows(results\_csv)

# dlg = QMessageBox(self)

# dlg.setWindowTitle("Успех!")

# dlg.setText("Файл экспортирован! Он лежит рядом с исполняемым файлом под названием results.csv")

# dlg.exec()

def img\_to\_label(self, img\_road, label):

pixmap = QPixmap(img\_road)

if pixmap.width() > label.width() and pixmap.height() <= label.height(): # Подгон размеров картинки под label

otn = pixmap.width() / pixmap.height()

pixmap = pixmap.scaled(label.width(), math.ceil(label.width() / otn))

elif pixmap.width() <= label.width() and pixmap.height() > label.height():

otn = pixmap.width() / pixmap.height()

pixmap = pixmap.scaled(int(label.height() \* otn), label.height())

elif pixmap.width() > label.width() and pixmap.height() > label.height():

otn = pixmap.width() / pixmap.height()

pixmap = pixmap.scaled(label.width(), math.ceil(label.width() / otn))

if pixmap.width() <= label.width() and pixmap.height() > label.height():

otn = pixmap.width() / pixmap.height()

pixmap = pixmap.scaled( math.ceil(label.height() \* otn), label.height())

if pixmap.width() < label.width() and pixmap.height() < label.height():

otn = pixmap.width() / pixmap.height()

pixmap = pixmap.scaled(int(label.height() \* otn), label.height())

if pixmap.width() > label.width() and pixmap.height() <= label.height():

otn = pixmap.width() / pixmap.height()

pixmap = pixmap.scaled(label.width(), math.ceil(label.width() / otn))

# Задавание пиксмапа

label.setPixmap(pixmap)

def change\_color(self):

global ch\_flag, color\_now

color\_now += 1

col = init\_style\_sheet()

if color\_now >= len(col):

color\_now = 0

app.setStyleSheet(col[color\_now][0].strip())

def init\_style\_sheet() -> tuple[tuple[str, tuple], ...]:

file = None

with open('json/styles.txt') as f:

file = f.readlines()

all\_styles = []

temp = []

colors = False

for i in file:

if i.strip() == '=123' and colors is False:

colors = temp

temp = ''

elif i.strip() == '#---------------------------------------------------------------------------------------------------------------------------------':

temp = temp.replace('{a}', colors[4]).replace('{b}', colors[3]).replace('{c}', colors[2]).replace('{d}', colors[1]).replace('{e}', colors[0]).replace('{font}', colors[-1]).replace('{blind}', colors[5])

all\_styles.append((temp, colors))

colors = False

temp = []

elif bool(colors) == False:

if i:

temp += [i.strip()]

else:

temp += i

return tuple(all\_styles)

def main():

global ch\_flag, app, color\_now

color\_now = 0

app = QApplication(sys.argv)

app.setStyleSheet(init\_style\_sheet()[color\_now][0])

Main\_window = Main\_Window()

Main\_window.show()

sys.exit(app.exec())

if \_\_name\_\_ == '\_\_main\_\_':

main()