ДЕПАРТАМЕНТ ОБРАЗОВАНИЯ И НАУКИ ГОРОДА МОСКВЫ

Государственное бюджетное общеобразовательное учреждение города Москвы Школа № 1502 «Энергия»

**ПРОЕКТ**

по информатике на тему

**«Разработка приложения для изучения технического английского языка»**

Исполнитель: ученик 10Т класса

Коныгина М.А.

Руководитель: учитель информатики

Кузьмина Е. Ф.

Москва, 2024

ОГЛАВЛЕНИЕ

[Введение 3](#_Toc156755897)

[1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 5](#_Toc156755898)

[1.1 Технический английский 5](#_Toc156755899)

[1.2 Языки программирования 7](#_Toc156755900)

[1.3 Библиотеки Python 9](#_Toc156755901)

[2. ТЕХНОЛОГИЧЕСКИЙ РАЗДЕЛ 11](#_Toc156755902)

[2.1 Методы и библиотеки 11](#_Toc156755903)

[2.3 Руководство для пользователя 13](#_Toc156755904)

[2.3 Формат тестирования 15](#_Toc156755905)

[2.4 Словарь 16](#_Toc156755906)

[Список литературы 19](#_Toc156755907)

# Введение

В современном мире начать собственное дело без IT практически невозможно. Иногда, это простые программы которые упрощают и систематизируют рабочий процесс, а иногда сложное ПО, которому нужна поддержка специально обученного человека. Даже если ваша деятельность не связана с технологическими решениями, вы все равно будете зависеть от IT. Английский язык в профессии программиста стоит на втором месте после специализированных знаний. Многие начинающие утверждают, что английский язык для программистов совершенно не важен и что главное, получить квалификацию и найти заинтересованных в твоей работе клиентов. На самом деле это заблуждение, ведь все технологические новинки и инновационное ПО сначала появляется в США и Европе, а значит, на английском языке.

Программист со знанием английского языка имеет явные преимущества над теми, кто его не знает, ведь у него есть возможность с первых моментов ориентироваться в нововведениях и современных новинках.

С развитием компьютерных технологий обучение языкам стало гораздо удобнее и интерактивнее. Интернет позволил появиться такой форме образования, как дистанционное обучение. Среди электронных форм обучения иностранным языкам одно из ведущих мест в последние годы занимают приложения. Поэтому разработка тренажера для изучения технического английского языка **актуальна** в наши дни.

**Цель**: разработка обучающего приложения для изучения технического английского языка на языке Python.

Для достижения цели необходимо решить следующие **задачи**:

* изучить термины в техническом английском;
* отобрать и систематизировать материал;
* изучить возможности языка Python, библиотеки PyQT5 и QT Designer;
* продумать функционал и интерфейс для приложения;
* разработать программный код;
* произвести отладку и корректировку проекта.

# 1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

1.1 Технический английский

Технический английский язык — это специализированная разновидность английского языка, используемая в области науки, техники и инженерии. Он содержит специфическую терминологию и грамматические конструкции, которые помогают точно и четко выражать технические и научные идеи. Сложность понимания технического стиля заключается в применении инверсии — обратного порядка слов, а также использования сложных аббревиатур и длинных цифр. Статьи на узкопрофильные темы кажутся непонятными простому человеку, даже если они написаны на русском языке. Поэтому для изучения таких терминов нужно овладеть базовым английским, постепенно усложняя программу. Сложность специализированного языка заключается даже не в терминах, а в умении работать с разными значениями английских слов. Неправильный перевод может исковеркать весь смысл текста, поэтому нужно знать и сам технический предмет, чтобы понять контекст.

Когда дело доходит до программирования, английский язык предпочтительнее, потому что он разрушает коммуникационные барьеры, которые могут существовать между разными программистами. Английский является наиболее объединяющим языком. Также большинство англоязычных стран были «домами» информатики. Вначале информатика преподавалась в США, Великобритании и Канаде. Другие страны начали сосредотачиваться на этом предмете годы спустя. Каждому IT-специалисту важно владеть английским языком на хорошем уровне. Для этого есть много причин:

* большинство языков программирования основано на ключевых словах английского происхождения, без их знания понимать и писать коды для вас будет очень долгими и сложным процессом;
* практически все справочные материалы и технические задания пишутся на английском;
* документации для различных языков программирования всегда пишутся на английском, хоть мы и можем найти переводы в интернете, но для лучшего понимаю оригинал всегда лучше;
* программистам нужно быть в курсе всех новостей и обновлений, а новейшие книги и статьи в IT-сфере публикуются на английском;
* в своей работе IT-специалисты периодически сталкиваются со сложными задачами и некоторыми проблемами, решение которых можно найти в англоязычной интернет-среде;
* понимание сленга, используемого программистами.

Часто слушая речь работников IT сферы, человеку, не ознакомленному со сленгом, обычно образованным от английских слов, может быть проблематично понять, о чем говорят его коллеги.

Пример:

Fork (вилка) – новое приложение, разработанное на основе существующего. Когда приложение "разветвляется", оно создает новую отдельную программу, а не новую ветвь разработки.

Как можно понять из приведенных выше примеров, уверенное владение техническим английским не только позволяет специалисту существенно вырасти в профессиональном плане, но и дает ему неоспоримое преимущество на рынке труда.

Типы терминов, используемых программистами:

* ключевые слова – в языках программирования слово, имеющее специальное значение;

Примеры:

* String – тип данных, используемый для представления текста.
* Class – это шаблонное определение методов и переменных в объекте определенного типа.
* While – используется для повторения определенного блока кода неизвестное количество раз, пока не будет выполнено условие.
* термины, используемые в документации для описания того, как язык/ библиотека работают, такие термины одинаковы для всех языков программирования;

Примеры:

* Function definition – определение функции.
* Initializing the module – инициализация модуля.
* термины, используемые при разработке.

Примеры:

* Backup –копия одного или нескольких файлов, созданных в качестве альтернативных в случае потери исходных данных или невозможности их использования.
* Debugging – поиск и исправление ошибок в программе.
* Encoding – процесс преобразования данных из одной формы в другую.

## 1.2 Языки программирования

Для разработки приложений используются разные языки программирования. Давайте рассмотрим некоторые из них.

C++ – считается отличным выбором для разработки мобильных приложений. Это универсальный объектно-ориентированный язык с общими и низкоуровневыми функциями управления памятью.

Плюсы:

* быстрый и стабильный, имеет обширный набор готовых к использованию встроенных библиотек и компиляторов;
* если вы знаете C ++, вы можете быстро выучить JAVA, C, C #;
* можно получить полный контроль, и приложение может работать на любой платформе и использовать преимущества любого оборудования.

Минусы:

* сложный синтаксис, небольшая стандартная библиотека;
* самый сложный язык программирования для освоения;
* объектная ориентация весьма важна по сравнению с другими языками.

JavaScript – лучше всего подходит для кросс-платформенной разработки приложений, мобильных приложений, используемых для различных платформ, и просмотра [веб](https://notissimus.com/luchshie-frejmvorki-dlya-veb-razrabotki-kotorye-stoit-vybrat-v-2023-godu/)-страниц.

Плюсы:

* универсальный и гибкий язык, который может использоваться многими способами благодаря Node.js;
* более быстрый просмотр на стороне клиента и не требует компиляции;
* простой в работе и простой в управлении.

Минусы:

* уязвим и в некоторых случаях может быть использован в злонамеренных целях;
* сценарии на стороне сервера всегда обеспечивают одинаковый результат, а на стороне клиента это немного непредсказуемо;
* в некоторых случаях вы можете столкнуться с некоторыми проблемами с поддержкой браузера.

Python – широко распространенный язык программирования используется для разработки веб-приложений, настольных приложений, медиа-инструментов, машинного обучения, сетевых серверов и многого другого. Примерами приложений, написанных на Python, являются YouTube, Instagram и Pinterest.

Плюсы:

* быстрый, простой в использовании, быстрый в развертывании, простой в освоении и отлично читаемый;
* язык программирования высокого уровня, используемый для анализа, вычисления научных и числовых данных, а также для разработки программного обеспечения;
* интерактивный язык, используемый для создания масштабируемых приложений и поддержки приложений с графическим интерфейсом.

Минусы:

* медленное время выполнения;
* не лучший выбор для задач, требующих большого объема памяти;
* трудность переноса проектов на другие системы.

## 1.3 Библиотеки Python

Библиотеки – это готовые модули кода, которые используют разработчики, чтобы не писать один и тот же код несколько раз. Python имеет большое количество библиотек.

Стандартные библиотеки Python – это модули кода, которые доступны разработчикам без дополнительной установки.

Например, библиотека math в Python дает возможность выполнять самые разные математические функции. Библиотека os Python необходима для работы с файловой системой компьютера и операционной системой. Другие стандартные библиотеки – random и datetime. Первая дает инструменты для работы со случайными числами, а вторая незаменима для программ, которые оперируют временем и датами.

Стандартные библиотеки Python не требуют отдельного подключения. Чтобы использовать их функционал, достаточно просто в начале программы написать import и название библиотеки.

Рассмотрим другие библиотеки Python.

Pandas – это библиотека обработки и анализа данных, которая предоставляет широкий спектр инструментов для манипуляции данными, включая чтение и запись данных из различных источников, манипулирование индексами и колонками, объединение и разделение данных и многое другое. Numpy – это фундаментальная библиотека для работы с многомерными массивами (или “векторами”) в Python. С помощью Numpy вы можете выполнять различные математические операции над этими массивами, включая сложение, вычитание, умножение, а также более сложные операции, такие как линейная регрессия или PCA (метод главных компонент). [Python Imaging Library](https://pillow.readthedocs.io/en/stable/), или PIL – это бесплатная библиотека Python, которая позволяет интерпретатору Python обрабатывать изображения. Иными словами, PIL дает возможность обрабатывать, открывать и сохранять различные форматы изображений в Python. [Pygame](https://www.pygame.org/) – это бесплатная библиотека Python с открытым исходным кодом. Она предназначена для разработки мультимедийных приложений на Python, в частности, двумерных игр. И как результат, она популярна среди как обычных, так и профессиональных разработчиков игр на Python.

# 2. ТЕХНОЛОГИЧЕСКИЙ РАЗДЕЛ

## 2.1 Методы и библиотеки

Для разработки данного приложения будет использоваться библиотеку PyQt в Python. PyQt – это библиотека Python для создания приложений с графическим интерфейсом с помощью инструментария Qt. Кроме того, позволяет работать с базами данных и другими внешними источниками данных. PyQt является открытым исходным кодом и распространяется под лицензией GPL. Для создания интерфейса в приложении используется Qt Designer. Он позволяет создавать графические интерфейсы пользователя при помощи ряда инструментов. Существует панель инструментов «Панель виджетов», в которой доступны для использования элементы интерфейса – виджеты, такие как, например, «выпадающий список» ComboBox, «поле ввода» LineEdit, «кнопка» PushButton и многие другие. Каждый виджет имеет свой набор свойств, определяемый соответствующим ему классом библиотеки Qt. Свойства виджета могут быть изменены при помощи «Редактора свойств». Для каждого класса свойств виджета существует свой специализированный редактор.

Для осуществления регистрации используется база данных созданная в SQLite. Это однофайловая СУБД, которая не имеет сервера и позволяет хранить всю базу локально на одном устройстве. СУБД – это система управления базами данных, которая позволяет хранить, изменять и извлекать данные. SQLite имеет ряд преимуществ, которые делают ее привлекательной для разработчиков. Во-первых, она очень компактна, что позволяет использовать ее на устройствах с ограниченными ресурсами. Во-вторых, она быстро работает, так как не требует дополнительных процессов или серверов. В-третьих, SQLite очень проста в использовании, что делает ее идеальной для начинающих разработчиков. Для работы с базами данных в Python используется встроенная библиотека sqlite3. Для установки соединения нужно указать название базы данных, к которой требуется подключиться.

Термины и их значения записываются в csv файлы. Python имеет встроенную библиотеку csv, которая предоставляет функции для чтения и записи данных в формате CSV.

Используемые методы:

* MainWindow() – главный экран для входа или регистрации;
* OpenWindow() – пустое окно на котором открывается главное окно, вход и регистрация;
* RegWindow() – окно регистрации;
* SignWindow() – окно для входа в аккаунт;
* Testing () – пустое окно с меню, в котором будут показываться все виджеты;
* MainPage() – окно с результатами и логином пользователя;
* DevelopTestWindow() – окно теста с словами для разработки;
* KeywordsTestWindow () – окно теста с ключевыми словами;
* ComputerTestWindow() – окно теста с частями компьютера;
* TermsTestWindow() – окно теста со вставлением терминов;
* KeywordsWindow() – окно теории с ключевыми словами;
* DevelopWindow() – окно теории с словами для разработки;
* TermsWindow() – окно словами Back-end разработки;
* ComputerWindow() – окно теории с частями компьютера;
* DictionaryWindow() – окно со словарем;
* ChooseWord() – окно с добавлением нового слова в словарь.

## 2.3 Руководство для пользователя

Для открытия проекта требуется открыть файл «TechEng.exe» (рис. 1).



Рисунок 1. Иконка приложения

Для корректной работы приложения требуется:

* операционная система «Windows 7/8/10», Linux или macOS;
* место на диске 100Мб;
* 64-разрядный процессор и операционная система;
* Доступ к интернету для скачивания приложения;
* приложение открытия exe файлов для Linux и macOS (например, Wine)

2.2 Структура приложения

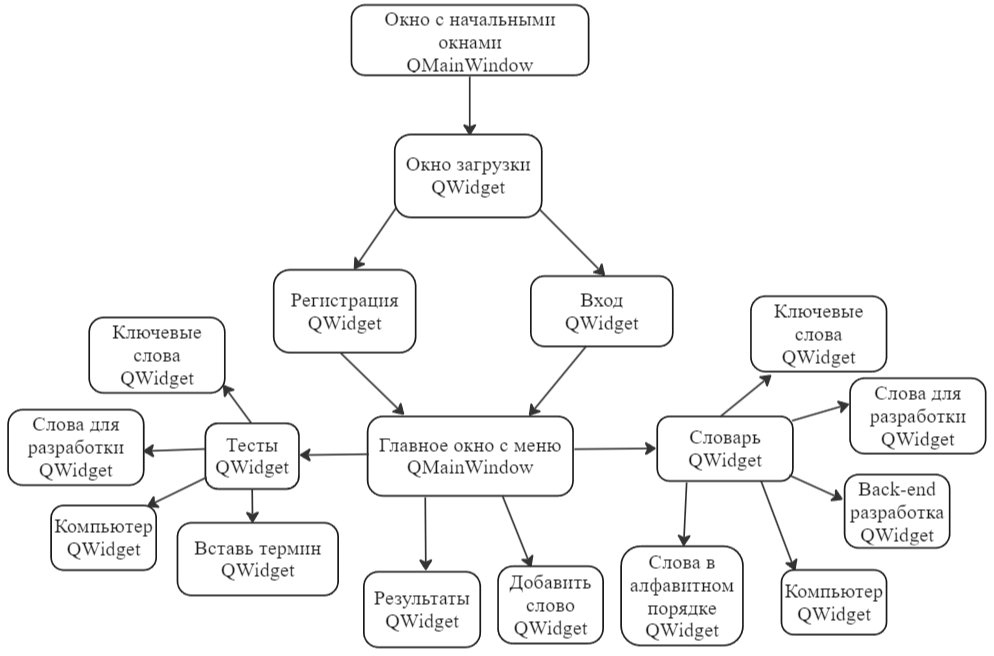


Рисунок 2. Структура приложения

При загрузке приложения появляется окно в котором пользователь должен выбрать “Регистрацию” или “Вход”. Для регистрации требуется логин из более чем 3 символов и пароль (рис. 3).

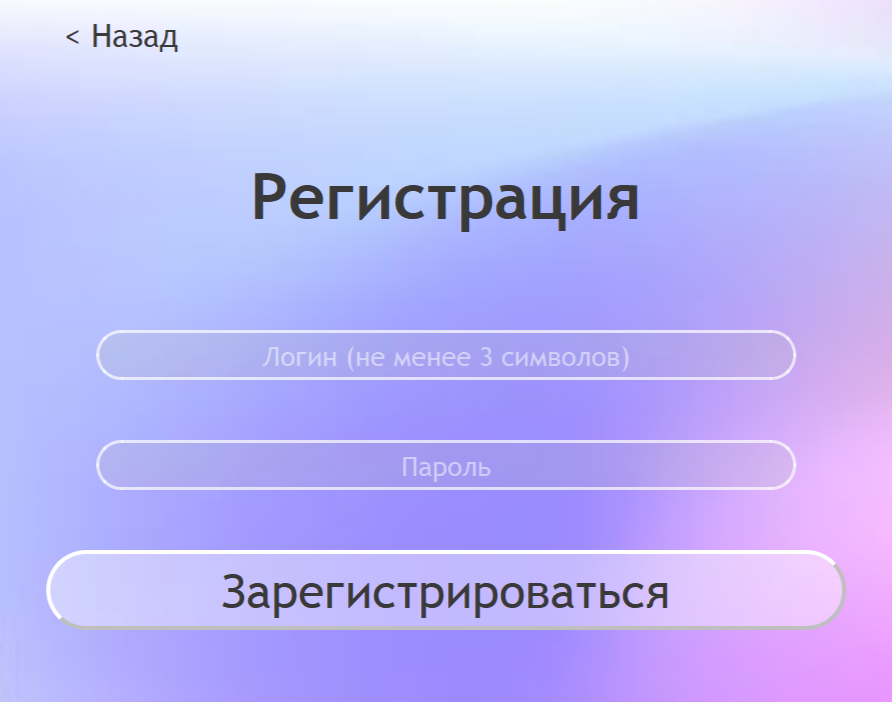


Рисунок 3. Регистрация

После входа пользователь видит главный экран предложения. На нем можно увидеть ваш логин, количество правильных ответов за последние пройденные тесты (рис. 4).

В меню в верхней части экрана находятся меню, где можно выбрать тесты и теорию для изучения, результаты или добавление нового слова.

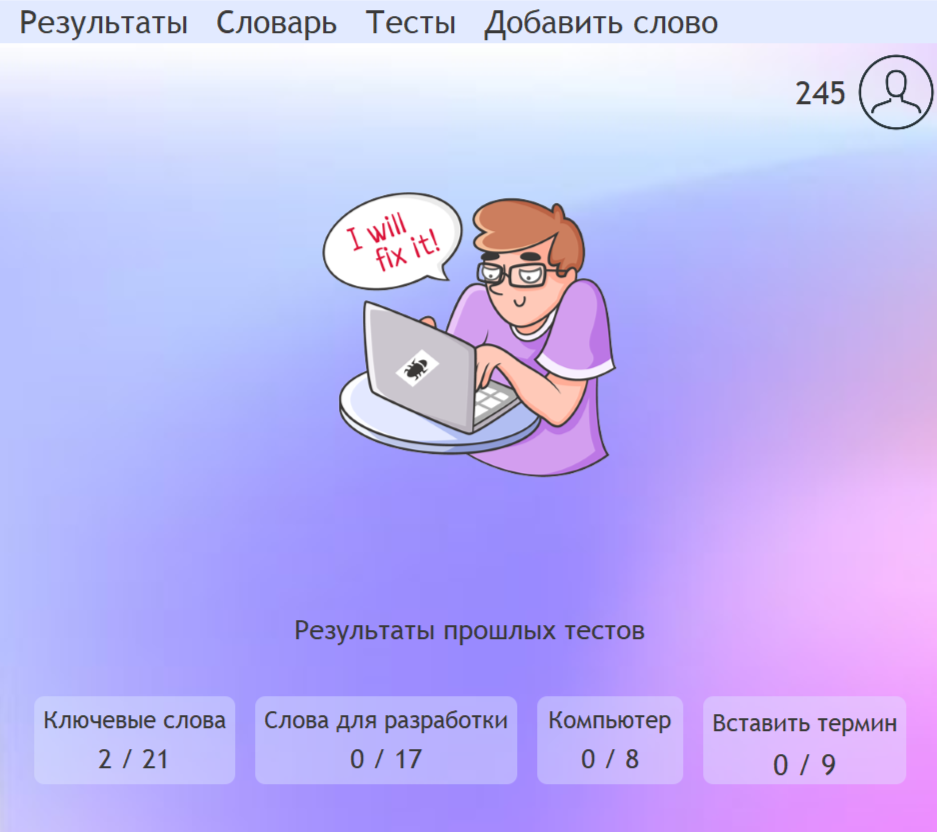


Рисунок 4. Главный экран

## 2.3 Формат тестирования

Во вкладке теория есть несколько типов тестирования.

Знание определений можно проверить в тестах «Ключевые слова» и «Слова для разработки». Для выбора ответа нужно нажать кнопку номера ответа. Она загорится красным или зеленым, соответствии с правильностью (рис. 5).

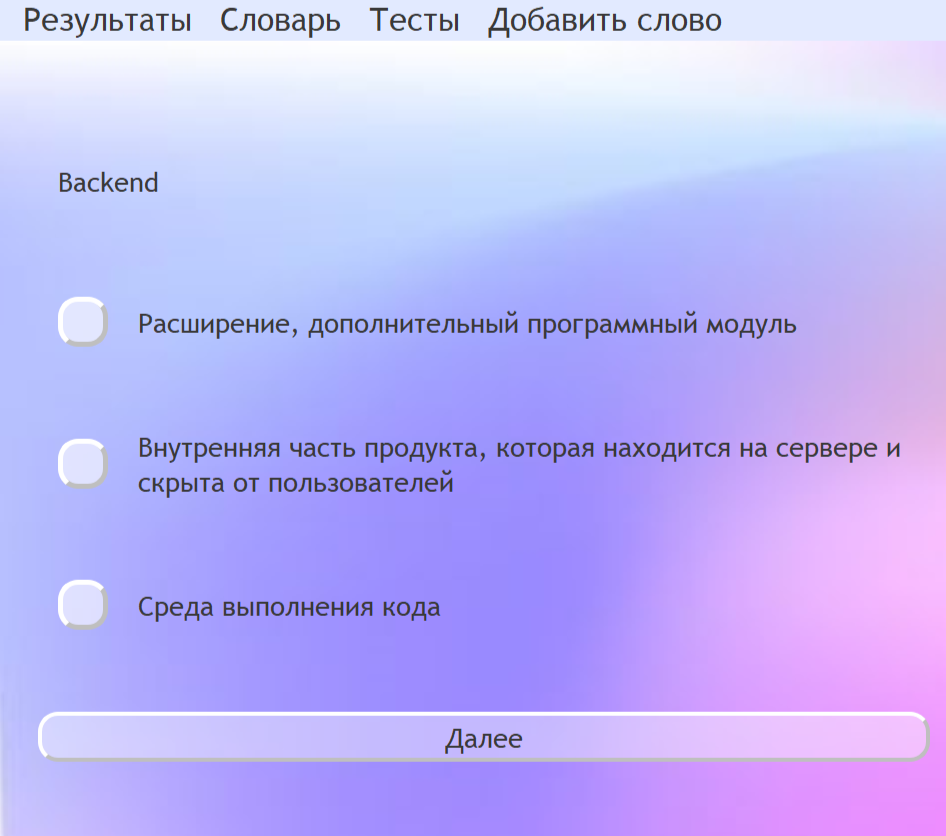


Рисунок 5. Тестирование

После прохождения теста высвечивается окно с количеством правильных ответов. Они записываются в базу данных (рис. 6).

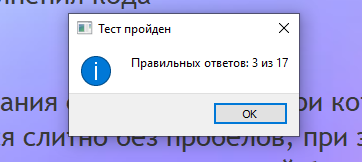


Рисунок 6. Окно окончания тестирования

При прохождении тестирования с элементами компьютера, пользователю нужно вести правильное название элемента на английском в окно с рядом с изображением (рис. 7).

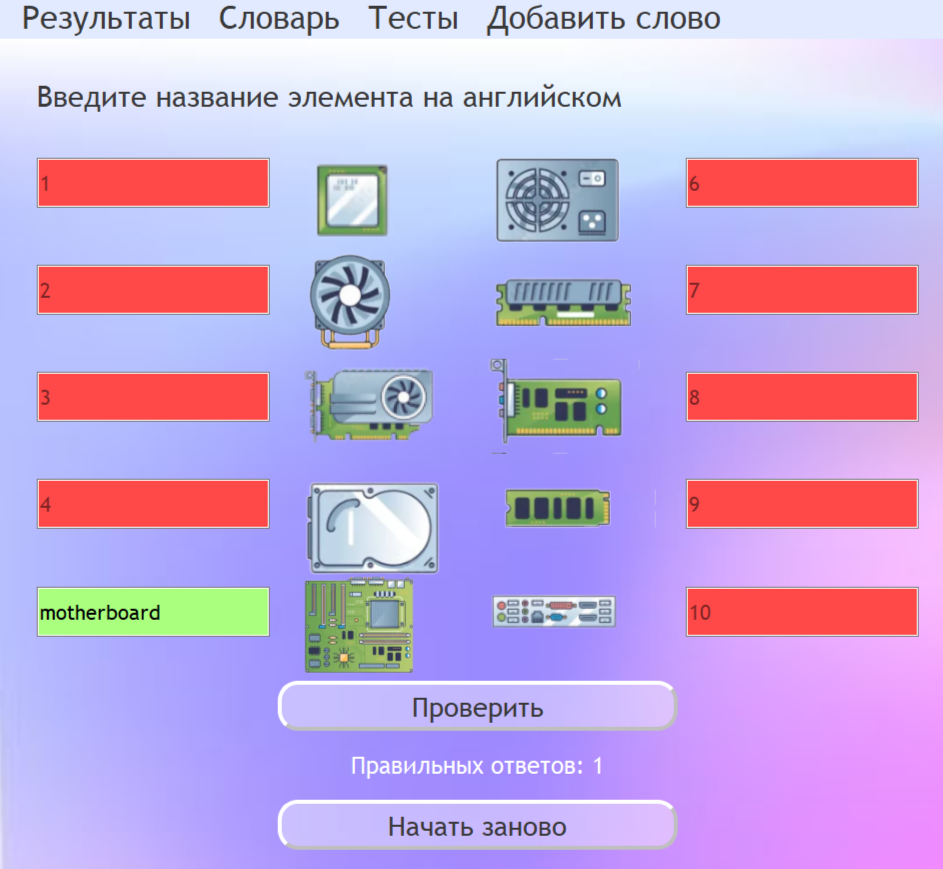


Рисунок 7. Тестирование элементов компьютера

## 2.4 Словарь

Для изучения слов есть раздел «Словарь». Слова разделены как по темам, так и в алфавитном порядке. При нажатии на слово внизу появляется его значение (рис. 8).

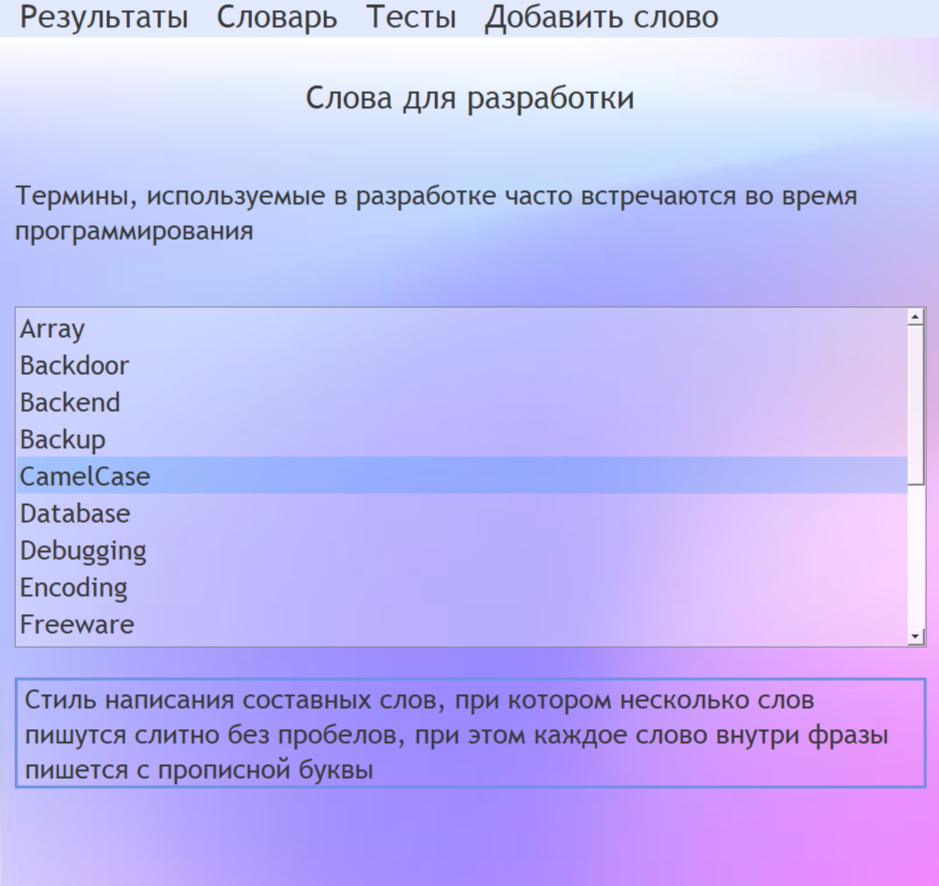


Рисунок 8. Слова в алфавитном порядке

В разделе «Компьютер» можно ознакомиться с названиями компонентов компьютера на русском и английском (рис. 9).



Рисунок 9. Раздел "Компьютер"

Добавлять новые слова в словарь можно в разделе «Добавить слово» в меню. Для этого нужно записать слово и его значение, затем выбрать тип слова и нажать кнопку “Добавить слово” (рис. 10)

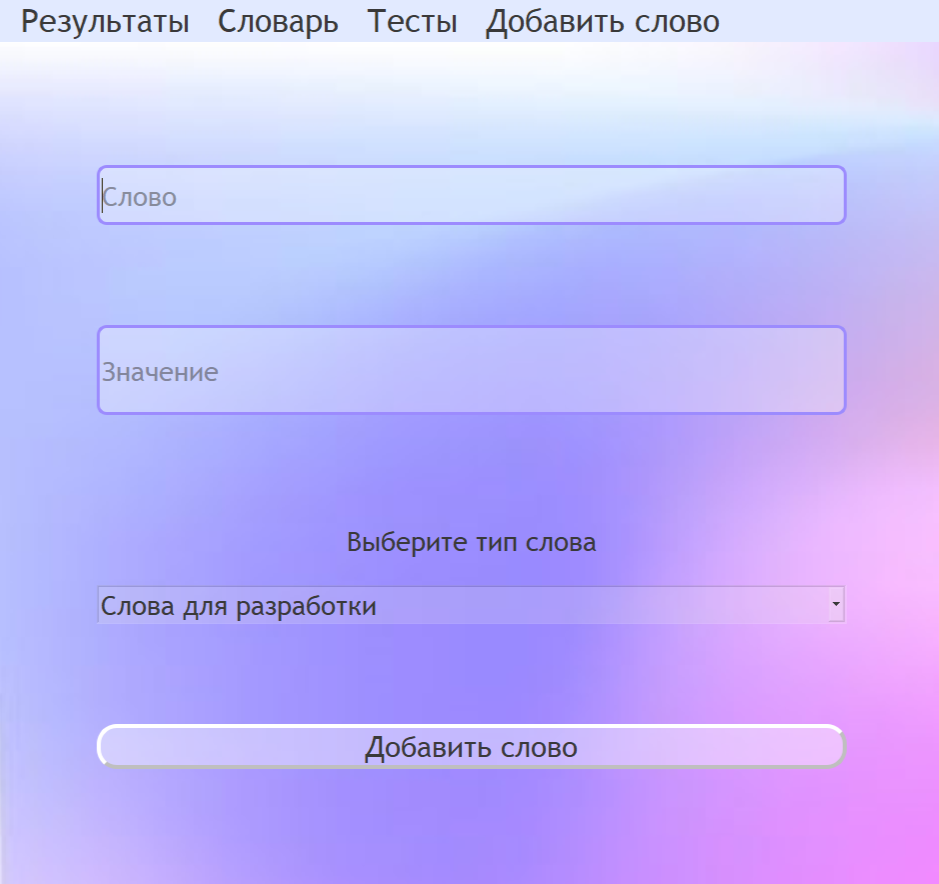


Рисунок 10. Добавить слово в словарь

Заключение

В результате проделанной работы было создано приложение, которое помогает программистам изучить и понять разные технические термины. Были решены все поставленные задачи. Было изучено большое количество технических терминов и информации о библиотеках языка Python. Был создан интерфейс и система тестирования пройденной информации. Проект может быть использован в образовательных учреждениях, на домашних компьютерах учеников или программистов.

В процессе работы было решено большое количество задач: ошибки в коде, неправильное отображение изображений, ошибки в использовании баз данных и открытии окон, отображение приложения на разных ОС. В проекте были использованы решения, которые используются при разработке приложений с использованием Qt.

В будущем проект может быть доработан: путём добавления новых заданий и теории, и рейтинга пользователей. Проект, в виду развития IT сферы становится всё более актуальным.

# Список литературы

1. Бжилянская Г. М. Английский язык для студентов технических вузов. English for Students at Higher Technical Schools: учебное пособие для вузов / Г. М. Бжилянская. 2-е изд. стер. - Санкт-Петербург: Лань, 2023. - 316 c.: ил. - Текст: непосредственный.
2. Английский язык для технических специальностей: учеб. пособие / О. А. Витрук. - Ростов н/Д: Феникс, 2024. - 155 с. - (Среднее профессиональное образование).
3. Python - это просто. Пошаговое руководство по программированию и анализу данных: Пер. с англ. - СПб.: БХВ-Петербург, 2022. - 416 с.г ил.
4. Python 3 и Ру 5. Разработка приложений. - 2-е изд., перераб. и доп. / Н. А. Прохоренок, В. А. Дронов. - СПб.: БХВ-Петербург, 2018. - 832 с.: ил. - (Профессиональное программирование)
5. Изучаем Python: программирование игр, визуализация данных, веб-приложения. 3-е изд. - СПб.: Питер, 2021. - 512 с.: ил. - (Серия «Библиотека программиста»).
6. Английский для IT. Как учиться программисту + 84 ресурса [Электронный ресурс]. Режим доступа URL: https://englex.ru/english-for-it-specialists/
7. 16 лучших языков программирования для разработки мобильных приложений 2023 [Электронный ресурс]. Режим доступа URL: <https://notissimus.com/16-luchshih-yazykov-programmirovaniya-dlya-razrabotki-mobilnyh-prilozhenij-2023/#Python>
8. PyQt6 - полное руководство для новичков [Электронный ресурс]. Режим доступа URL: <https://habr.com/ru/companies/skillfactory/articles/599599/>
9. Python and PyQt: Creating Menus, Toolbars, and Status Bars [Электронный ресурс]. Режим доступа URL: <https://realpython.com/python-menus-toolbars/#connecting-signals-and-slots-in-menus-and-toolbars>
10. TechTerms. The Computer Dictionary [Электронный ресурс]. Режим доступа URL: <https://techterms.com/browse/>
11. Английский язык для программиста: how to [Электронный ресурс]. Режим доступа URL: <https://blog.skillfactory.ru/anglijskie-slova-i-termini-it/>
12. Как работать с модулем sqlite3 в Python [Электронный ресурс]. Режим доступа URL: <https://sky.pro/media/kak-rabotat-s-modulem-sqlite3-v-python/>
13. Яндекс Лицей [Электронный ресурс]. Режим доступа URL: https://lyceum.yandex.ru/

import sys  
import io  
import sqlite3  
import csv  
  
from random import choice, sample, randint  
from PyQt5 import uic, QtCore  
from PyQt5.QtWidgets import QApplication, QMainWindow, QMessageBox, QDialog, QWidget, QAction, QMenu  
from PyQt5.QtGui import QPixmap  
  
*#окно где открывается главное окно, окно регистрации и входа*class OpenWindow(QMainWindow):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 uic.loadUi('openingpage.ui', self)  
 *# объявляем базу данных* self.con = sqlite3.connect('TechEngDB.sqlite')  
 self.show()  
 self.showMain()  
  
 *#лавное окно* def showMain(self):  
 mainWindow = MainWindow(self.con, self)  
 self.setCentralWidget(mainWindow)  
 mainWindow.registr.clicked.connect(self.regButton)  
 mainWindow.signin.clicked.connect(self.signButton)  
 mainWindow.show()  
 *#кнопка показывающая окно регистрации* def regButton(self):  
 regWindow = RegWindow(self.con, self)  
 self.setCentralWidget(regWindow)  
 regWindow.backButton.clicked.connect(self.showMain)  
 regWindow.show()  
  
 *# #кнопка показывающая окно входа* def signButton(self):  
 signWindow = SignWindow(self.con, self)  
 self.setCentralWidget(signWindow)  
 signWindow.backButton.clicked.connect(self.showMain)  
 signWindow.show()  
 *#открывается окно приложения* def openTesting(self, userId):  
 self.testingWindow = Testing(userId)  
 try:  
 self.testingWindow.show()  
 except Exception as err:  
 print(err)  
 self.hide()  
  
*# главное окно*class MainWindow(QWidget):  
 def \_\_init\_\_(self, con, openingwindow):  
 super().\_\_init\_\_(openingwindow)  
 uic.loadUi('main.ui', self)  
 self.setStyleSheet('.QWidget {background-image: url(69c461fe46c248901150bfd7f23ea340.jpg);}')  
  
  
*# окно регистрации*class RegWindow(QWidget):  
 def \_\_init\_\_(self, connection, openingwindow):  
 super().\_\_init\_\_(openingwindow)  
 self.con = connection  
 uic.loadUi('reg.ui', self)  
 self.regButton.clicked.connect(self.addData)  
 self.setStyleSheet('.QWidget {background-image: url(8b5504825ea01ea1eb747632b8428926.png);}')  
 self.openingwindow = openingwindow  
  
 *# добавляем пользователя* def addData(self):  
 try:  
 login = self.regLogin.text()  
 password = self.regPassword.text()  
 cur = self.con.cursor()  
 insert\_data = """INSERT INTO users  
 (login, password)  
 VALUES  
 (?, ?)"""  
 *# проверка введеных данных* if login == '':  
 QMessageBox.critical(self, 'Ошибка', 'Логин не введен')  
  
 elif len(login) < 3:  
 QMessageBox.critical(self, 'Ошибка', 'Недостаточно символов')  
  
 elif password == '':  
 QMessageBox.critical(self, 'Ошибка', 'Пароль не введен')  
  
 elif len(cur.execute('''SELECT login FROM users  
 WHERE login = ?''', (login,)).fetchall()) != 0:  
 QMessageBox.critical(self, 'Ошибка', 'Такой пользователь уже существует')  
 except Exception as error:  
 print(error)  
 else:  
  
 *# регистрируем человека в базу данных* try:  
 cur.execute(insert\_data, (login, password))  
 self.con.commit()  
 self.hide()  
 users = cur.execute('''SELECT id, password FROM users  
 WHERE login = ?''', (login,)).fetchall()  
 print(users)  
 userId = users[0][0]  
 print(userId)  
 self.openingwindow.openTesting(userId)  
 except Exception as err:  
 print(err)  
  
  
*# окно входа*class SignWindow(QWidget):  
 def \_\_init\_\_(self, connection, openingwindow):  
 super().\_\_init\_\_(openingwindow)  
 uic.loadUi('sign.ui', self)  
 self.signButton.clicked.connect(self.logIn)  
 self.con = connection  
 self.setStyleSheet('.QWidget {background-image: url(8b5504825ea01ea1eb747632b8428926.png);}')  
 self.openingwindow = openingwindow  
  
 def logIn(self):  
 try:  
 login = self.signLogin.text()  
 password = self.signPassword.text()  
 cur = self.con.cursor()  
 *# ищем логин в базе данных и запоминаем айди и пароль* users = cur.execute('''SELECT id, password FROM users  
 WHERE login = ?''', (login,)).fetchall()  
 *# проверка введеных данных* if login == '':  
 QMessageBox.critical(self, 'Ошибка', 'Логин не введен')  
  
 elif password == '':  
 QMessageBox.critical(self, 'Ошибка', 'Пароль не введен')  
  
 elif len(users) == 0:  
 QMessageBox.critical(self, 'Ошибка', 'Такой пользователь не существует')  
  
 elif password != users[0][1]:  
 QMessageBox.critical(self, 'Ошибка', 'Неверный пароль')  
  
 else:  
 *# записываем айди, чтобы его позже передать в следующее окно* userId = users[0][0]  
 self.openingwindow.openTesting(userId)  
 except Exception as err:  
 print(err)  
  
*#пустое окно с меню, где открываются все окна*class Testing(QMainWindow):  
 def \_\_init\_\_(self, userId):  
 super().\_\_init\_\_()  
 uic.loadUi('mainpage.ui', self)  
   
 self.con = sqlite3.connect('TechEngDB.sqlite')  
 self.setStyleSheet('.QWidget {background-image: url(8b5504825ea01ea1eb747632b8428926.png);}')  
 *# айди пользователя* self.id = userId  
 self.onMainAction()  
 *# подключаем меню* self.mainMenu = QMenu("Результаты", self)  
 self.menubar.addMenu(self.mainMenu)  
   
 self.mainPageAction = QAction("Результаты", self)  
 self.mainMenu.addAction(self.mainPageAction)  
 self.mainPageAction.triggered.connect(self.onMainAction)  
  
 self.theoryMenu = QMenu("Словарь", self)  
 self.menubar.addMenu(self.theoryMenu)  
   
 self.testsMenu = QMenu("Тесты", self)  
 self.menubar.addMenu(self.testsMenu)  
  
 self.newWordMenu= QMenu("Добавить слово", self)  
 self.menubar.addMenu(self.newWordMenu)  
  
   
 *#добавляем элементы в меню  
 #меню с заданиями* self.newWordAction = QAction("Добавить слово", self)  
 self.newWordMenu.addAction(self.newWordAction)  
 self.newWordAction.triggered.connect(self.onNewWordAction)  
  
 self.keywordstestingAction = QAction("Ключевые слова", self)  
 self.testsMenu.addAction(self.keywordstestingAction)  
 self.keywordstestingAction.triggered.connect(self.onKeywordsAction)  
  
 self.developtestingAction = QAction("Слова для разработки", self)  
 self.testsMenu.addAction(self.developtestingAction)  
 self.developtestingAction.triggered.connect(self.onDevelopAction)  
  
 self.comptestingAction = QAction("Компьютер", self)  
 self.testsMenu.addAction(self.comptestingAction)  
 self.comptestingAction.triggered.connect(self.onCompTesting)  
  
 self.termstestingAction = QAction("Вставь термин", self)  
 self.testsMenu.addAction(self.termstestingAction)  
 self.termstestingAction.triggered.connect(self.onTermsTesting)  
  
 *#меню с теорией* self.keywordstheoryAction = QAction("Ключевые слова", self)  
 self.theoryMenu.addAction(self.keywordstheoryAction)  
 self.keywordstheoryAction.triggered.connect(self.onKeywordsThAction)  
  
 self.developtheoryAction = QAction("Слова для разработки", self)  
 self.theoryMenu.addAction(self.developtheoryAction)  
 self.developtheoryAction.triggered.connect(self.onDevelopThAction)  
  
 self.comptheoryAction = QAction("Компьютер", self)  
 self.theoryMenu.addAction(self.comptheoryAction)  
 self.comptheoryAction.triggered.connect(self.onCompTheory)  
  
 self.termstheoryAction = QAction("Back-end разработка", self)  
 self.theoryMenu.addAction(self.termstheoryAction)  
 self.termstheoryAction.triggered.connect(self.onTermsTheory)  
 self.setStyleSheet('''.QMenu {background-color: rgb(226, 234, 255 );  
 font-size: 20px;  
 font-family: "Trebuchet MS", "Lucida Grande",  
 "Lucida Sans Unicode", "Lucida Sans", Tahoma, sans-serif;  
 color:rgb(57, 57, 57);}''')  
 self.dictionaryAction = QAction('Слова в алфавитном порядке')  
 self.theoryMenu.addAction(self.dictionaryAction)  
 self.dictionaryAction.triggered.connect(self.onDictionary)  
  
 def onNewWordAction(self):  
 self.newWord = ChooseWord(self)  
 self.setCentralWidget(self.newWord)  
 self.newWord.show()  
  
 def onDictionary(self):  
 self.dictionary = DictionaryWindow(self.con, self.id, self)  
 self.setCentralWidget(self.dictionary)  
 self.dictionary.show()  
  
 def onKeywordsAction(self):  
 print('do')  
 self.keywordstesting = KeywordsTestWindow(self.con, self.id, self)  
 self.setCentralWidget(self.keywordstesting)  
 self.keywordstesting.show()  
 print('posle')  
  
 def onDevelopAction(self):  
 self.developtesting = DevelopTestWindow(self.con, self.id, self)  
 self.setCentralWidget(self.developtesting)  
 self.developtesting.show()  
  
 def onCompTesting(self):  
 self.comptesting = ComputerTestWindow(self.con, self.id, self)  
 self.setCentralWidget(self.comptesting)  
 self.comptesting.show()  
  
 def onTermsTesting(self):  
 self.termstesting = TermsTestWindow(self.con, self.id, self)  
 self.setCentralWidget(self.termstesting)  
 self.termstesting.show()  
  
 def onKeywordsThAction(self):  
 self.keywordswindow = KeywordsWindow(self.con, self.id, self)  
 self.setCentralWidget(self.keywordswindow)  
 self.keywordswindow.show()  
  
 def onDevelopThAction(self):  
 self.developwindow = DevelopWindow(self.con, self.id, self)  
 self.setCentralWidget(self.developwindow)  
 self.developwindow.show()  
  
 def onCompTheory(self):  
 self.computerwindow = ComputerWindow(self.con, self.id, self)  
 self.setCentralWidget(self.computerwindow)  
 self.computerwindow.show()  
  
 def onTermsTheory(self):  
 self.termswindow = TermsWindow(self.con, self.id, self)  
 self.setCentralWidget(self.termswindow)  
 self.termswindow.show()  
   
 def onMainAction(self):  
 self.mainpage = MainPage(self.con, self.id, self)  
 self.setCentralWidget(self.mainpage)  
 self.mainpage.show()  
  
*#окно со словарем*class DictionaryWindow(QWidget):  
 def \_\_init\_\_(self, connection, userId, testWindow):  
 super().\_\_init\_\_(testWindow)  
 uic.loadUi('dictionary.ui', self)  
   
 self.con = connection  
 cur = self.con.cursor()  
 self.setStyleSheet('.QWidget {background-image: url(8b5504825ea01ea1eb747632b8428926.png);}')  
 *#слово по клику* self.wordList.currentTextChanged.connect(self.selectionChanged)  
 self.refresh()  
 *#обновляем словарь* def refresh(self):  
  
 self.wordList.clear()  
 self.words = []  
 self.disc = []  
 self.lst = []  
 self.discWord.setText('')  
 *#составляем список слов* with open('developing.csv', encoding='windows-1251') as csvf:  
 reader = csv.reader(csvf, delimiter=';', quotechar='"')  
 for x in reader:  
 self.words.append(x[0])  
 self.disc.append(x[1])  
   
 with open('keywords.csv', encoding='windows-1251') as csvf:  
 reader = csv.reader(csvf, delimiter=';', quotechar='"')  
 for x in reader:  
 self.words.append(x[0])  
 self.disc.append(x[1])  
 try:   
 with open('termsdisc.csv', encoding='windows-1251') as csvf:  
 reader = csv.reader(csvf, delimiter=';', quotechar='"')  
 for x in reader:  
 self.words.append(x[0])  
 self.disc.append(x[1])  
 *#изначальный список слов* for i in range(len(self.words)):  
 self.lst.append(self.words[i])  
 *#слова в алфавитном прядке* self.words = sorted(self.words)  
 *#выводим слова на экран* for i in range(len(self.words)):  
 self.wordList.addItem(self.words[i])   
 except Exception as error:  
 print(error)  
  
 *#выводим значение слова* def selectionChanged(self, item):  
 if item:  
 ind = self.lst.index(item)  
 self.discWord.setText(self.disc[ind])  
  
*#добавление слова*class ChooseWord(QWidget):  
 def \_\_init\_\_(self, testWindow):  
 super().\_\_init\_\_(testWindow)  
 self.testWindow = testWindow  
 uic.loadUi('chooseword.ui', self)  
 self.setStyleSheet('.QWidget {background-image: url(8b5504825ea01ea1eb747632b8428926.png);}')  
   
 self.addword.clicked.connect(self.addWord)  
  
  
 def addWord(self):  
 newWord = self.word.text()  
 meaning = self.meaning.text()  
 if newWord != '' and meaning != '':  
 typeWord = self.comboBox.currentText()  
 *#выбор типа слова* if typeWord == 'Ключевые слова':  
 with open('keywords.csv', mode='a', encoding='windows-1251') as csvf:  
 s = f'{newWord};{meaning}\n'  
 csvf.writelines(s)  
 if typeWord == 'Слова для разработки':  
 with open('developing.csv', mode='a', encoding='windows-1251') as csvf:  
 s = f'{newWord};{meaning}\n'  
 csvf.writelines(s)  
 if typeWord == 'Back-end разработка':  
 with open('termsdisc.csv', mode='a', encoding='windows-1251') as csvf:  
 s = f'{newWord};{meaning}\n'  
 csvf.writelines(s)  
 self.testWindow.onDictionary()  
  
 else:  
 QMessageBox.critical(self, 'Ошибка', 'Неверный ввод')  
  
  
   
class MainPage(QWidget):  
 def \_\_init\_\_(self, connection, userId, testWindow):  
 super().\_\_init\_\_(testWindow)  
 uic.loadUi('lessons.ui', self)  
 self.con = connection  
 cur = self.con.cursor()  
 self.setStyleSheet('.QWidget {background-image: url(8b5504825ea01ea1eb747632b8428926.png);}')  
 *# айди пользователя* self.id = userId  
 self.refresh()  
 self.picture.setStyleSheet('background-image: url(english-for-it-specialists-1.png);')  
 self.picture\_2.setStyleSheet('background-image: url(pngwing.com.png);')  
 try:  
 name = cur.execute('''SELECT login FROM users  
 WHERE id = ?''', (self.id,)).fetchall()[0][0]  
 print(name)  
 self.login.setText(name)  
 except Exception as error:  
 print(error)  
  
   
 *# показываем обновленные данные пользователей* def refresh(self):  
 cur = self.con.cursor()  
 kc = 0  
 for row in open("keywords.csv", encoding='windows-1251'):  
 kc += 1  
 dc = 0  
 for row in open("developing.csv", encoding='windows-1251'):  
 dc += 1  
 keywordsPointsData = cur.execute('''SELECT points FROM test\_results  
 WHERE user\_id = ? and lesson = ? ORDER BY column DESC LIMIT 1''', (self.id, 'keywords')).fetchall()  
 keywordsPoints = 0  
 if len(keywordsPointsData) > 0:  
 keywordsPoints = keywordsPointsData[0][0]  
 self.keywordsPoints.setText(f'{keywordsPoints // 10} / {kc}')  
  
  
 developPointsData = cur.execute('''SELECT points FROM test\_results  
 WHERE user\_id = ? and lesson = ? ORDER BY column DESC LIMIT 1''', (self.id, 'develop')).fetchall()  
 developPoints = 0  
 if len(developPointsData) > 0:  
 developPoints = developPointsData[0][0]  
 self.developPoints.setText(f'{developPoints // 10} / {dc}')  
  
 compPointsData = cur.execute('''SELECT points FROM test\_results  
 WHERE user\_id = ? and lesson = ? ORDER BY column DESC LIMIT 1''', (self.id, 'computer')).fetchall()  
 compPoints = 0  
 if len(compPointsData) > 0:  
 compPoints = compPointsData[0][0]  
 self.compPoints.setText(f'{compPoints // 10} / 8')  
  
 termsPointsData = cur.execute('''SELECT points FROM test\_results  
 WHERE user\_id = ? and lesson = ? ORDER BY column DESC LIMIT 1''', (self.id, 'terms')).fetchall()  
 termsPoints = 0  
 if len(termsPointsData) > 0:  
 termsPoints = termsPointsData[0][0]  
 self.termsPoint.setText(f'{termsPoints // 10} / 9')  
  
*#цвета кнопок для тестирования*greenButton = '''  
QPushButton{background: rgba(0, 170, 127, 0.8);  
border-radius: 20px;  
border-style: outset;  
 border-width: 5px;  
border-color:rgb(191, 191, 191);  
color:rgb(57, 57, 57);}  
'''  
redButton = '''  
QPushButton{background: rgba(197, 52, 23, 0.8);  
border-radius: 20px;  
border-style: outset;  
 border-width: 5px;  
border-color:rgb(191, 191, 191);  
color:rgb(57, 57, 57);}  
'''  
greyButton = '''  
QPushButton{background: rgba(255, 255, 255, 0.6);  
border-radius: 20px;  
border-style: outset;  
 border-width: 5px;  
border-color:rgb(191, 191, 191);  
color:rgb(57, 57, 57);}  
'''  
  
  
*# окно тестов с ключевыми словами*class KeywordsTestWindow(QWidget):  
 global greenButton  
 global redButton  
 global greyButton  
 def \_\_init\_\_(self, connection, userId, testWindow):  
 super().\_\_init\_\_(testWindow)  
 uic.loadUi('keywordsquestions.ui', self)  
 self.con = connection  
 self.id = userId  
 self.testWindow = testWindow  
 self.setStyleSheet('.QWidget {background-image: url(8b5504825ea01ea1eb747632b8428926.png);}')  
  
 *# cоздаем список слов и их объяснений* with open('keywords.csv', encoding='windows-1251') as csvf:  
 reader = csv.reader(csvf, delimiter=';', quotechar='"')  
 self.words = []  
 self.disc = []  
 for x in reader:  
 self.words.append(x[0])  
 self.disc.append(x[1])  
 self.continueButton.clicked.connect(self.chooseWord)  
 *# кнопки ответов* self.anButton1.clicked.connect(self.firstButtonCheck)  
 self.anButton2.clicked.connect(self.secondButtonCheck)  
 self.anButton3.clicked.connect(self.thirdButtonCheck)  
 self.reset()  
 self.chooseWord()  
  
 *# сбрасываем переменные чтобы заново проходить тест* def reset(self):  
 *# правильный ответ* self.rightAnswer = ''  
 self.answerCount = 0  
 *# индексы слов для тестов* self.left = [int(i) for i in range(0, len(self.words))]  
  
 *# кнопка 1* def firstButtonCheck(self):  
 if self.disc1.text() == self.rightAnswer:  
 self.anButton1.setStyleSheet(greenButton)  
 self.answerCount += 1  
  
 else:  
 self.anButton1.setStyleSheet(redButton)  
 if self.disc2.text() == self.rightAnswer:  
 self.anButton2.setStyleSheet(greenButton)  
 elif self.disc3.text() == self.rightAnswer:  
 self.anButton3.setStyleSheet(greenButton)  
  
 *# кнопка 2* def secondButtonCheck(self):  
 if self.disc2.text() == self.rightAnswer:  
 self.anButton2.setStyleSheet(greenButton)  
 self.answerCount += 1  
  
 else:  
 self.anButton2.setStyleSheet(redButton)  
 if self.disc1.text() == self.rightAnswer:  
 self.anButton1.setStyleSheet(greenButton)  
 elif self.disc3.text() == self.rightAnswer:  
 self.anButton3.setStyleSheet(greenButton)  
  
 *# кнопка 3* def thirdButtonCheck(self):  
 *# проверяем на правильность* if self.disc3.text() == self.rightAnswer:  
 self.anButton3.setStyleSheet(greenButton)  
 self.answerCount += 1  
 else:  
 *# отмечаем что неправильно и отмечаем правильное* self.anButton3.setStyleSheet(redButton)  
 if self.disc2.text() == self.rightAnswer:  
 self.anButton2.setStyleSheet(greenButton)  
 elif self.disc1.text() == self.rightAnswer:  
 self.anButton1.setStyleSheet(greenButton)  
  
 *# рандомно выбираем неповторяющиеся индексы* def randAns(self, a, b, lst):  
 while True:  
 n = randint(a, b)  
 if n not in lst:  
 return n  
  
 def chooseWord(self):  
 *# сбрасываем цвет кнопок* button\_style = greyButton  
 self.anButton1.setStyleSheet(button\_style)  
 self.anButton2.setStyleSheet(button\_style)  
 self.anButton3.setStyleSheet(button\_style)  
 *# проверяем, что есть еще слова* try:  
 if len(self.left) > 0:  
 variants = []  
 *# индекс правильного слова с правильным ответом и двух рандомных ответов* ind = choice(self.left)  
 self.left.remove(ind)  
 variants.append(ind)  
  
 ans1 = self.randAns(0, len(self.words)-1, variants)  
 variants.append(ans1)  
  
 ans2 = self.randAns(0, len(self.words)-1, variants)  
  
 self.word.setText(self.words[ind])  
 rightAn = self.disc[ind]  
 self.rightAnswer = rightAn  
 *# список вариантов ответов* answers = [rightAn, self.disc[ans1], self.disc[ans2]]  
 choice1 = choice(answers)  
 *# вписываем варианты ответов* self.disc1.setText(choice1)  
 answers.remove(choice1)  
 choice2 = choice(answers)  
 self.disc2.setText(choice2)  
 answers.remove(choice2)  
 self.disc3.setText(answers[0])  
 else:  
 *# когда тест закончился* points = 10 \* self.answerCount  
 cur = self.con.cursor()  
 cur.execute('''INSERT INTO test\_results(user\_id, lesson, points)  
 VALUES(?, "keywords", ?)''', (self.id, points,))  
 self.con.commit()  
 kc = 0  
 for row in open("keywords.csv", encoding='windows-1251'):  
 kc += 1  
 QMessageBox.information(self, 'Тест пройден', f"Правильных ответов: {points // 10} из {kc}")  
 self.testWindow.onMainAction()  
  
 except Exception as error:  
 print(error)  
  
*#окно теста со словами для разработки*class DevelopTestWindow(QWidget):  
 global greenButton  
 global redButton  
 global greyButton  
 def \_\_init\_\_(self, connection, userId, testWindow):  
 super().\_\_init\_\_(testWindow)  
 uic.loadUi('developquestions.ui', self)  
 self.con = connection  
 self.id = userId  
 self.testWindow = testWindow  
 cur = self.con.cursor()  
 self.setStyleSheet('.QWidget {background-image: url(8b5504825ea01ea1eb747632b8428926.png);}')  
 with open('developing.csv', encoding='windows-1251') as csvf:  
 reader = csv.reader(csvf, delimiter=';', quotechar='"')  
 self.words = []  
 self.disc = []  
 for x in reader:  
 self.words.append(x[0])  
 self.disc.append(x[1])  
 self.continueButton.clicked.connect(self.chooseWord)  
  
 *# кнопки ответов* self.anButton1.clicked.connect(self.firstButtonCheck)  
 self.anButton2.clicked.connect(self.secondButtonCheck)  
 self.anButton3.clicked.connect(self.thirdButtonCheck)  
 self.reset()  
 self.chooseWord()  
  
 *# сбрасываем переменные чтобы заново проходить тест* def reset(self):  
 *# правильный ответ* self.rightAnswer = ''  
 self.answerCount = 0  
 *# индексы слов для тестов* self.left = [int(i) for i in range(0, len(self.words))]  
  
 *# кнопка 1* def firstButtonCheck(self):  
 if self.disc1.text() == self.rightAnswer:  
 self.anButton1.setStyleSheet(greenButton)  
 self.answerCount += 1  
 else:  
 self.anButton1.setStyleSheet(redButton)  
 if self.disc2.text() == self.rightAnswer:  
 self.anButton2.setStyleSheet(greenButton)  
 elif self.disc3.text() == self.rightAnswer:  
 self.anButton3.setStyleSheet(greenButton)  
  
 *# кнопка 2* def secondButtonCheck(self):  
 if self.disc2.text() == self.rightAnswer:  
 self.anButton2.setStyleSheet(greenButton)  
 self.answerCount += 1  
 else:  
 self.anButton2.setStyleSheet(redButton)  
 if self.disc1.text() == self.rightAnswer:  
 self.anButton1.setStyleSheet(greenButton)  
 elif self.disc3.text() == self.rightAnswer:  
 self.anButton3.setStyleSheet(greenButton)  
  
 *# кнопка 3* def thirdButtonCheck(self):  
 *# проверяем на правильность* if self.disc3.text() == self.rightAnswer:  
 self.anButton3.setStyleSheet(greenButton)  
 self.answerCount += 1  
 else:  
 *# отмечаем что неправильно и отмечаем правильное* self.anButton3.setStyleSheet(redButton)  
 if self.disc2.text() == self.rightAnswer:  
 self.anButton2.setStyleSheet(greenButton)  
 elif self.disc1.text() == self.rightAnswer:  
 self.anButton1.setStyleSheet(greenButton)  
 *# рандомно выбираем неповторяющиеся индексы* def randAns(self, a, b, lst):  
 while True:  
 n = randint(a, b)  
 if n not in lst:  
 return n  
  
 def chooseWord(self):  
 button\_style = greyButton  
 self.anButton1.setStyleSheet(button\_style)  
 self.anButton2.setStyleSheet(button\_style)  
 self.anButton3.setStyleSheet(button\_style)  
 *# проверяем, что есть еще слова* try:  
 if len(self.left) > 0:  
 variants = []  
 *# индекс правильного слова с правильным ответом и двух рандомных ответов* ind = choice(self.left)  
 self.left.remove(ind)  
 variants.append(ind)  
  
 ans1 = self.randAns(0, len(self.words) - 1, variants)  
 variants.append(ans1)  
  
 ans2 = self.randAns(0, len(self.words) - 1, variants)  
  
 self.word.setText(self.words[ind])  
 rightAn = self.disc[ind]  
 self.rightAnswer = rightAn  
 *# список вариантов ответов* answers = [rightAn, self.disc[ans1], self.disc[ans2]]  
 choice1 = choice(answers)  
 self.disc1.setText(choice1)  
 answers.remove(choice1)  
 choice2 = choice(answers)  
 self.disc2.setText(choice2)  
 answers.remove(choice2)  
 self.disc3.setText(answers[0])  
 else:  
 *# когда тест закончился* points = 10 \* self.answerCount  
 cur = self.con.cursor()  
 *# вставляем результат в таблицу* cur.execute('''INSERT INTO test\_results(user\_id, lesson, points)  
 VALUES(?, "develop", ?)''', (self.id, points,))  
 self.con.commit()  
 dc = 0  
 for row in open("developing.csv", encoding='windows-1251'):  
 dc += 1  
 QMessageBox.information(self, 'Тест пройден', f"Правильных ответов: {points // 10} из {dc}")  
 self.testWindow.onMainAction()  
 except Exception as error:  
 print(error)  
  
*#тест с компьютером*class ComputerTestWindow(QWidget):  
 def \_\_init\_\_(self, connection, userId, testWindow):  
 super().\_\_init\_\_(testWindow)  
 uic.loadUi('comptest.ui', self)  
 self.con = connection  
 self.id = userId  
 cur = self.con.cursor()  
 self.setStyleSheet('.QWidget {background-image: url(8b5504825ea01ea1eb747632b8428926.png);}')  
 self.checkAnswer.clicked.connect(self.check)  
 *#поля ответом* self.elements = [self.answer1, self.answer2, self.answer3, self.answer4, self.answer5, self.answer6,  
 self.answer7, self.answer8, self.answer9, self.answer10]  
  
 self.cpu.setStyleSheet('background-image: url(cpu.png);')  
 self.cpucooler.setStyleSheet('background-image: url(cpucooler.png);')  
 self.graphiccard.setStyleSheet('background-image: url(graphiccard.png);')  
 self.hdd.setStyleSheet('background-image: url(hdd.png);')  
 self.motherboard.setStyleSheet('background-image: url(motherboard.png);')  
 self.powersupply.setStyleSheet('background-image: url(powersupply.png);')  
 self.ram.setStyleSheet('background-image: url(ram.png);')  
 self.soundcard.setStyleSheet('background-image: url(soundcard.png);')  
 self.ssd.setStyleSheet('background-image: url(ssd.png);')  
 self.ports.setStyleSheet('background-image: url(ports.png);')  
  
 self.tryAgain.clicked.connect(self.again)  
 *#ачать заново* def again(self):  
 self.points.setText(' ')  
 style = '''color:rgb(57, 57, 57);  
 background-color: rgba(221, 221, 221, 0.1)  
 '''  
 for i in range(0, 10):  
 self.elements[i].setStyleSheet(style)  
 self.elements[i].setText('')  
 *#проверка правильности* def check(self):  
 try:  
 k = 0  
 words = ['cpu', 'cpu cooler', 'Graphics card', 'hdd', 'motherboard',  
 'power supply', 'ram', 'sound card', 'ssd', 'ports']  
 for i in range(0, 10):  
 if self.elements[i].text().lower() == words[i]:  
 k+= 1  
 self.elements[i].setStyleSheet('background: rgb(170, 255, 127);')  
 self.elements[i].setStyleSheet('background: rgb(170, 255, 127);')  
 else:  
 self.elements[i].setStyleSheet('background: rgb(255, 73, 73);')  
 points = 10 \* k  
 cur = self.con.cursor()  
 *# вставляем результат в таблицу* cur.execute('''INSERT INTO test\_results(user\_id, lesson, points)  
 VALUES(?, "computer", ?)''', (self.id, points,))  
 self.con.commit()  
 self.points.setText(f'Правильных ответов: {points // 10}')  
   
   
 except Exception as error:  
 print(error)  
  
*#тест вставляем слово*class TermsTestWindow(QWidget):  
 global greenButton  
 global redButton  
 global greyButton  
 def \_\_init\_\_(self, connection, userId, testWindow):  
 super().\_\_init\_\_()  
 uic.loadUi('termstest.ui', self)  
 self.con = connection  
 self.id = userId  
 cur = self.con.cursor()  
 self.testWindow = testWindow  
 self.setStyleSheet('.QWidget {background-image: url(8b5504825ea01ea1eb747632b8428926.png);}')  
 with open('terms.csv', encoding='windows-1251') as csvf:  
 reader = csv.reader(csvf, delimiter=';', quotechar='"')  
 self.words = []  
 self.sentence = []  
 for x in reader:  
 self.words.append(x[0])  
 self.sentence.append(x[1])  
 self.continueButton.clicked.connect(self.chooseWord)  
  
 *# кнопки ответов* self.anButton1.clicked.connect(self.firstButtonCheck)  
 self.anButton2.clicked.connect(self.secondButtonCheck)  
 self.anButton3.clicked.connect(self.thirdButtonCheck)  
  
 self.reset()  
 self.chooseWord()  
  
 *# сбрасываем переменные чтобы заново проходить тест* def reset(self):  
 *# правильный ответ* self.rightAnswer = ''  
 self.answerCount = 0  
 *# индексы слов для тестов* self.left = [int(i) for i in range(0, len(self.sentence) - 1)]  
  
  
 *# кнопка 1* def firstButtonCheck(self):  
 if self.disc1.text() == self.rightAnswer:  
 self.anButton1.setStyleSheet(greenButton)  
 self.answerCount += 1  
 else:  
 self.anButton1.setStyleSheet(redButton)  
 if self.disc2.text() == self.rightAnswer:  
 self.anButton2.setStyleSheet(greenButton)  
 elif self.disc3.text() == self.rightAnswer:  
 self.anButton3.setStyleSheet(greenButton)  
  
 *# кнопка 2* def secondButtonCheck(self):  
 if self.disc2.text() == self.rightAnswer:  
 self.anButton2.setStyleSheet(greenButton)  
 self.answerCount += 1  
 else:  
 self.anButton2.setStyleSheet(redButton)  
 if self.disc1.text() == self.rightAnswer:  
 self.anButton1.setStyleSheet(greenButton)  
 elif self.disc3.text() == self.rightAnswer:  
 self.anButton3.setStyleSheet(greenButton)  
  
 *# кнопка 3* def thirdButtonCheck(self):  
 *# проверяем на правильность* if self.disc3.text() == self.rightAnswer:  
 self.anButton3.setStyleSheet(greenButton)  
 self.answerCount += 1  
 else:  
 *# отмечаем что неправильно и отмечаем правильное* self.anButton3.setStyleSheet(redButton)  
 if self.disc2.text() == self.rightAnswer:  
 self.anButton2.setStyleSheet(greenButton)  
 elif self.disc1.text() == self.rightAnswer:  
 self.anButton1.setStyleSheet(greenButton)  
 *# рандомно выбираем неповторяющиеся индексы* def randAns(self, a, b, lst):  
 while True:  
 n = randint(a, b)  
 if n not in lst:  
 return n  
  
 def chooseWord(self):  
 button\_style = greyButton  
 self.anButton1.setStyleSheet(button\_style)  
 self.anButton2.setStyleSheet(button\_style)  
 self.anButton3.setStyleSheet(button\_style)  
 *# проверяем, что есть еще слова* try:  
 if len(self.left) > 0:  
 variants = []  
 *# индекс правильного слова с правильным ответом и двух рандомных ответов* ind = choice(self.left)  
 self.left.remove(ind)  
 variants.append(ind)  
  
 ans1 = self.randAns(0, len(self.sentence) - 1, variants)  
 variants.append(ans1)  
  
 ans2 = self.randAns(0, len(self.sentence) - 1, variants)  
  
 self.word.setText(self.sentence[ind])  
 rightAn = self.words[ind]  
 self.rightAnswer = rightAn  
 *# список вариантов ответов* answers = [rightAn, self.words[ans1], self.words[ans2]]  
 choice1 = choice(answers)  
 self.disc1.setText(choice1)  
 answers.remove(choice1)  
 choice2 = choice(answers)  
 self.disc2.setText(choice2)  
 answers.remove(choice2)  
 self.disc3.setText(answers[0])  
 else:  
 *# когда тест закончился* points = 10 \* self.answerCount  
 cur = self.con.cursor()  
 *# вставляем результат в таблицу* cur.execute('''INSERT INTO test\_results(user\_id, lesson, points)  
 VALUES(?, "terms", ?)''', (self.id, points,))  
 self.con.commit()  
 QMessageBox.information(self, 'Тест пройден', f"Правильных ответов: {points // 10} из 9")  
 self.testWindow.onMainAction()  
 except Exception as error:  
 print(error)  
   
*##############################################  
  
# окно урока с ключевыми словами*class KeywordsWindow(QWidget):  
 def \_\_init\_\_(self, connection, userId, theoryWindow):  
 super().\_\_init\_\_()  
 uic.loadUi('keywords.ui', self)  
 self.con = connection  
 self.id = userId  
 self.setStyleSheet('.QWidget {background-image: url(8b5504825ea01ea1eb747632b8428926.png);}')  
 cur = self.con.cursor()  
  
 *#слово по клику* try:  
 self.wordList.currentTextChanged.connect(self.selectionChanged)  
 self.refresh()  
 except Exception as error:  
 print(error)  
  
  
 def refresh(self):  
 self.wordList.clear()  
 self.words = []  
 self.disc = []  
 self.lst = []  
 self.answer.setText('')  
 *#составляем список слов* with open('keywords.csv', encoding='windows-1251') as csvf:  
 reader = csv.reader(csvf, delimiter=';', quotechar='"')  
 for x in reader:  
 self.words.append(x[0])  
 self.disc.append(x[1])  
 for i in range(len(self.words)):  
 self.lst.append(self.words[i])  
 *#слова в алфавитном прядке* self.words = sorted(self.words)  
 *#выводим слова на экран* for i in range(len(self.words)):  
 self.wordList.addItem(self.words[i])  
  
  
 def selectionChanged(self, item):  
 if item:  
 ind = self.lst.index(item)  
 self.answer.setText(self.disc[ind])  
  
  
  
  
  
*# окно урока со словами для разработки*class DevelopWindow(QWidget):  
 def \_\_init\_\_(self, connection, userId, theoryWindow):  
 super().\_\_init\_\_()  
 uic.loadUi('develop.ui', self)  
 self.con = connection  
 self.id = userId  
 self.setStyleSheet('.QWidget {background-image: url(8b5504825ea01ea1eb747632b8428926.png);}')  
 cur = self.con.cursor()  
 *#слово по клику* try:  
 self.wordList.currentTextChanged.connect(self.selectionChanged)  
 self.refresh()  
 except Exception as error:  
 print(error)  
  
  
 def refresh(self):  
 self.wordList.clear()  
 self.words = []  
 self.disc = []  
 self.lst = []  
 self.answer.setText('')  
 *#составляем список слов* with open('developing.csv', encoding='windows-1251') as csvf:  
 reader = csv.reader(csvf, delimiter=';', quotechar='"')  
 for x in reader:  
 self.words.append(x[0])  
 self.disc.append(x[1])  
 for i in range(len(self.words)):  
 self.lst.append(self.words[i])  
 *#слова в алфавитном прядке* self.words = sorted(self.words)  
 *#выводим слова на экран* for i in range(len(self.words)):  
 self.wordList.addItem(self.words[i])  
  
 def selectionChanged(self, item):  
 if item:  
 ind = self.lst.index(item)  
 self.answer.setText(self.disc[ind])  
  
   
class ComputerWindow(QWidget):  
 def \_\_init\_\_(self, connection, userId, theoryWindow):  
 super().\_\_init\_\_()  
 uic.loadUi('comp.ui', self)  
 self.con = connection  
 self.id = userId  
 self.setStyleSheet('.QWidget {background-image: url(8b5504825ea01ea1eb747632b8428926.png);}')  
 self.theoryWindow = theoryWindow  
  
 self.cpu.setStyleSheet('background-image: url(cpu.png);')  
 self.cpucooler.setStyleSheet('background-image: url(cpucooler.png);')  
 self.graphiccard.setStyleSheet('background-image: url(graphiccard.png);')  
 self.hdd.setStyleSheet('background-image: url(hdd.png);')  
 self.motherboard.setStyleSheet('background-image: url(motherboard.png);')  
 self.powersupply.setStyleSheet('background-image: url(powersupply.png);')  
 self.ram.setStyleSheet('background-image: url(ram.png);')  
 self.soundcard.setStyleSheet('background-image: url(soundcard.png);')  
 self.ssd.setStyleSheet('background-image: url(ssd.png);')  
 self.ports.setStyleSheet('background-image: url(ports.png);')  
  
  
class TermsWindow(QWidget):  
 def \_\_init\_\_(self, connection, userId, theoryWindow):  
 super().\_\_init\_\_()  
 uic.loadUi('terms.ui', self)  
 self.con = connection  
 self.id = userId  
 self.setStyleSheet('.QWidget {background-image: url(8b5504825ea01ea1eb747632b8428926.png);}')  
 cur = self.con.cursor()  
  
 self.setStyleSheet('.QWidget {background-image: url(8b5504825ea01ea1eb747632b8428926.png);}')  
 *#слово по клику* try:  
 self.wordList.currentTextChanged.connect(self.selectionChanged)  
 self.refresh()  
 except Exception as error:  
 print(error)  
  
 def refresh(self):  
 self.wordList.clear()  
 self.words = []  
 self.disc = []  
 self.lst = []  
 self.answer.setText('')  
 *#составляем список слов* with open('termsdisc.csv', encoding='windows-1251') as csvf:  
 reader = csv.reader(csvf, delimiter=';', quotechar='"')  
 for x in reader:  
 self.words.append(x[0])  
 self.disc.append(x[1])  
 for i in range(len(self.words)):  
 self.lst.append(self.words[i])  
 *#слова в алфавитном прядке* self.words = sorted(self.words)  
 *#выводим слова на экран* for i in range(len(self.words)):  
 self.wordList.addItem(self.words[i])  
  
 def selectionChanged(self, item):  
 if item:  
 ind = self.lst.index(item)  
 self.answer.setText(self.disc[ind])  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app = QApplication(sys.argv)  
 ex = OpenWindow()  
 sys.exit(app.exec\_())