**РОСЖЕЛДОР**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**

**ВЫСШЕГО ОБРАЗОВАНИЯ**

**СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**ПУТЕЙ СООБЩЕНИЯ (СГУПС)**

|  |  |  |  |
| --- | --- | --- | --- |
| **К защите:** |  |  | |
| **Заведующий кафедрой** | **Информационные** | |
| **технологии транспорта** | | |
|  | д-р техн. наук, проф. | |
|  |  | В. И. Хабаров | |
| *подпись* |  | *инициалы, фамилия* | |
|  |  |  | |
| *дата* |  |  | |

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Тема:** | Разработка прототипа Телеграм бота «Помощник по | | | | | |
|  | изучению английского языка» | | | | | |
|  | |  | БР.БИСТ.11.2022 |  |  |
|  | |  | *шифр документа* |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | **Выполнил** |  |  |  | **Руководитель** |
|  |  | С. Б. Листунов |  |  |  | канд. техн. наук, доц.  С. П. Сарычев |
| *подпись* |  | *инициалы, фамилия* |  | *подпись* |  | *инициалы, фамилия* |
|  |  |  |  |  |  |  |
| *дата* |  |  |  | *дата* |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Консультанты по разделам** |  |  |  |  |
| 1 |  |  |  |  |
|  |  |  |  |  |
| 2 |  |  |  |  |
|  |  |  |  |  |
| 3 |  |  |  |  |
|  |  |  |  |  |
| Нормоконтролер работы |  |  |  | ст. преп.  Т. А. Распопина |
|  |  | *подпись* |  |  |
|  |  |  |  |  |
|  |  | *дата* |  |  |

**2022 г.**

**СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**ПУТЕЙ СООБЩЕНИЯ (СГУПС)**

Факультет: Бизнес-информатики

Кафедра: Информационные технологии транспорта

Направление: 09.03.02 «Информационные системы и технологии»

Профиль: Интеллектуальные транспортные системы

|  |  |
| --- | --- |
|  | ***УТВЕРЖДАЮ****: зав. кафедрой «Информационные технологии транспорта»*  д-р техн. наук, проф.  В. И. Хабаров |
|  | *«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_ 2022 г.* |

**З А Д А Н И Е**

**на выполнение выпускной квалификационной работы**

|  |  |  |  |
| --- | --- | --- | --- |
| студенту | Листунову Сергею Борисовичу | | |
|  |  | | |
| 1. Тема «Разработка прототипа Телеграм бота «Помощник по изучению английского языка» утверждена приказом № 203/с от «30» мая 2022 г. | | | |
| 2. Задание выдано «12» мая 2022 г. | | | |
| 3. Срок сдачи законченной работы на кафедру «17» июня 2022 г. | | | |
| 4. Исходные данные: данные, полученные в ходе прохождения преддипломной практики | | | |
| 5. Содержание расчетно-пояснительной записки | | | |
| Наименование разделов и вопросов | | Примерное количество страниц | График (сроки) выполнения |
| Введение | | 3 | 13.05.2022 |
| Аналитическое исследование | | 4 | 13.05.2022 |
| Проектирование Telegram-бота | | 13 | 15.05.2022 |
| Создание программы | | 16 | 28.05.2022 |
| Заключение | | 1 | 01.06.2022 |

6. Содержание и объемы графической части

|  |  |  |
| --- | --- | --- |
| Наименование графического документа (чертежа, схемы, графика) | Количество  листов  формата А1 | График  (сроки)  выполнения |
| Презентация PowerPoint | 15 | 05.05.2022 |

7. Консультанты по разделам

|  |  |  |
| --- | --- | --- |
| Наименование  раздела | Фамилия, И. О.  консультанта | Подпись консультанта,  дата выдачи задания |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| Руководитель |  | С. П. Сарычев |
|  | *(подпись, фамилия, И.О.)* |  |
| Задание к использованию принял |  | С. Б. Листунов |
|  | *(подпись студента)* |  |

УДК 004.588

**АННОТАЦИЯ**

В работе 45 страниц, 35 рисунков, 17 таблиц, 15 источников, 2 приложения.

Ключевые слова: *Telegram-бот, Английский язык, обучение, помощник в обучении*.

Предметная область – Telegram-бот. Цель работы – создать прототип Telegram-бота являющимся помощником в изучении Английского языка.

Во время выполнения работы проведен анализ предметной области, описаны бизнес-процессы, обоснован выбор языка и программного обеспечения, рассмотрено поэтапное проектирование программы.

Результатом выполнения работы является программа, разработанная в среде PyCharm 2022.1, на языке Python, с использованием компактной встраиваемой СУБД SQLite.

**ABSTRACT**

The work contains 45 pages, 35 figures, 17 tables, 15 sources, 2 appendices.

Keywords: *Telegram bot, English language, learning, learning assistant.*

Subject area – Telegram bot. The purpose of the work is to create a prototype of a Telegram bot that is an assistant in learning the English language.

During the execution of the work, an analysis of the subject area was carried out, business processes were described, the choice of language and software was justified, and a phased design of the program was considered.

The result of the work is a program developed in the PyCharm 2022.1 environment, in Python, using a compact embedded SQLite DBMS.

**ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ**

БД – это поименованная собрание данных, которое отражает состояние объектов и их отношений в рассматриваемой предметной области.

ИС – информационная система, это система, которая предназначена для хранения, обработки и поиска информации, а также персонал, который обеспечивает и распространяет информацию.

Мессенджер – программа по обмену мгновенными текстовыми и голосовыми сообщениями, осуществлению телефонных звонков и ведению разговоров по видеосвязи.

ПО – программное обеспечение, это программа или множество программ, которые используются для управления компьютером.

Чат-бот – виртуальная программа-собеседник, которая выясняет потребности пользователя, а затем удовлетворяет их посредством ведения диалога с клиентами на естественном языке.

IDE – Integrated Development Environment, это система программных средств, используемая программистами для разработки программного обеспечения.

**СОДЕРЖАНИЕ**

[Введение 6](#_Toc106215662)

[1 Аналитическое исследование 9](#_Toc106215663)

[1.1 Определение предметной области 9](#_Toc106215664)

[1.2 Определение проблемы предметной области 9](#_Toc106215665)

[1.3 Решение проблемы 10](#_Toc106215666)

[1.4 Постановка задачи 10](#_Toc106215667)

[1.5 Вывод об аналитическом исследовании 11](#_Toc106215668)

[2 Проектирование Telegram-бота 12](#_Toc106215669)

[2.1 Моделирование бизнес-процессов 12](#_Toc106215670)

[2.2 Обоснование выбора языка программирования 15](#_Toc106215671)

[2.3 Обоснование выбора среды разработки 17](#_Toc106215672)

[2.4 Описание структуры базы данных 23](#_Toc106215673)

[2.5 Вывод о проектировании программы 27](#_Toc106215674)

[3 Разработка программы 28](#_Toc106215675)

[3.1 Разработка логики чат-бота 28](#_Toc106215676)

[3.2 Разработка и интеграция базы данных 29](#_Toc106215677)

[3.3 Вывод по разрабатываемому продукту 34](#_Toc106215678)

[4 Руководство пользователя 35](#_Toc106215679)

[4.1 Подготовка к работе 35](#_Toc106215680)

[4.2 Описание операций 38](#_Toc106215681)

[Заключение 43](#_Toc106215682)

[Список использованных источников 44](#_Toc106215683)

[Приложение А Код функциональной части 46](#_Toc106215684)

[Приложение Б Код базы данных 57](#_Toc106215685)

**ВВЕДЕНИЕ**

В повседневной жизни мы часто пользуемся различными мессенджерами, например: WhatsApp, Viber, Telegram, VK, FaceBook и др. С каждым днём мессенджеры всё больше заполняют нашу повседневную жизнь, ведь они невероятно удобны и стали обычным средством для коммуникации. Всё чаще в мессенджерах появляются новые функции. Одна из таких последних функций является чат бот.

Чат-боты могут выполнять различные функции. Существуют различные разновидности чат-ботов. Их можно разделить по алгоритму, виду и функциональности.

По алгоритму бывают примитивные и саморазвивающиеся чат-боты:

* примитивные, к ним относятся самые простые чат-боты. У них небольшой функционал и заранее подготовленный скрипт, по которому они общаются с пользователями;
* в основе саморазвивающихся чат-ботах лежит нейронная сеть. Благодаря им, бот понимает, что пользователь пишет. Чат-бот имеет возможность отвечать пользователю и его ответы похожи на ответы реального человека.

По виду чат-боты делятся на:

* кнопочные, общение с чат-ботами происходит посредством заготовленных кнопок, с заранее заготовленными ответами;
* текстовые, диалог между чат-ботом и пользователем происходит посредством текста, бот распознаёт ключевые слова и на их основе чат-бот выдаёт ответы.

По функционалу выделяют коммуникационных и функциональных чат-ботов:

* коммуникационные чат-боты берут на себя обязанности консультантов. Их основная задача – это общение с пользователем. В их функционал может входить ответы на вопросы пользователей, помощь в подборе товара или услуги, а также информирование о различных акциях и скидках;
* благодаря функциональным чат-ботам становиться возможным совершать определенные действия, такие как покупка товаров или услуг, проведение оплаты и так далее.

Популярность чат-ботов – это не веяния современных трендов, а удобное программное решение для различных типовых задач, например:

* помочь пользователю сделать заказ;
* принять заявку;
* собрать обратную связь от пользователя;
* дать консультацию;
* собрать контактные данные;
* реализовать различные товары и услуг;
* помочь пользователю записаться на приём;
* скачать музыку, книгу или документы.

И это далеко не весь список возможных функций чат ботов [1].

Очень часто мы можем увидеть таких ботов, как бизнес ботов, которые используются для развития бизнеса. Они помогут вам, если вам понадобился прайс-лист; или же вы хотите узнать информацию о товарах; они сообщат вам о новых поступлениях, акциях, скидках и бонусных программах; они сообщат вам о бонусном балансе; так же они могут принимать заявку на заказ; так же они помогут и ответят на распространённые вопросы.

Ещё один из интересных ботов, это бот записи к врачу. После того как пациент написал в чат, бот предлагает указать город, в котором пациент проживает и комфортное для посещения учреждение. На основании полученных данных пользователю будут предложены специалисты. После введения всей необходимой информации, останется лишь указать удобное время, дату, свои ФИО, дату рождения и номер телефона. Если всё прошло успешно, пациенту будет отправлен электронный талон [2].

Бот же, который будет разрабатываться в этой выпускной квалификационной работе, это обучающий чат-бот. Обучение проводится в мессенджере «Telegram» мобильного телефона. Клиент может начать обучение в любое удобное для него время, просто запустив беседу с «наставником» в чате. Контент адаптирован так, чтобы пользователь в непринуждённой беседе изучал английский язык, проходя упражнения различного уровня сложности.

Для разработки чат-бота, который является помощником по изучению английского языка, нам понадобится:

* проанализировать предметную область;
* разработать и проанализировать список требований к функционалу чат-бота;
* разработать методические материалы, которые будем использовать при обучении;
* разработать базу данных, в которой будет хранится информация о пользователях;
* разработать чат-бота с учетом всех требований и особенностей предметной области;
* протестировать разработанную систему и при необходимости исправить недостатки системы.

В ходе разработки системы будут использованы:

* интегрированная среда разработки программного обеспечения PyCharm 2022.1;
* компактная встраиваемая СУБД SQLite;
* инструмент для моделирования бизнес-процессов Visual Paradigm;
* при необходимости могут быть использованы и другие инструменты.

**1 Аналитическое исследование**

* 1. **Определение предметной области**

Предметной областью выпускной квалификационной работы является чат-бота в мессенджере Telegram. Бот будет использоваться, как помощник для изучения английского языка. Данный бот будет предоставлять учебные материалы для изучения тем по английскому языку, а после изучения материалов пользователь бота проходит задания. Этот Telegram бот может использоваться как дополнительное учебное пособие в школах и университетах. В последствии данного бота можно адаптировать к другим различным мессенджерам.

**1.2 Определение проблемы предметной области**

На данный момент существует большое многообразие различных обучающих программ. Обучающие программы – это программы, предназначенные для самостоятельного изучения материала учащимися. Данные программы должны стимулировать обучающегося на прохождение материла и поощрять его успехи.

В таблице 1.1 приведены самые популярные обучающие программы и их разработчики.

Таблица 1.1 – Обучающие программы

|  |  |
| --- | --- |
| Ресурсы | Издатель |
| 1С: Репетитор. Физика. | 1C |
| 1С: Репетитор. Химия. | 1C |
| 1С: Репетитор. Русский язык. | 1C |
| English Gold 2000 | Мультимедиа технологии и ДО |
| English Reading Club | Мультимедиа технологии и ДО |

Однако, все эти программы являются десктопными и платными, что может являться, некоторым препятствием для их использования. Также нужно предустанавливать программное обеспеченье, они могут не подходить по системным требованиям.

Со всеми вышеперечисленными проблемами пользователь может встретиться при выборе нужного ему ресурса.

**1.3 Решение проблемы**

Определив проблему предметной области, перейдем к ее решению, которое видится в разработке помощника по изучению английского языка, реализованного в форме чат-бота.

Чат-бот (англ. chatbot) – это программа, которая имитирует реальный разговор с пользователем. Чат-боты позволяют общаться с помощью текстовых или аудио сообщений на сайтах, в мессенджерах, мобильных приложениях или по телефону.

Помощник по изучению английского языка – это сервис для смартфонов и персональных компьютеров, предназначение которого состоит в предложении пользователю заданий и обучающих материалов.

Данный сервис включает в себя следующие возможности:

– регистрация в боте;

– возможность выбора тем и сложности заданий;

– возможность проходить тесты по английскому языку;

– получение очков за решения заданий;

– возможность просмотра позиции в таблице рейтинга.

1.4 Постановка задачи

Исходя из предложенного решения проблемы, на данном этапе необходимо установить задачи.

Для достижения поставленной цели необходимо выполнить следующие задачи:

* описать бизнес-процессы Telegram-бота;
* построить концептуальную модель;
* обосновать выбор языка и среды разработки;
* разработать Telegram-бота;
* разработать руководство пользователя.

1.5 Вывод об аналитическом исследовании

Анализ предметной области показал, что разрабатываемая программа должна обеспечить: интерес пользователя к процессу обучения, снижение нагрузки на преподавательский состав, занимающийся обучением учеников, предоставление заданий и правил, относящихся к ним, возможность использования ПО на любой платформе, а также отсутствие монетизации.

2 Проектирование Telegram-бота

2.1 Моделирование бизнес-процессов

Для моделирования бизнес-процессов выбрана методология UML. Unified Modeling Language (UML) – унифицированный язык моделирования: оно подразумевает создание модели, описывающей объект. Подходит для широкого класса проектируемых программных систем, различных областей приложений, типов организаций, уровней компетентности, размеров проектов. Диаграммы UML описывают границы, структуру и поведение как всей системы, так и отдельных объектов в ее составе [3].

Унифицированный язык включает в себя 12 типов диаграмм, среди которых: четыре типа диаграмм относятся к структурным, пять типов описывают поведенческие особенности системы, и три представляют физическое функционирование системы (диаграммы взаимодействия).

Так, наиболее актуальными являются следующие типы диаграмм UML:

* диаграмма классов (Class diagram);
* диаграмма вариантов использования (Usecase diagram);
* диаграмма последовательности (Sequence diagram);
* диаграмма состояний (State diagram);
* диаграмма деятельности (Activity diagram).

Чаще всего диаграммы UML используются для описания рабочих процессов при разработке программного обеспечения. UML-диаграммы доступно передают информацию всем заинтересованным участникам процесса создания продукта.

Также стоит отметить, что при помощи UML можно легко преобразовать объектно-ориентированную модель в схему реляционной базы данных. Первым шагом построена диаграмма вариантов использования (Use Case Diagram), так как именно она является исходным концептуальным представлением программы в процессе ее проектирования и разработки.

Суть данной диаграммы состоит в следующем: проектируемая система представляется в виде множества актеров, взаимодействующих с системой с помощью, так называемых вариантов использования. Диаграмма вариантов использования процесса «Работа Телеграм бота» представлена на рисунке 2.1.

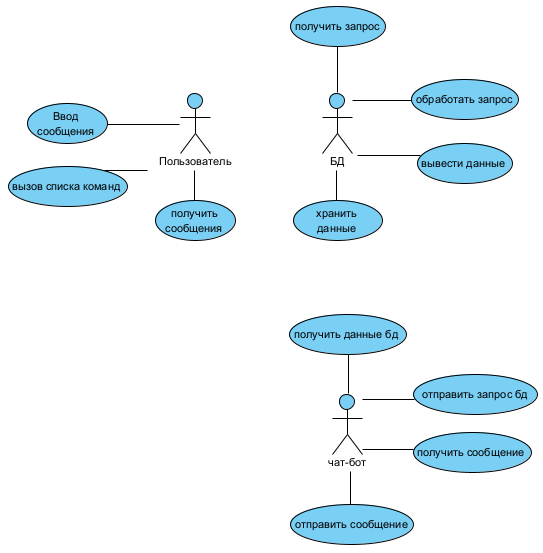


Рисунок 2.1 – Диаграмма вариантов использования процесса «Работа Телеграм бота»

На диаграмме видно, что в процессе работы программы участвуют три актера:

* пользователь;
* база данных;
* чат-бот.

Вторым шагом построена диаграмма последовательности (Sequence Diagram) - диаграмма, которая служит для представления взаимодействия элементов модели в форме последовательности сообщений и соответствующих событий на линиях жизни объектов.

Данные диаграммы фокусируются на времени и визуально показывают порядок взаимодействия, используя вертикальную ось диаграммы для представления времени, когда отправляются сообщения. Полученная диаграмма последовательности процесса использования «Telegram-бота» представлена на рисунке 2.2.

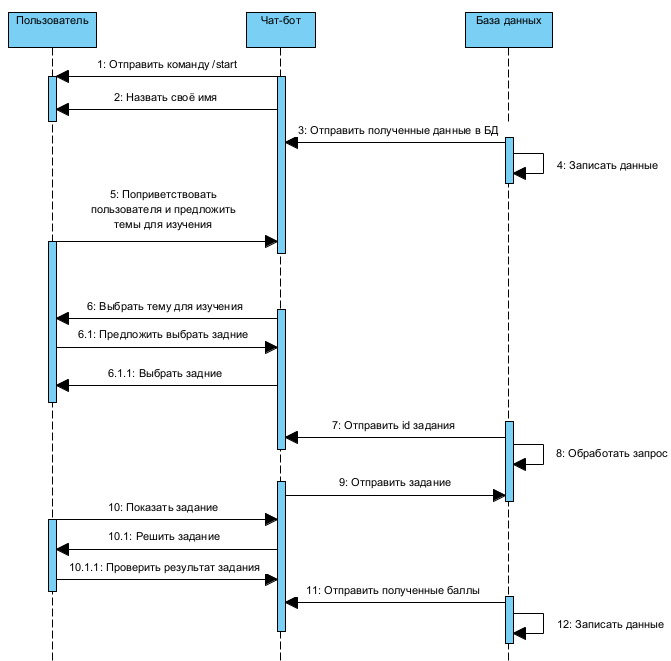


Рисунок 2.2 – Диаграмма последовательности процесса использования «Telegram-бота»

Следующим шагом построена диаграмма активности (Activity diagram), которая используется для создания более «продвинутых» блок-схем, в этом случае часто применяются конструкции принятия решения, а также разделения и слияния потоков управления. Полученная диаграмма активностей процесса «Telegram-бота» представлена на рисунке 2.3.

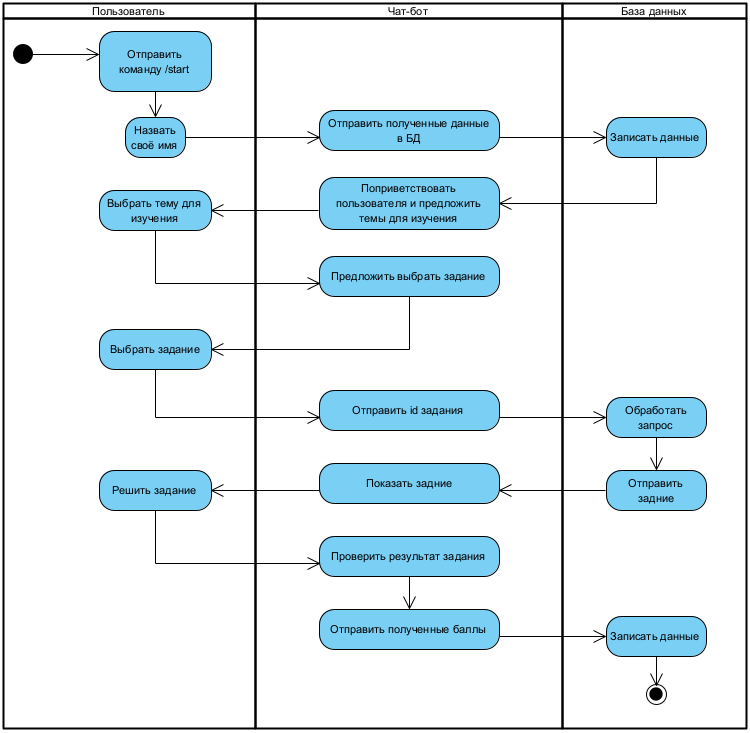


Рисунок 2.3 – Диаграмма активностей процесса «Telegram-бота»

На диаграммах активностей и последовательности можно увидеть весь процесс работы Telegram-бота.

**2.2 Обоснование выбора языка программирования**

После моделирования и анализа бизнес-процессов необходимо выбрать наиболее подходящий язык программирования для разработки программы. На сегодняшний день существует большое количество различных языков программирования, каждый из которых имеет свою специфику применения.

Так, для того чтобы выбрать наиболее актуальный язык программирования, была изучена статистика на 2021 год [4]. Статистика была составлена на основе опроса программистов, работающих в сфере разработки. Для наглядности была создана диаграмма, рисунок 2.4.

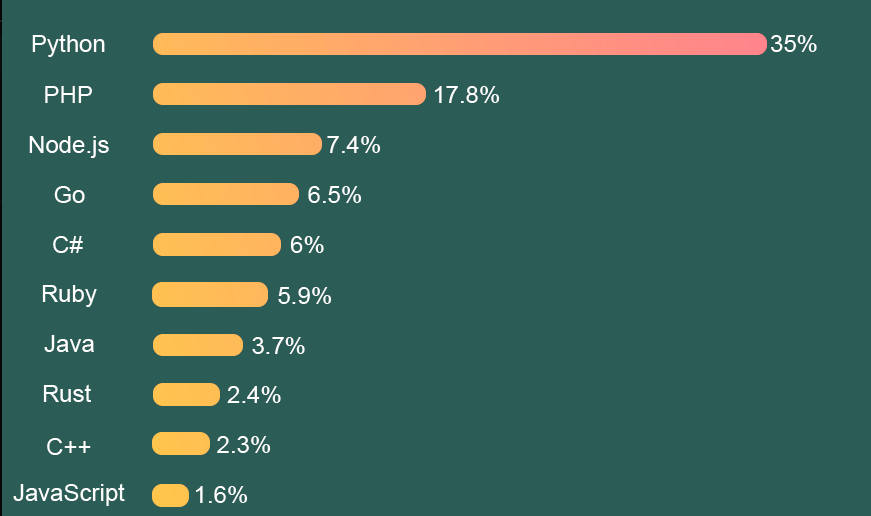


Рисунок 2.4 – Рейтинг языков программирования

Анализируя диаграмму, можно сделать вывод, что большее количество опрошенных программистов рекомендуют язык Python для разработки Telegram-бота. Поэтому этот язык будет использоваться для программы по тестированию мобильного приложения.

Python – высокоуровневый язык программирования общего назначения с динамической строгой типизацией и автоматической управлением памятью. Поддерживающий мультипарадигмный подход (процедурное, императивное, структурное, функциональное, метапрограммирование и объектно-ориентированное программирование).

Достоинства языка Python:

* сборка мусора – автоматическое освобождение памяти, занятой недоступными неиспользуемыми объектами;
* легко читается код, код на Python выглядит очень просто и лаконично он легок для восприятия;
* бесплатный и с открытым исходным кодом, Python разработан под лицензией с открытым исходным кодом одобренным OSI;
* помимо стандартных библиотек Python, к проекту можно подключить множество дополнительных модулей и библиотек;
* интерпретируемый;
* портативный и универсальный, что позволяет работать над проектом на разных машинах и с разными ОС;
* расширяемый, то есть код Python может быть написан и на других языках;
* поддержка графического интерфейса, Python предлагает различные инструменты, такие как wxPython и JPython они позволяют разрабатывать графический интерфейс.

Python – высокоуровневый язык, что позволяет программистам не запоминать всю архитектуру языка.

**2.3 Обоснование выбора среды разработки**

Для разработки стоит выбрать IDE редактор кода. IDE редактор - это программа, предназначенная для разработки программного обеспечения. Как следует из названия, IDE объединяет несколько инструментов, специально предназначенных для разработки. Для Python есть огромное количество сред разработки (IDE). Для выбора следует сравнить их достоинства и недостатки с требованиями к IDE [5].

Python IDLE – редактор, поставляемый вместе с Python. Это базовый, упрощенный режим программирования на Python. Однако IDLE хороший редактор для изучения программирования и понимания основ языка. В нем есть оболочка Python – интерактивный интерпретатор. Его возможности обширны: автозавершение кода, подсветка синтаксиса, подбор отступа и базовый встроенный отладчик. Также, благодаря использованию библиотеки Tkinter, IDLE может применяться на многих платформах, среди которых Windows, Mac OS, Unix-подобные ОС [6].

Интерфейс данного IDE представлен на рисунке 2.5.

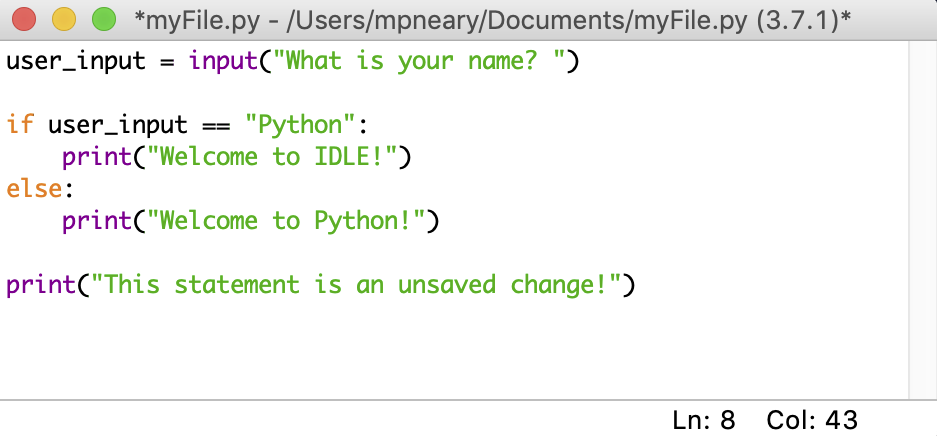


Рисунок 2.5 – Среда разработки Python IDLE

Sublime Text - популярный редактор кода. Он поддерживает множество языков разметки и программирования, такие как: C, C++, C#, CSS, D, Dylan, Erlang, HTML, Groovy, Haskell, Java, JavaScript. В рамках этих языковмы можем добавлять функции через плагины, созданные сообществом и поддерживаемые в рамках лицензий на бесплатное программное обеспечение. Преимущества данного IDE редактора: кроссплатформенная поддержка; открытый исходный код; много плагинов, которые интегрируются в одно место; простота в использовании; полностью настраиваемый [7]. Интерфейс Sublime Text представлен на рисунке 2.6.

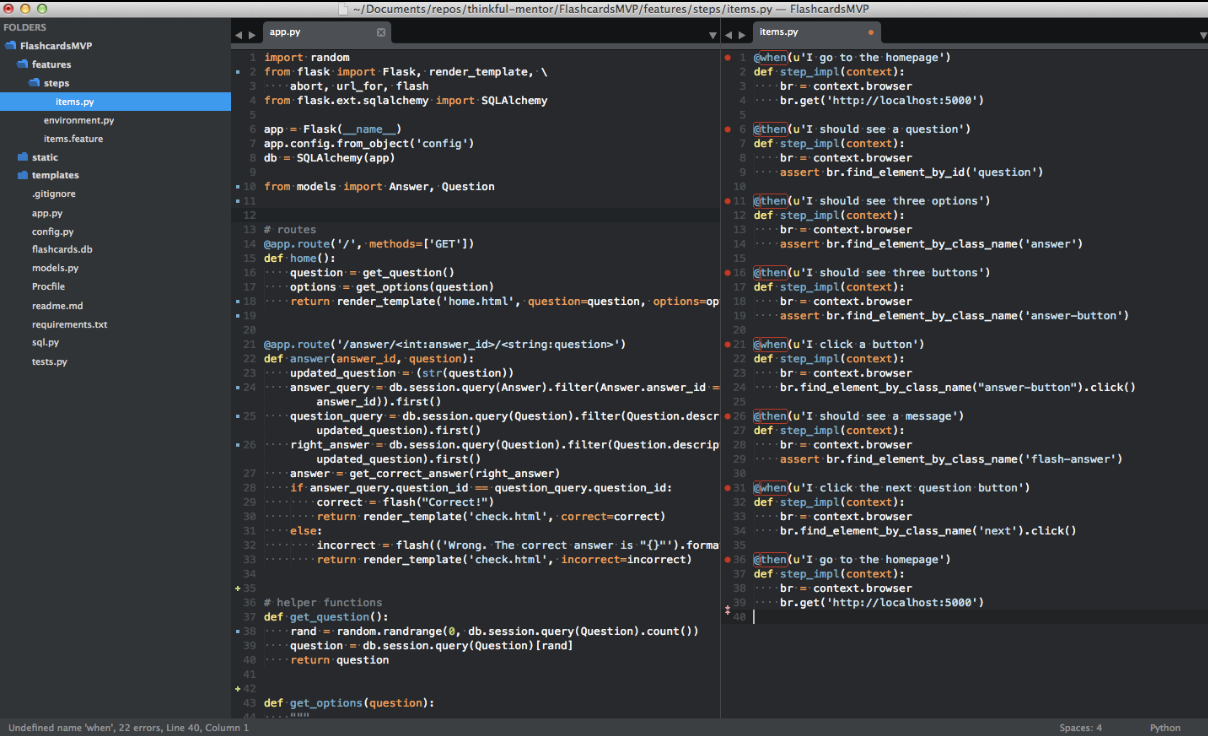


Рисунок 2.6 – Среда разработки Sublime Text

Atom (в прошлом Atomicity) – бесплатный текстовый редактор с открытым исходным кодом для Linux, macOS, Windows с поддержкой плагинов, написанных на JavaScript, и встраиваемых под управлением Git. Большинство плагинов имеют статус свободного программного обеспечения, разрабатываются и поддерживаются сообществом.

Atom основан на Electron (ранее известный как Atom Shell) – фреймворке кроссплатформенной разработки с использованием Chromium и io.js. Поддержку Python можно также подключить с помощью расширения, которое можно установить прямо в Atom.

Из недостатков – отсутствует встроенный компилятор и экрана отладки, их нужно добавлять с помощью расширений [8].

Интерфейс данного IDE представлен на рисунке 2.7.

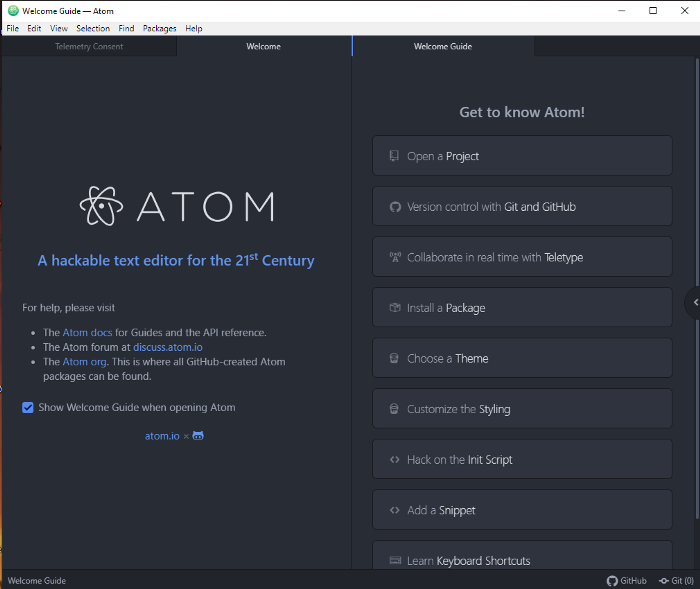


Рисунок 2.7 – Среда разработки Atom

Интегрированная среда разработки Visual Studio – это стартовая площадка для написания, отладки и сборки кода, а также последующей публикации приложений. Помимо стандартного редактора и отладчика, которые есть в большинстве сред IDE, Visual Studio включает в себя компиляторы, средства автозавершения кода, графические конструкторы и многие другие функции для улучшения процесса разработки.

Python Tools for Visual Studio (PTVS) – это бесплатный плагин OSS, который превращает Visual Studio в Python IDE. PTVS поддерживает CPython, IronPython, редактирование, просмотр, Intellisense, смешанную отладку Python/C++, удаленную отладку Linux/MacOS, профилирование, несколько REPL, IPython, Django, Flask, Bottle и т. д. и облачные вычисления с клиентскими библиотеками для Windows, Linux. и MacOS. Интерфейс данного IDE представлен на рисунке 2.8.

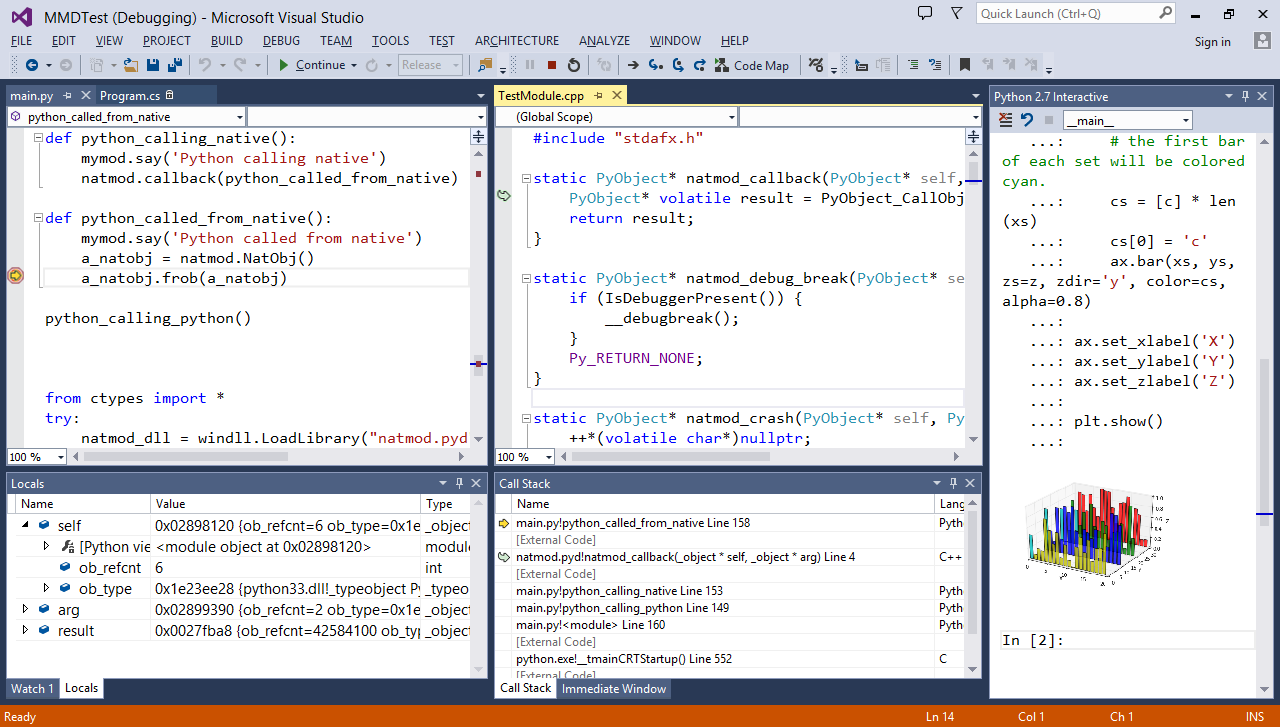


Рисунок 2.8 – Среда разработки Microsoft Visual Studio

PyCharm – это самая интеллектуальная Python IDE с полным набором средств для эффективной разработки на языке Python. Выпускается в двух вариантах – бесплатная версия PyCharm Community Edition и поддерживающая больший набор возможностей PyCharm Professional Edition. PyCharm выполняет инспекцию кода на лету, автодополнение, в том числе основываясь на информации, полученной во время исполнения кода, навигацию по коду, обеспечивает множество рефакторингов. Редактор имеет широкий набор возможностей, таких как автозавершение и инспекции кода, подсветка ошибок, исправления, отладка. IDE доступна на Microsoft Windows, Linux и MacOS [9].

Из недостатков – необходимо наличие у ПК 8 ГБ оперативной памяти, в противном случае IDE притормаживает и работает довольно медленно.

Интерфейс данного IDE представлен на рисунке 2.9.

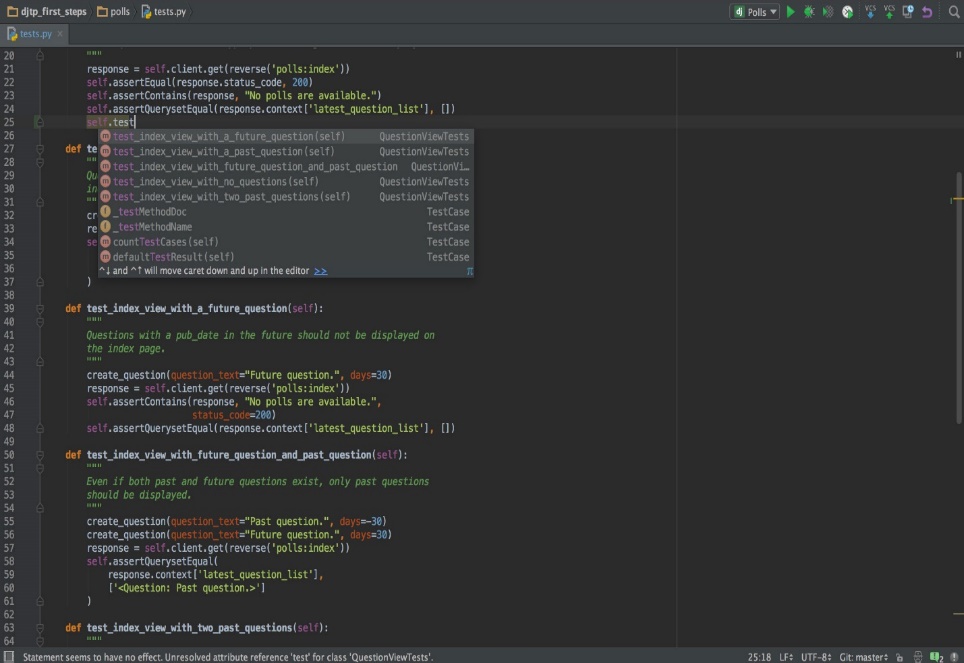


Рисунок 2.9 – Среда разработки PyCharm

После того, как перечислены и описаны наиболее распространенные и популярные среды разработки для языка Python, сопоставим характеристики представленных IDE в форме таблицы.

Результат представлен в таблице 2.1.

Таблица 2.1 – Сравнение сред разработки

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Критерий | Python IDLE | Sublime Text | PyCharm | Atom | Microsoft Visual Studio |
| Бесплатный, в открытом доступе | + | + | + | + | + |
| Встроенный компилятор и/или интерпретатор | + | + | + | - | + |
| Запуск кода из среды | - | + | + | + | + |
| Встроенная отладка | + | + | + | - | + |
| Не требуется расширение для Python | + | - | + | - | - |
| Расширенный функционал | - | - | + | + | + |

Анализируя таблицу, видно, что самыми подходящими средами разработки являются PyCharm и Microsoft Visual Studio, последний следует выбирать в том случае, если разработчик уже знаком с VS и имеет опыт работы. Однако при написании выпускной квалификационной работы в качестве среды разработки будет выбран PyCharm, так как он не перегружен обширным количеством компонентов и сосредоточен на одном языке, что облегчит процесс разработки программного кода.

2.4 Описание структуры базы данных

Перед разработкой базы данных необходимо описать ее структуру при помощи схемы «Сущность-связь» (Entity Relationship). ER-диаграмма показывает взаимосвязь разных сущностей внутри системы. При построении для описания сущностей используются прямоугольники, а для их связи – соединительные линии. ER-диаграммы чаще всего применяются для проектирования и отладки реляционных баз данных в сфере образования, исследования и разработки программного обеспечения и информационных систем для бизнеса [10].

Сущность (entity) представляет тип объектов, которые должны храниться в базе данных. Каждая таблица в базе данных должна представлять одну сущность. Как правило, сущности соответствуют объектам из реального мира. У каждой сущности определяется набор атрибутов.

Над данными линиями обычно указываются числа, описывающие типы бинарных связей (1:1, 1:M, M:N) и обозначающие максимальное количество сущностей на каждой стороне связи.

Проведя анализ предметной области, выделено пятнадцать сущностей: «Пользователь», «Present Simple», «Past Simple», «Present Continuous», «Past Continuous», «Present Perfect», «Past Perfect», «Модальные глаголы», «Инфинитив», «Артикли», «Герундий», «Множественное число», «Словарь, 1 уровень сложности», «Словарь, 2 уровень сложности», «Словарь, 3 уровень сложности».

В таблицах 2.2 – 2.16 дано описание каждой сущности.

Таблица 2.2 – Описание сущности «Пользователь»

|  |  |
| --- | --- |
| Имя поля | Тип данных |
| User\_id | integer |
| Имя пользователя | text |
| Фамилия пользователя | text |
| Ник | text |
| Счёт пользователя | integer |

Таблица 2.3 – Описание сущности «Present Simple»

|  |  |
| --- | --- |
| Имя поля | Тип данных |
| ID | integer |
| Задание | text |
| Ответ | text |

Таблица 2.4 – Описание сущности «Past Simple»

|  |  |
| --- | --- |
| Имя поля | Тип данных |
| ID | integer |
| Задание | text |
| Ответ | text |

Таблица 2.5 – Описание сущности «Present Continuous»

|  |  |
| --- | --- |
| Имя поля | Тип данных |
| ID | integer |
| Задание | text |
| Ответ | text |

Таблица 2.6 – Описание сущности «Past Continuous»

|  |  |
| --- | --- |
| Имя поля | Тип данных |
| ID | integer |
| Задание | text |
| Ответ | text |

Таблица 2.7 – Описание сущности «Present Perfect»

|  |  |
| --- | --- |
| Имя поля | Тип данных |
| ID | integer |
| Задание | text |
| Ответ | text |

Таблица 2.8 – Описание сущности «Past Perfect»

|  |  |
| --- | --- |
| Имя поля | Тип данных |
| ID | integer |
| Задание | text |
| Ответ | text |

Таблица 2.9 – Описание сущности «Модальные глаголы»

|  |  |
| --- | --- |
| Имя поля | Тип данных |
| ID | integer |
| Задание | text |
| Ответ | text |

Таблица 2.10 – Описание сущности «Инфинитив»

|  |  |
| --- | --- |
| Имя поля | Тип данных |
| ID | integer |
| Задание | text |
| Ответ | text |

Таблица 2.11 – Описание сущности «Артикли»

|  |  |
| --- | --- |
| Имя поля | Тип данных |
| ID | integer |
| Задание | text |
| Ответ | text |

Таблица 2.12 – Описание сущности «Герундий»

|  |  |
| --- | --- |
| Имя поля | Тип данных |
| ID | integer |
| Задание | text |
| Ответ | text |

Таблица 2.13 – Описание сущности «Множественное число»

|  |  |
| --- | --- |
| Имя поля | Тип данных |
| ID | integer |
| Задание | text |
| Ответ | text |

Таблица 2.14 – Описание сущности «Словарь, 1 уровень сложности»

|  |  |
| --- | --- |
| Имя поля | Тип данных |
| ID | integer |
| Задание | text |
| Ответ | text |

Таблица 2.15 – Описание сущности «Словарь, 2 уровень сложности»

|  |  |
| --- | --- |
| Имя поля | Тип данных |
| ID | integer |
| Задание | text |
| Ответ | text |

Таблица 2.16 – Описание сущности «Словарь, 3 уровень сложности»

|  |  |
| --- | --- |
| Имя поля | Тип данных |
| ID | integer |
| Задание | text |
| Ответ | text |

Так, полученная ER-диаграмма представлена на рисунке 2.10.

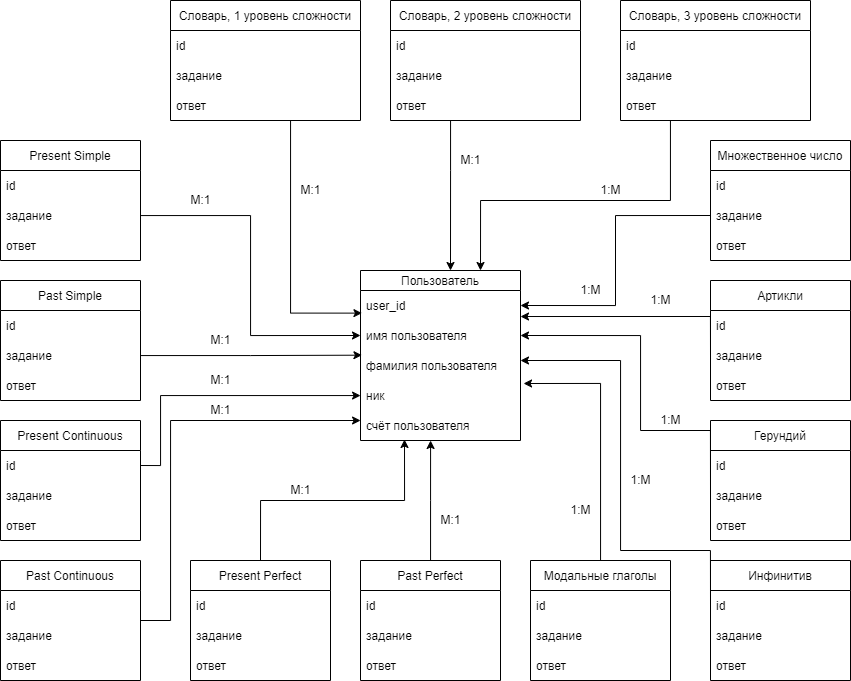


Рисунок 2.10 – Концептуальная модель базы данных

**2.5 Вывод о проектировании программы**

Подводя итоги проектирования, можно сказать, что создание Telegram-бота во многом автоматизирует работу преподавателей. Также методом сравнительного анализа произведен выбор языка программирования, среды разработки.

Таким образом, для написания программного продукта необходимо использовать язык Python, разработку вести в среде PyCharm, а внедрять систему следует на базе платформы Telegram. Также методом сравнительного анализа произведен выбор языка программирования, среды разработки. Таким образом, для написания программного продукта необходимо использовать язык Python, разработку вести в среде PyCharm, а внедрять систему следует на базе платформы Telegram.

3 Разработка программы

3.1 Разработка логики чат-бота

В первую очередь при разработке логики программы было создано основное меню, которое состоит из следующих разделов: «Список тем», «Работа со словарём», «Таблица лидеров».

Несмотря на то, что меню можно открывать прописывая сообщения с их названием вручную, была добавлена специальная клавиатура, прописанная внутри кода, клавиатура выглядит как кнопки, расположенные внизу экрана, которые можно активировать нажатием пальца, если работа идет с сенсорным экраном, либо по нажатию кнопки мыши, если пользователь работает за персональным компьютером.

Следующим этапом было создано меню, в котором содержатся все темы заданий, представленные в боте. Меню содержит 8 кнопок: «Инфинитив», «Множественное число», «Времена», «Артикли», «Герундий», «Модальные глаголы», «Правила», «Назад».

Внутри меню «Времена» было прописано 8 кнопок, 6 из которых отвечают за переход к заданиям на выбранные темы, такие как: «Present Simple», «Past Simple», «Present Continuous», «Past Continuous», «Present Perfect», «Past Perfect», а также кнопки вызова правил и ухода в главное меню.

Далее было создано меню, которое открывается по нажатию на раздел «Таблица лидеров». Меню состоит из трёх кнопок: «Вывести лучшую десятку», «Вывести своё место», «Назад».

После этого было создано меню для режима «Работа со словарём». В этом разделе описаны следующие кнопки: «1 уровень сложности», «2 уровень сложности», «3 уровень сложности», «Правила режима «Работа со словарём», «Назад».

Чтобы прописать возможность бота реагировать на введенные пользователем сообщения, такие как выборы пунктов меню, или ответы на задания, был разработан специальный метод, также для того чтобы избежать ввод пользователем неизвестных или не ожидаемых команд были созданы несколько возможных состояний.

В зависимости от представленных состояний происходит идентификация ответа. Например, если состояние command\_not\_defined, то бот будет обрабатывать отправленные пользователем сообщения как неизвестные команды. Если же состояние бота будет иметь значение command\_dictionary\_one, то отправленные пользователем сообщения будут трактоваться, как ответ на задание, код функциональной части представлен в приложении А.

Перед тем, как реализовывать методы для создания перечисленных команд используется декоратор, который реагирует на входящие сообщения. В данный декоратор могут передаваться команды, лямбда-функции, регулярные выражения, поддерживаемые типы содержимого сообщений, типы чатов [11].

3.2 Разработка и интеграция базы данных

Так же была осуществлена работа с базами данных. Подключение будет осуществляться к базе данных SQLite, так как по умолчанию стандартная библиотека Python содержит модуль sqlite3.

Сперва необходимо загрузить библиотеку. Для этого нужно использовать следующую команду: import sqlite3 [12]. Далее создадим соединение и укажем путь к имеющейся базе данных при помощи кода, представленного на рисунке 3.1.

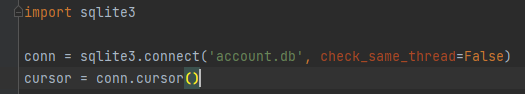


Рисунок 3.1 – Создание подключения к базе данных

В данном коде происходит импорт sqlite3. Затем создаётся переменная conn, которая принимает путь к базе данных SQLite. Метод connect() принимает в качестве параметра путь к базе данных SQLite [13]. Если база данных в указанном месте существует, будет установлено соединение, в противном случае по указанному пути будет создана новая база данных и так же установлено соединение.

Затем создаётся переменная cursor, которая при использовании будет обращаться к подключенной БД.

Создание самых таблиц для БД происходит следующим образом.

Сперва в SQLiteStudio выбирается опция Create table. На рисунке 3.2 показана опция «Create table».

Затем создается столбец с именем, в нем так же указывается тип данных. Активируются следующие ограничения:

* уникальность. Уникальное ограничение - это отдельное поле или комбинация полей, однозначно определяющая запись. Некоторые поля могут содержать нулевые значения, если комбинация значений является уникальной [14];
* not Null. По умолчанию столбец может содержать значения null. Если столбцу запрещено иметь значение null, нужно определить такое ограничение для этого столбца, указав, что null теперь не разрешено для этого столбца. Null не совпадает с данными, а представляет собой неизвестные данные [15].

На рисунке 3.3 показан процесс создания столбца.

После создания первого столбца, присвоить таблице имя, чтобы однозначно ее идентифицировать

При необходимости создаются новые таблицы.

В столбцы вносятся значения, соответствующие указанным типам данных.

На рисунке 3.4 показан пример готовой таблицы.

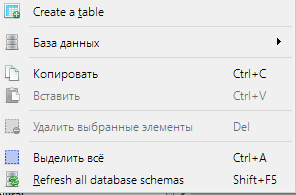


Рисунок 3.2 – Опция «Create a table»

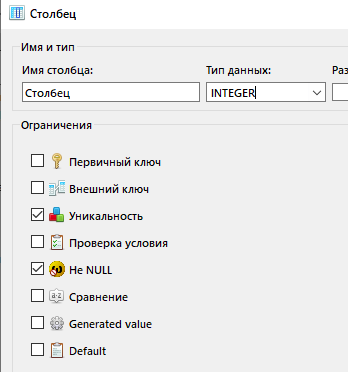


Рисунок 3.3 – Меню создания столбца

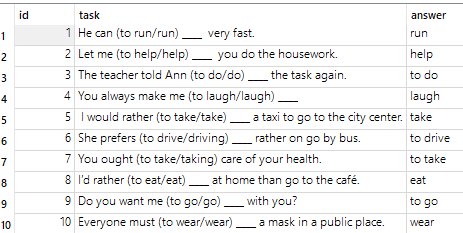


Рисунок 3.4 – Таблица «Infinitive»

По представленному типажу сделаны все таблицы заданий, первый столбец отвечает за id задания, чтобы по нему имелась возможность обратиться к строке, второй столбец представляет собой само задание, которое будет отображаться пользователю, во время прохождения тестов, третий столбец содержит в себе список ответов.

Готовая база данных содержит в себе 15 таблиц, 14 из которых содержат в себе все задания, поделенные на темы, оставшаяся таблица хранит в себе информацию о пользователях, зарегистрировавшихся в боте. На рисунке 3.5 представлена таблица пользователей, на рисунке 3.6 изображена итоговая база данных со всеми, содержащимися в ней таблицами.

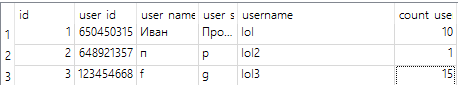


Рисунок 3.5 – Таблица «users»

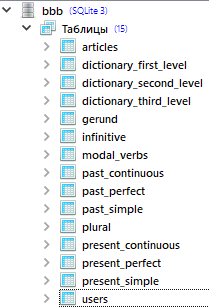


Рисунок 3.6 – Заполненная база данных

В первый раз мы обращаемся к базе данных, когда начинаем общение с нашим Telegram-ботом. В базу данных добавляется данные о пользователе, такие как: «user\_id», «user\_name», «user\_surname», «username», «count». Код описан в методе db\_table\_val() и представлен на рисунке 3.7.

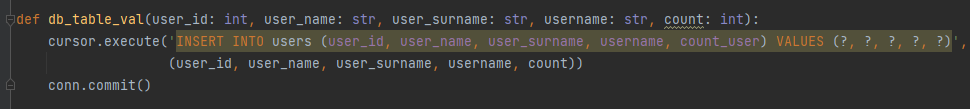


Рисунок 3.7 – Добавление пользователя в БД

Ещё мы взаимодействуем с базами данных на этапе автоматического выбора задания. Рассмотрим на примере выбора задания для «1 уровня сложности» в режиме «Работа со словарём». Бот выбирает случайное задание из заданного диапазона и делает запрос по «id» в базу данных. Результатом запроса является строка, которая позже делится и раскладывается в соответствующие переменные. Код метода choice\_task\_from\_dictionary\_first() представлен на рисунке 3.8.

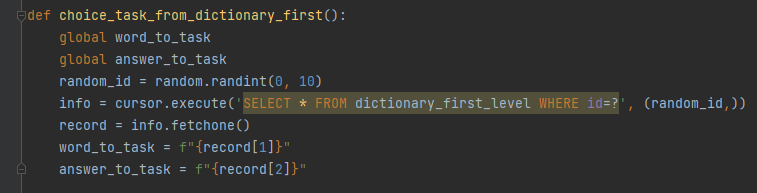


Рисунок 3.8 – Код метода choice\_task\_from\_dictionary\_first()

По аналогии сделаны методы для остальных заданий, в которых идут запросы в свою таблицу подключенной базы данных.

Также взаимодействие с базами данных происходит и на зачисления очков за правильные ответы. Бот сравнивает полученный ответ с правильным записанным на предыдущем этапе, и если всё верно, то пользователю начисляется определённое количество очков за правильно выполненное задание. Если же ответ не верный, то пользователю выводится соответствующее сообщение и указывается правильный ответ. Вся логика описана в методе check\_answer(). Полный код базы расположен в приложении Б. Код представлен на рисунке 3.9.

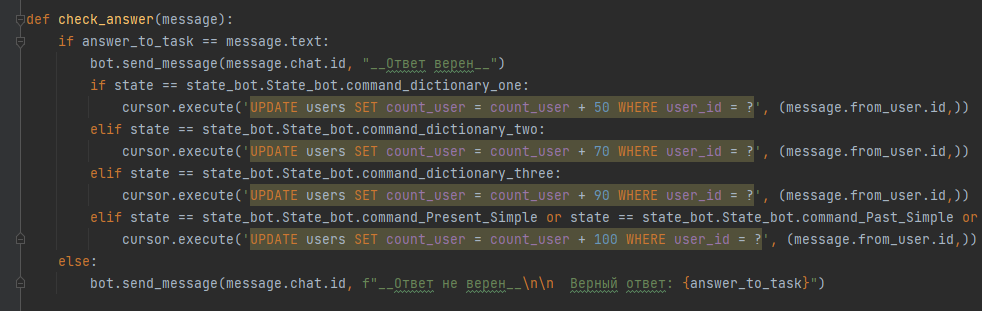


Рисунок 3.9 – Код метода check\_answer()

3.3 Вывод по разрабатываемому продукту

В третьей главе на языке Python в среде программирования PyCharm и при помощи СУБД SQLite был разработан прототип Телеграм бота. Была прописана логика программ, описаны функции нажатия кнопок, обработаны исключения, прописана возможность считывания ответов пользователя на задания. Было проведено подключение к базе данных и реализованы запросы, а также возможность внесения информации в упомянутую базу данных о пользователях, такой как: логин и число очков, набранных за прохождение заданий. Программа функционирует на платформе Telegram и предоставляет доступ как со смартфонов, так и с персональных компьютеров, невзирая на операционную сеть.

4 Руководство пользователя

4.1 Подготовка к работе

Перед началом работы с чат-ботом необходимо установить на устройство мессенджер «Telegram». Файл установки можно найти на официальном сайте <https://tlgrm.ru> или в интернет-магазине приложений, например, Google Play Market.

После установки необходимо произвести регистрацию.

Алгоритм регистрации для платформы Android следующий:

* установить приложение через Google Play Market;
* открыть приложение;
* выбрать страну;
* ввести свой номер телефона;
* подождать, пока на указанный номер не придет сообщение с проверочным кодом, после чего необходимо ввести указанный код в предназначенное для этого окно.

Последовательность действий для регистрации в Telegram на платформе IOS следующая:

* установить приложение через App Store;
* открыть мессенджер, нажать на кнопку «Start messaging»;
* ввести свой номер телефона, нажать на кнопку «Далее»;
* ввести проверочный код из сообщения, либо же код сам автоматически добавится в окно, если приложение открыто;
* указать имя и фамилию, при желании выбрать фотографию профиля, нажать далее.

Для того чтобы зарегистрироваться в Telegram на устройствах Windows phone нужно воспроизвести следующие действия:

* загрузить приложение из официального магазина Microsoft Store;
* открыть мессенджер;
* ввести проверочный код из сообщения в предназначенное для этого окно;
* указать свои контактные данные.

Процесс регистрации и установки приложения на персональный компьютер схож с Android версией:

* запускается установочный файл и указывается папка установки, на рисунке 4.1 показано окно установки;
* после установки запустить приложение, появится окно «Start messaging», нажать на появившуюся кнопку, на рисунке 4.2 изображено данное окно;
* в появившемся окне ввести номер телефона, как показано на рисунке 4.3;
* ввести полученный код, который должен прийти на мобильный телефон, после этого указывается имя, фамилия, по желанию выставляется фотография, это окно продемонстрировано на рисунке 4.4.

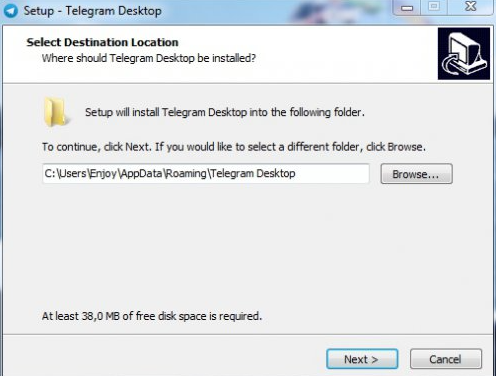


Рисунок 4.1 – Окно установки Telegram

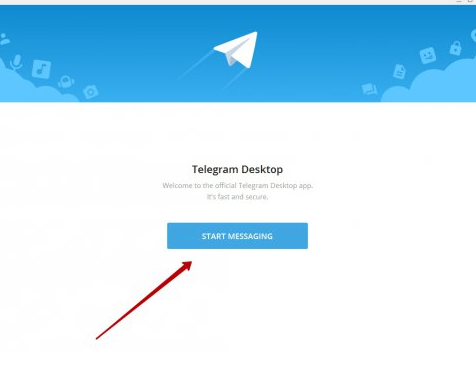


Рисунок 4.2 – Окно «Start messaging»

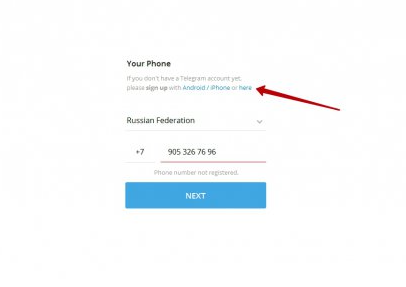


Рисунок 4.3 – Окно ввода номера телефона

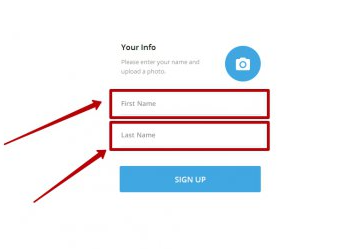


Рисунок 4.4 – Окно ввода информации о пользователе

Для того, чтобы найти бота, в поисковую строку следует ввести @Trial4567\_bot и отправить любое сообщение, либо выбрать команду /start. Результат представлен на рисунке 4.5.

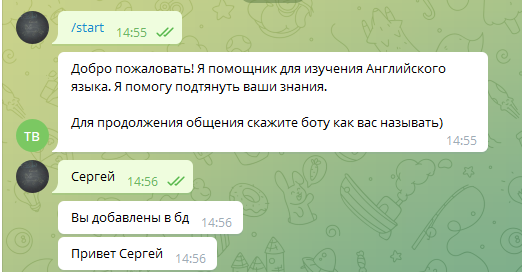


Рисунок 4.5 – Запуск чат-бота

4.2 Описание операций

Для работы с чат-ботом можно использовать кнопки, предлагаемые ботом. Процесс ввода команды представлен на рисунке 4.6.

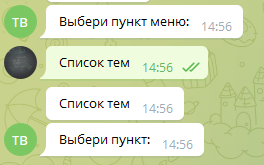


Рисунок 4.6 – Ввод и выбор команды

Чат-бот имеет главное меню, которое включает в себя следующие основные пункты: «Список тем», «Работа со словарем», «Таблицы лидеров».

Главное меню представлено на рисунке 4.7.

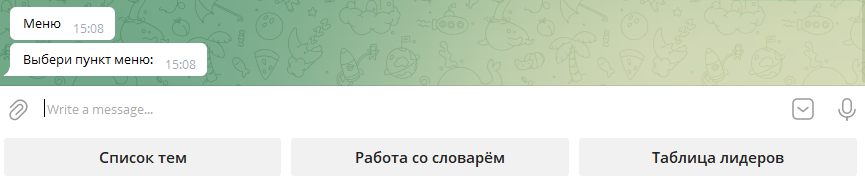


Рисунок 4.7 – Главное меню чат-бота

Во вкладке «Список тем» отображаются пункты: «Времена», «Модальные глаголы», «Инфинитив», «Артикли», «Герундий», «Множественное число».

Дополнительные пункты меню представлены на рисунке 4.8.

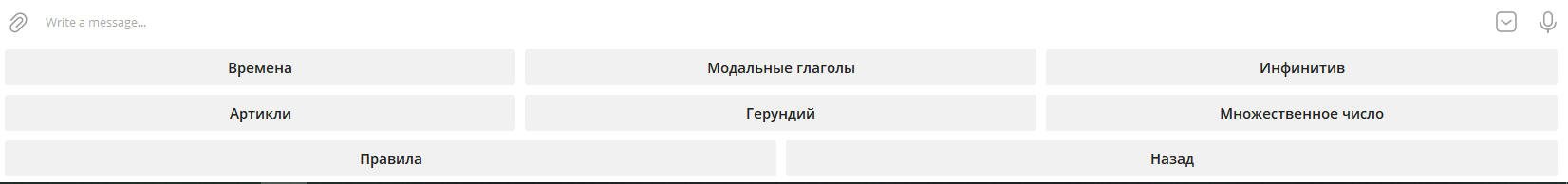


Рисунок 4.8 – Дополнительные пункты меню чат-бота

Во вкладке «Времена» отображаются пункты: «present simple», «past simple», «present continuous», «past continuous», «present perfect», «Past perfect», «правила».

На рисунке 4.9 изображено меню «Времена»



Рисунок 4.9 – Меню «Времена»

При выборе какого-либо из времен начнется тест состоящий из 5 вопросов, на каждый из которых нужно дать ответ, введя его в строку сообщений, разновидность задания представлена на рисунке 4.10.

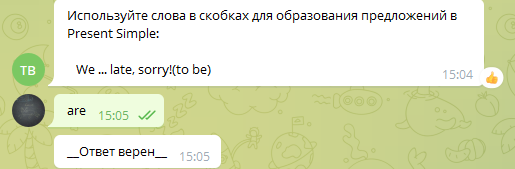


Рисунок 4.10 – Разновидность задания

Во вкладке «Таблица лидеров» отображаются два пункта «Вывести лучшую десятку» и «Вывести своё место». При выборе первого нам показывают десять лучших пользователей, при выборе второго пункта показывается лишь место, которое занимает пользователь.

На рисунке 4.11. представлено меню «Таблица лидеров». Н

На рисунке 4.12 – результат команды «Вывести лучшую десятку».

На рисунке 4.13 – результат команды «Вывести свое место».

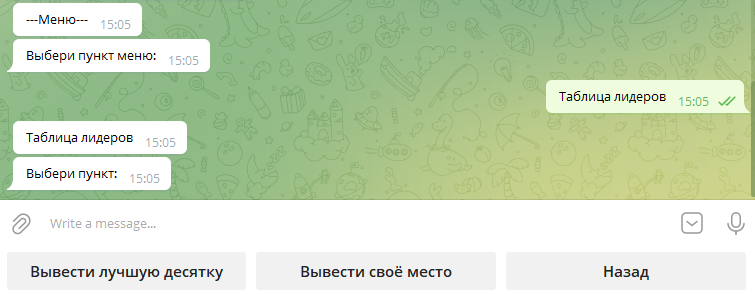


Рисунок 4.11 – Меню «Таблица лидеров»

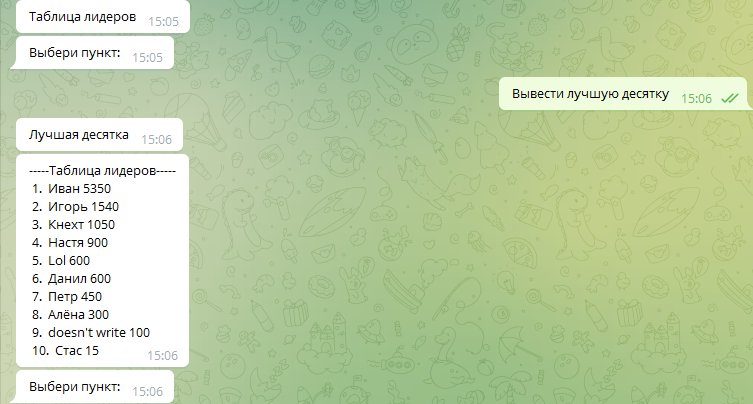


Рисунок 4.12 – Результат команды «Вывести лучшую десятку»



Рисунок 4.13 – Результат команды «Вывести свое место»

Во вкладке «Работа со словарём» отображаются пункты «1 уровень сложности», «2 уровень сложности», «3 уровень сложности», «Правила режима «Работа со словарём»». В первых трёх пунктах у нас выдаётся слова для перевода с различной сложностью, в последнем пункте у нас выдаётся информация об этом режиме. На рисунке 4.14 представлено меню «Работа со словарем», на рисунке 4.15 представлен выбор сложности словаря и разновидность задания, на рисунке 4.16. изображен вывод правил при выборе соответствующего пункта меню.

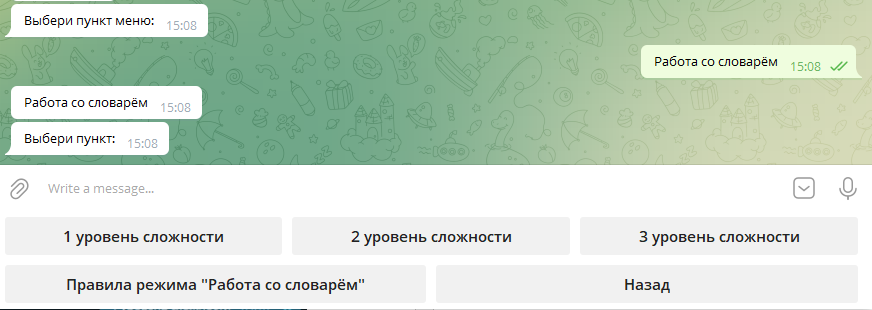


Рисунок 4.14 – Меню «Работа со словарем»

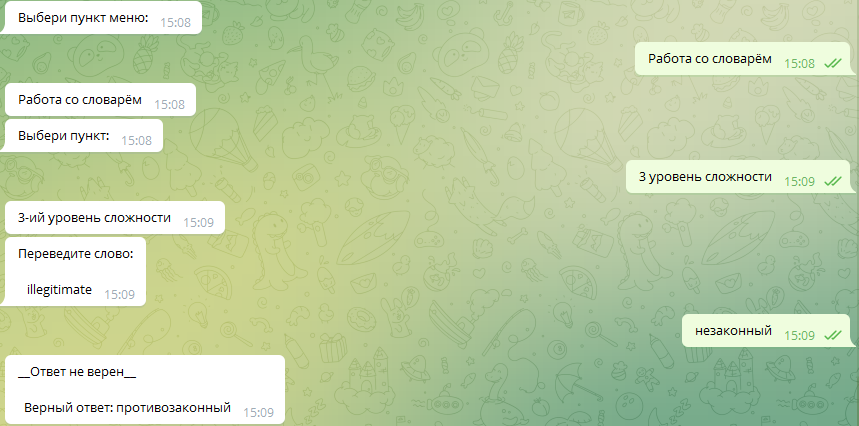


Рисунок 4.15 – Работа со словарем

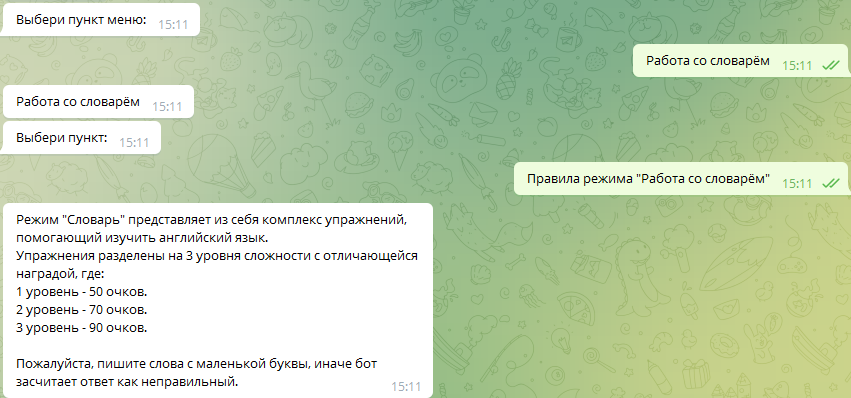


Рисунок 4.16 – Правила режима «Работа со словарем»

ЗАКЛЮЧЕНИЕ

Цель данной работы заключалась в написании прототипа Telegram бота «Помощник по изучению английского языка»

Для достижения поставленной цели был решен ряд задач:

* описаны бизнес-процессы, проходящие в ходе работы Telegram-бота;
* построена концептуальная модель, являющаяся отражением предметной области в виде совокупности понятий, их характеристик и связей;
* обоснован выбор языка и среды разработки. Языком разработки был выбран Python, средой – Pycharm. Для разработки базы данных была выбрана среда SQLite;
* разработан Telegram-бот;
* разработано руководство пользователя.

Работа была проведена в 4 этапа: аналитическое исследование, проектирование Телеграм бота, разработка Телеграм бота, написание руководства пользователя. В процессе проведения этих этапов были решены поставленные задачи, что привело к достижению цели. Данную выпускную квалификационную работу можно считать полностью проработанной и завершенной.

В результате выполнения данной работы был создан рабочий прототип Телеграм бота «Помощник по изучению английского языка», полностью удовлетворяющий всем требованиям.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Документации: Телеграм-боты [Электронный ресурс]. – URL: https://tlgrm.ru/docs/bots (Дата обращения: 03.06.2022).

2 Примеры чат-ботов. Удачные решения, которые можно использовать в бизнесе [Электронный ресурс]. – URL: https://sendpulse.com/ru/blog/chatbot-examples (Дата обращения: 03.06.2022).

3 Разновидности диаграмм UML [Электронный ресурс]. – URL: https://www.lucidchart.com/blog/ru/types-of-UML-diagrams (Дата обращения: 03.06.2022).

4 Создание бота, какой язык программирования выбрать? [Электронный ресурс]. – URL: https://qna.habr.com/q/247221 (Дата обращения: 03.06.2022).

5 Выбираем самый удобный редактор кода на Python [Электронный ресурс]. – URL: https://habr.com/ru/company/skillfactory/blog/521838/ (Дата обращения: 03.06.2022).

6 Python - IDLE [Электронный ресурс]. – URL: https://www.tutorialsteacher.com/python/python-idle (Дата обращения: 03.06.2022).

7 Плагин Sublime Text и среда Python [Электронный ресурс]. – URL: https://russianblogs.com/article/4063765703/ (Дата обращения: 03.06.2022).

8 Atom [Электронный ресурс]. – URL: https://atom.io/ (Дата обращения: 03.06.2022).

9 Возможности – PyCharm [Электронный ресурс]. – URL: https://www.jetbrains.com/ru-ru/pycharm/features/ (Дата обращения: 03.06.2022).

10 Проектирование ER-диаграммы [Электронный ресурс]. – URL: https://nationalteam.worldskills.ru/skills/proektirovanie-er-diagrammy/ (Дата обращения: 03.06.2022).

11 Пишем бота для Telegram на языке Python [Электронный ресурс]. – URL: https://www.gitbook.com/book/groosha/telegram-bot-lessons/details (Дата обращения: 03.06.2022).

12 Python, введение в БД [Электронный ресурс]. – URL: https://habr.com/ru/post/481084/ (Дата обращения: 03.06.2022).

13 Учебник по SQLite3 в Python [Электронный ресурс]. – URL: https://digitology.tech/posts/uchebnik-po-sqlite3-v-python/ (Дата обращения: 03.06.2022).

14 Unique constraints SQLite [Электронный ресурс]. – URL: https://oracleplsql.ru/unique-constraints-sqlite.html (Дата обращения: 03.06.2022).

15 Первичные ключи в базах данных SQLite [Электронный ресурс]. – URL: https://zametkinapolyah.ru/zametki-o-mysql (Дата обращения: 03.06.2022).

ПРИЛОЖЕНИЕ А

**Код функциональной части программы**

**import** random  
**import** sqlite3  
**from** telebot **import** types  
**import** telebot  
**from** enum **import** Enum  
  
**import** state\_bot  
  
*#токен бота*bot = telebot.TeleBot(**"5367300522:AAHdcl90tYRgQCR\_0LZF1SnZm3nBgfEGcp0"**)  
  
*#подключени бд*conn = sqlite3.connect(**'account.db'**, check\_same\_thread=**False**)  
cursor = conn.cursor()  
  
state = state\_bot.State\_bot.command\_not\_defined  
count = 0  
progress\_task = 0  
word\_to\_task = **""**answer\_to\_task = **""**return\_word = **""**task\_avability = 0  
  
  
*#добавление новогопользователя в бд***def** db\_table\_val(user\_id: int, user\_name: str, user\_surname: str, username: str, count: int):  
 cursor.execute(**'INSERT INTO users (user\_id, user\_name, user\_surname, username, count\_user) VALUES (?, ?, ?, ?, ?)'**,  
 (user\_id, user\_name, user\_surname, username, count))  
 conn.commit()  
  
  
@bot.message\_handler(commands=[**'start'**])  
**def** start\_message(message):  
 bot.send\_message(message.chat.id, **'Добро пожаловать! Я помощник для изучения Английского языка. Я помогу '  
 'подтянуть ваши знания.\n\nДля продолжения общения скажите боту как вас называть)'**)  
  
  
*#обработка всех текстовых сообщений*@bot.message\_handler(content\_types=[**'text'**])  
**def** get\_text\_messages(message):  
 **global** count  
 **global** state  
 **global** task\_avability  
 **global** progress\_task  
 **global** dictionary  
 **global** return\_word  
 **if** message.text == **'Таблица лидеров'**:  
 bot.send\_message(message.chat.id, **'Таблица лидеров'**)  
 make\_keyboard\_table\_liders(message)  
 return\_word = **'Меню'  
 elif** message.text == **'Работа со словарём'**:  
 bot.send\_message(message.chat.id, **'Работа со словарём'**)  
 make\_keyboard\_dictionary(message)  
 return\_word = **'Меню'  
 elif** message.text == **'Список тем'**:  
 bot.send\_message(message.chat.id, **'Список тем'**)  
 make\_keyboard\_topic\_selection(message)  
 return\_word = **'Меню'  
 elif** message.text == **"Вывести лучшую десятку"**:  
 bot.send\_message(message.chat.id, **'Лучшая десятка'**)  
 conclusion\_top\_ten\_users(message)  
 make\_keyboard\_table\_liders(message)  
 **elif** message.text == **"Вывести своё место"**:  
 bot.send\_message(message.chat.id, **'Место'**)  
 conclusion\_user\_position(message)  
 make\_keyboard\_table\_liders(message)  
 **elif** message.text == **"1 уровень сложности"**:  
 bot.send\_message(message.chat.id, **'1-ый уровень сложности'**, reply\_markup=types.ReplyKeyboardRemove())  
 zeroing\_variables()  
 state = state\_bot.State\_bot.command\_dictionary\_one  
 unidefeined\_commands(message)  
 **elif** message.text == **"2 уровень сложности"**:  
 bot.send\_message(message.chat.id, **'2-ой уровень сложности'**, reply\_markup=types.ReplyKeyboardRemove())  
 zeroing\_variables()  
 state = state\_bot.State\_bot.command\_dictionary\_two  
 unidefeined\_commands(message)  
 **elif** message.text == **"3 уровень сложности"**:  
 bot.send\_message(message.chat.id, **'3-ий уровень сложности'**, reply\_markup=types.ReplyKeyboardRemove())  
 zeroing\_variables()  
 state = state\_bot.State\_bot.command\_dictionary\_three  
 unidefeined\_commands(message)  
 **elif** message.text == **"Правила режима \"Работа со словарём\""**:  
 bot.send\_message(message.chat.id, **"Режим \"Словарь\" представляет из себя комплекс упражнений, помогающий "  
 "изучить английский язык. \nУпражнения разделены на 3 уровня сложности с "  
 "отличающейся наградой, где:\n1 уровень - 50 очков.\n2 уровень - 70 "  
 "очков.\n3 уровень - 90 очков.\n\nПожалуйста, пишите слова с маленькой "  
 "буквы, иначе бот засчитает ответ как неправильный."**)  
 make\_keyboard\_dictionary(message)  
 **elif** message.text == **"Времена"**:  
 make\_keyboard\_time(message)  
 return\_word = **"Список тем"  
 elif** message.text == **"Present Simple"**:  
 make\_keyboard\_present\_simple(message)  
 return\_word = **"Времена"  
 elif** message.text == **"Задание на \"Present Simple\""**:  
 bot.send\_message(message.chat.id, **"Present Simple"**, reply\_markup=types.ReplyKeyboardRemove())  
 state = state\_bot.State\_bot.command\_Present\_Simple  
 unidefeined\_commands(message)  
 **elif** message.text == **"Правило для темы \"Present Simple\""**:  
 bot.send\_message(message.chat.id, **"Утверждение\n\nПодлежащее + V1. Глагол не изменяется (первая форма "  
 "глагола). Исключением является глагол to be, который изменяется по числам "  
 "и лицам.\n\nОтрицание\n\nПодлежащее + do/does + not + V1 (does для 3-го лица "  
 "ед. числа)\n\nВопрос\n\nDo/does + подлежащее + V1"**)  
 **elif** message.text == **"Past Simple"**:  
 make\_keyboard\_past\_simple(message)  
 return\_word = **"Времена"  
 elif** message.text == **"Задание на \"Past Simple\""**:  
 bot.send\_message(message.chat.id, **"Past Simple"**, reply\_markup=types.ReplyKeyboardRemove())  
 state = state\_bot.State\_bot.command\_Past\_Simple  
 unidefeined\_commands(message)  
 **elif** message.text == **"Правило для темы \"Past Simple\""**:  
 bot.send\_message(message.chat.id, **"Утверждение\n\nПодлежащее + V2. 2-я форма у правильных глаголов образуется "  
 "путем прибавления к основе окончания –ed. Неправильные глаголы необходимо "  
 "запомнить.\n\nОтрицание\n\nПодлежащее + did + not + V1.\n\nВопрос\n\nDid + "  
 "подлежащее + V1?"**)  
 **elif** message.text == **"Present Continuous"**:  
 make\_keyboard\_present\_continuous(message)  
 return\_word = **"Времена"  
 elif** message.text == **"Задание на \"Present Continuous\""**:  
 bot.send\_message(message.chat.id, **"Present Continuous"**, reply\_markup=types.ReplyKeyboardRemove())  
 state = state\_bot.State\_bot.command\_Present\_Continuous  
 unidefeined\_commands(message)  
 **elif** message.text == **"Правило для темы \"Present Continuous\""**:  
 bot.send\_message(message.chat.id, **"Утверждение\n\nПодлежащее + am/is/are + смысловой глагол с окончанием "  
 "-ing\n\nОтрицание\n\nПодлежащее + am/is/are + not + "  
 "V-ing\n\nВопрос\n\nAm/is/are + подлежащее + V-ing?"**)  
 **elif** message.text == **"Past Continuous"**:  
 make\_keyboard\_past\_continuous(message)  
 return\_word = **"Времена"  
 elif** message.text == **"Задание на \"Past Continuous\""**:  
 bot.send\_message(message.chat.id, **"Past Continuous"**, reply\_markup=types.ReplyKeyboardRemove())  
 state = state\_bot.State\_bot.command\_Past\_Continuous  
 unidefeined\_commands(message)  
 **elif** message.text == **"Правило для темы \"Past Continuous\""**:  
 bot.send\_message(message.chat.id, **"Утверждение\n\nПодлежащее + was/were + смысловой глагол с окончанием "  
 "–ing\n\nОтрицание\n\nПодлежащее + was/were + not + V-ing\n\nВопрос\n\nWas/were"  
 " + подлежащее + V-ing?"**)  
 **elif** message.text == **"Present Perfect"**:  
 make\_keyboard\_present\_perfect(message)  
 return\_word = **"Времена"  
 elif** message.text == **"Задание на \"Present Perfect\""**:  
 bot.send\_message(message.chat.id, **"Present Perfect"**, reply\_markup=types.ReplyKeyboardRemove())  
 state = state\_bot.State\_bot.command\_Present\_Perfect  
 unidefeined\_commands(message)  
 **elif** message.text == **"Правило для темы \"Present Perfect\""**:  
 bot.send\_message(message.chat.id, **"Утверждение\n\nПодлежащее + have/has + V3\n\nОтрицание\n\nПодлежащее + "  
 "have/has + not +V3"**)  
 **elif** message.text == **"Past Perfect"**:  
 make\_keyboard\_past\_perfect(message)  
 return\_word = **"Времена"  
 elif** message.text == **"Задание на \"Past Perfect\""**:  
 bot.send\_message(message.chat.id, **"Past Perfect"**, reply\_markup=types.ReplyKeyboardRemove())  
 state = state\_bot.State\_bot.command\_Past\_Perfect  
 unidefeined\_commands(message)  
 **elif** message.text == **"Правило для темы \"Past Perfect\""**:  
 bot.send\_message(message.chat.id, **"Утверждение\n\nПодлежащее + had + V3\n\nОтрицание\n\nПодлежащее + had + not "  
 "+ V3"**)  
 **elif** message.text == **"Модальные глаголы"**:  
 make\_keyboard\_modal\_verbs(message)  
 return\_word = **"Список тем"  
 elif** message.text == **"Задание на \"Модальные глаголы\""**:  
 bot.send\_message(message.chat.id, **"Модальные глаголы"**, reply\_markup=types.ReplyKeyboardRemove())  
 state = state\_bot.State\_bot.command\_Modal\_Verbs  
 unidefeined\_commands(message)  
 **elif** message.text == **"Правило для темы \"Модальные глаголы\""**:  
 bot.send\_message(message.chat.id, **"• can — мочь, уметь\n• could — мог, умел\n• must — должен, нужно, "  
 "надо\n• have to — быть вынужденным, быть должным, приходится, "  
 "надо\n• may — стоит, следует\n• might — должен, следует, стоит\n• should "  
 "— мочь, мог бы\n• ought to — мочь, мог бы\n\nПравила употребления "  
 "модальных глаголов\n1. Модальный глагол всегда употребляется со смысловым "  
 "глаголом, который стоит в начальной форме (must do, could see).\nI can type "  
 "very fast. — Я могу печатать очень быстро.\n2. Модальные глаголы имеют "  
 "одну форму во всех лицах и числах. Исключением будет лишь have to, "  
 "который принимает форму has после местоимений he, she, it.\n3. Модальные "  
 "глаголы всегда стоят в настоящем времени (даже в контексте будущего "  
 "времени), однако некоторые из них могут принимать прошедшую форму: can — "  
 "could; may — might, must — had to.\n4. В отрицательных и вопросительных "  
 "предложениях модальные глаголы не употребляются со вспомогательными, "  
 "за исключением have to.\nI can’t do it. — Я не могу этого сделать.\nMay I "  
 "use your laptop? — Могу я воспользоваться вашим ноутбуком.\nWhat do I have "  
 "to sign? — Что я должен подписать?"**)  
 **elif** message.text == **"Инфинитив"**:  
 make\_keyboard\_infinitive(message)  
 return\_word = **"Список тем"  
 elif** message.text == **"Задание на \"Инфинитив\""**:  
 bot.send\_message(message.chat.id, **"Инфинитив"**, reply\_markup=types.ReplyKeyboardRemove())  
 state = state\_bot.State\_bot.command\_Infinitive  
 unidefeined\_commands(message)  
 **elif** message.text == **"Правило для темы \"Инфинитив\""**:  
 bot.send\_message(message.chat.id, **'https://www.native-english.ru/grammar/infinitive'**, disable\_web\_page\_preview=**True**)  
 **elif** message.text == **"Артикли"**:  
 make\_keyboard\_articles(message)  
 return\_word = **"Список тем"  
 elif** message.text == **"Задание на \"Артикли\""**:  
 bot.send\_message(message.chat.id, **"Артикли"**, reply\_markup=types.ReplyKeyboardRemove())  
 state = state\_bot.State\_bot.command\_Articles  
 unidefeined\_commands(message)  
 **elif** message.text == **"Правило для темы \"Артикли\""**:  
 bot.send\_message(message.chat.id, **'https://englex.ru/using-articles-in-english/'**, disable\_web\_page\_preview=**True**)  
 **elif** message.text == **"Герундий"**:  
 make\_keyboard\_gerund(message)  
 return\_word = **"Список тем"  
 elif** message.text == **"Задание на \"Герундий\""**:  
 bot.send\_message(message.chat.id, **"Герундий"**, reply\_markup=types.ReplyKeyboardRemove())  
 state = state\_bot.State\_bot.command\_Gerund  
 unidefeined\_commands(message)  
 **elif** message.text == **"Правило для темы \"Герундий\""**:  
 bot.send\_message(message.chat.id, **"https://englex.ru/gerund/"**, disable\_web\_page\_preview=**True**)  
 **elif** message.text == **"Множественное число"**:  
 make\_keyboard\_plural(message)  
 return\_word = **"Список тем"  
 elif** message.text == **"Задание на \"Множественное число\""**:  
 bot.send\_message(message.chat.id, **"Множественное число"**, reply\_markup=types.ReplyKeyboardRemove())  
 state = state\_bot.State\_bot.command\_Plural  
 unidefeined\_commands(message)  
 **elif** message.text == **"Правило для темы \"Множественное число\""**:  
 bot.send\_message(message.chat.id, **"https://skyeng.ru/articles/kak-obrazovat-mnozhestvennoe-chislo-v-anglijskom/"**, disable\_web\_page\_preview=**True**)  
 **elif** message.text == **"Правила"**:  
 bot.send\_message(message.chat.id, **"Правила"**)  
 bot.send\_message(message.chat.id, **"В этом разделе представлен комплекс упражнений, помогающий в изучении и "  
 "повторениитем Английского языка.\nУпражнения разделены по темам, "  
 "темы же могут быть разделены на подтемы.\nЗа каждый верный ответ в "  
 "упражнение будет начислено 50 очков.\n\nПожалуйста пишите ответы с "  
 "маленькой буквы, иначе бот засчитает ответ как неправильный."**)  
 make\_keyboard\_topic\_selection(message)  
 **elif** message.text == **"Назад"**:  
 return\_back(message)  
 **elif** message.text == **"---Меню---"**:  
 bot.send\_message(message.chat.id, **'Меню'**)  
 make\_keyboard\_menu(message)  
 **else**:  
 info = cursor.execute(**'SELECT \* FROM users WHERE user\_id=?'**, (message.from\_user.id,))  
 **if** info.fetchone() **is None**:  
 bot.send\_message(message.chat.id, **'Вы добавлены в бд'**)  
 bot.send\_message(message.chat.id, **f'Привет {**message.text**}'**)  
  
 us\_id = message.from\_user.id  
 us\_name = message.from\_user.first\_name  
 us\_sname = message.from\_user.last\_name  
 username = message.text  
  
 db\_table\_val(user\_id=us\_id, user\_name=us\_name, user\_surname=us\_sname, username=username, count=0)  
 bot.send\_message(message.chat.id, **'Меню'**)  
 make\_keyboard\_menu(message)  
 **else**:  
 unidefeined\_commands(message)  
 count += 1  
  
  
*#метод который смотрит на слово возврата и возвращает к нужному "меню" или "подменю"***def** return\_back(message):  
 **global** return\_word  
 bot.send\_message(message.chat.id, **f"{**return\_word**}"**)  
 **if** return\_word == **"Меню"**:  
 make\_keyboard\_menu(message)  
 **elif** return\_word == **"Список тем"**:  
 return\_word = **'Меню'** make\_keyboard\_topic\_selection(message)  
 **elif** return\_word == **"Времена"**:  
 return\_word = **'Список тем'** make\_keyboard\_time(message)  
  
  
*#метод обработки всех неизвестных комманд, сделан в о сновном, чтобы по состоянию бота выбрвть соответствующее задание***def** unidefeined\_commands(message):  
 **global** progress\_task  
 **global** word\_to\_task  
 **global** state  
 **global** task\_avability  
 **if** state == state\_bot.State\_bot.command\_not\_defined:  
 bot.send\_message(message.chat.id, **"Команда неизвестна"**)  
 **elif** state == state\_bot.State\_bot.command\_dictionary\_one:  
 task\_completion(message)  
 **elif** state == state\_bot.State\_bot.command\_dictionary\_two:  
 task\_completion(message)  
 **elif** state == state\_bot.State\_bot.command\_dictionary\_three:  
 task\_completion(message)  
 **elif** state == state\_bot.State\_bot.command\_Present\_Simple:  
 task\_completion(message)  
 **elif** state == state\_bot.State\_bot.command\_Past\_Simple:  
 task\_completion(message)  
 **elif** state == state\_bot.State\_bot.command\_Present\_Continuous:  
 task\_completion(message)  
 **elif** state == state\_bot.State\_bot.command\_Past\_Continuous:  
 task\_completion(message)  
 **elif** state == state\_bot.State\_bot.command\_Present\_Perfect:  
 task\_completion(message)  
 **elif** state == state\_bot.State\_bot.command\_Past\_Perfect:  
 task\_completion(message)  
 **elif** state == state\_bot.State\_bot.command\_Modal\_Verbs:  
 task\_completion(message)  
 **elif** state == state\_bot.State\_bot.command\_Infinitive:  
 task\_completion(message)  
 **elif** state == state\_bot.State\_bot.command\_Articles:  
 task\_completion(message)  
 **elif** state == state\_bot.State\_bot.command\_Gerund:  
 task\_completion(message)  
 **elif** state == state\_bot.State\_bot.command\_Plural:  
 task\_completion(message)  
  
*#вбор соответствующего задания, и проверка выполнненых***def** task\_completion(message):  
 **global** progress\_task  
 **global** word\_to\_task  
 **global** state  
 **global** task\_avability  
 **if** progress\_task < 5 **and** task\_avability == 0:  
 **if** state == state\_bot.State\_bot.command\_dictionary\_one:  
 choice\_task\_from\_dictionary\_first()  
 **elif** state == state\_bot.State\_bot.command\_dictionary\_two:  
 choice\_task\_from\_dictionary\_second()  
 **elif** state == state\_bot.State\_bot.command\_dictionary\_three:  
 choice\_task\_from\_dictionary\_third()  
 **elif** state == state\_bot.State\_bot.command\_Present\_Simple:  
 choice\_task\_from\_Present\_Simple()  
 **elif** state == state\_bot.State\_bot.command\_Past\_Simple:  
 choice\_task\_from\_Past\_Simple()  
 **elif** state == state\_bot.State\_bot.command\_Present\_Continuous:  
 choice\_task\_from\_Present\_Continuous()  
 **elif** state == state\_bot.State\_bot.command\_Past\_Continuous:  
 choice\_task\_from\_Past\_Continuous()  
 **elif** state == state\_bot.State\_bot.command\_Present\_Perfect:  
 choice\_task\_from\_Present\_Perfect()  
 **elif** state == state\_bot.State\_bot.command\_Past\_Perfect:  
 choice\_task\_from\_Past\_Perfect()  
 **elif** state == state\_bot.State\_bot.command\_Modal\_Verbs:  
 choice\_task\_from\_modal\_verbs()  
 **elif** state == state\_bot.State\_bot.command\_Infinitive:  
 choice\_task\_from\_Infinitive()  
 **elif** state == state\_bot.State\_bot.command\_Articles:  
 choice\_task\_from\_Articles()  
 **elif** state == state\_bot.State\_bot.command\_Gerund:  
 choice\_task\_from\_Gerund()  
 **elif** state == state\_bot.State\_bot.command\_Plural:  
 choice\_task\_from\_Plural()  
 task\_text\_output(message)  
 **if** task\_avability == 1:  
 task\_avability = 0  
 progress\_task += 1  
 check\_answer(message)  
 unidefeined\_commands(message)  
 **if** progress\_task == 5:  
 state = state\_bot.State\_bot.command\_not\_defined  
 progress\_task += 1  
 bot.send\_message(message.chat.id, **'---Меню---'**)  
 make\_keyboard\_menu(message)  
 task\_avability = 1  
  
  
*#метод выдачи задания***def** task\_text\_output(message):  
 **if** state == state\_bot.State\_bot.command\_dictionary\_one **or** state == state\_bot.State\_bot.command\_dictionary\_two **or** state == state\_bot.State\_bot.command\_dictionary\_three:  
 bot.send\_message(message.chat.id, **f"\nПереведите слово:\n\n {**word\_to\_task**}"**)  
 **elif** state == state\_bot.State\_bot.command\_Present\_Simple:  
 bot.send\_message(message.chat.id, **f"\nИспользуйте слова в скобках для образования предложений в Present Simple:\n\n {**word\_to\_task**}"**)  
 **elif** state == state\_bot.State\_bot.command\_Past\_Simple:  
 bot.send\_message(message.chat.id, **f"\nИспользуйте слова в скобках для образования предложений в Past Simple:\n\n {**word\_to\_task**}"**)  
 **elif** state == state\_bot.State\_bot.command\_Present\_Continuous:  
 bot.send\_message(message.chat.id, **f"\nИспользуйте слова в скобках для образования предложений в Present Continuous:\n\n {**word\_to\_task**}"**)  
 **elif** state == state\_bot.State\_bot.command\_Past\_Continuous:  
 bot.send\_message(message.chat.id, **f"\nИспользуйте слова в скобках для образования предложений в Past Continuous:\n\n {**word\_to\_task**}"**)  
 **elif** state == state\_bot.State\_bot.command\_Present\_Perfect:  
 bot.send\_message(message.chat.id, **f"\nИспользуйте слова в скобках для образования предложений в Present Perfect:\n\n {**word\_to\_task**}"**)  
 **elif** state == state\_bot.State\_bot.command\_Past\_Perfect:  
 bot.send\_message(message.chat.id, **f"\nИспользуйте слова в скобках для образования предложений в Past Perfect:\n\n {**word\_to\_task**}"**)  
 **elif** state == state\_bot.State\_bot.command\_Modal\_Verbs:  
 bot.send\_message(message.chat.id, **f"\nВставьте подходящий модальный глагол:\n\n {**word\_to\_task**}"**)  
 **elif** state == state\_bot.State\_bot.command\_Infinitive:  
 bot.send\_message(message.chat.id, **f"\nВставьте подходящий модальный глагол:\n\n {**word\_to\_task**}"**)  
 **elif** state == state\_bot.State\_bot.command\_Articles:  
 bot.send\_message(message.chat.id, **f"\nВставьте подходящий артикли:\n\n {**word\_to\_task**}"**)  
 **elif** state == state\_bot.State\_bot.command\_Gerund:  
 bot.send\_message(message.chat.id, **f"\nРаскройте скобки, поставив глагол в форму герундия:\n\n {**word\_to\_task**}"**)  
 **elif** state == state\_bot.State\_bot.command\_Plural:  
 bot.send\_message(message.chat.id, **f"\nЗапишите данные слова во множественном числе:\n\n {**word\_to\_task**}"**)  
  
  
*#обнуление переменных***def** zeroing\_variables():  
 **global** task\_avability  
 **global** progress\_task  
 task\_avability = 0  
 progress\_task = 0  
  
  
*#случайный выбор задания из базы данных***def** choice\_task\_from\_dictionary\_first():  
 **global** word\_to\_task  
 **global** answer\_to\_task  
 random\_id = random.randint(0, 10)  
 info = cursor.execute(**'SELECT \* FROM dictionary\_first\_level WHERE id=?'**, (random\_id,))  
 record = info.fetchone()  
 word\_to\_task = **f"{**record[1]**}"** answer\_to\_task = **f"{**record[2]**}"  
  
  
def** choice\_task\_from\_dictionary\_second():  
 **global** word\_to\_task  
 **global** answer\_to\_task  
 word\_to\_task = **""** random\_id = random.randint(0, 10)  
 info = cursor.execute(**'SELECT \* FROM dictionary\_second\_level WHERE id=?'**, (random\_id,))  
 record = info.fetchone()  
 word\_to\_task = **f"{**record[1]**}"** answer\_to\_task = **f"{**record[2]**}"  
  
  
def** choice\_task\_from\_dictionary\_third():  
 **global** word\_to\_task  
 **global** answer\_to\_task  
 random\_id = random.randint(0, 10)  
 info = cursor.execute(**'SELECT \* FROM dictionary\_third\_level WHERE id=?'**, (random\_id,))  
 record = info.fetchone()  
 word\_to\_task = **f"{**record[1]**}"** answer\_to\_task = **f"{**record[2]**}"  
  
  
def** choice\_task\_from\_Present\_Simple():  
 **global** word\_to\_task  
 **global** answer\_to\_task  
 random\_id = random.randint(0, 10)  
 info = cursor.execute(**'SELECT \* FROM present\_simple WHERE id=?'**, (random\_id, ))  
 record = info.fetchone()  
 word\_to\_task = **f"{**record[1]**}"** answer\_to\_task = **f"{**record[2]**}"  
  
  
def** choice\_task\_from\_Past\_Simple():  
 **global** word\_to\_task  
 **global** answer\_to\_task  
 random\_id = random.randint(0, 10)  
 info = cursor.execute(**'SELECT \* FROM past\_simple WHERE id=?'**, (random\_id, ))  
 record = info.fetchone()  
 word\_to\_task = **f"{**record[1]**}"** answer\_to\_task = **f"{**record[2]**}"  
  
  
def** choice\_task\_from\_Present\_Continuous():  
 **global** word\_to\_task  
 **global** answer\_to\_task  
 random\_id = random.randint(0, 10)  
 info = cursor.execute(**'SELECT \* FROM present\_continuous WHERE id=?'**, (random\_id, ))  
 record = info.fetchone()  
 word\_to\_task = **f"{**record[1]**}"** answer\_to\_task = **f"{**record[2]**}"  
  
  
def** choice\_task\_from\_Past\_Continuous():  
 **global** word\_to\_task  
 **global** answer\_to\_task  
 random\_id = random.randint(0, 10)  
 info = cursor.execute(**'SELECT \* FROM past\_continuous WHERE id=?'**, (random\_id, ))  
 record = info.fetchone()  
 word\_to\_task = **f"{**record[1]**}"** answer\_to\_task = **f"{**record[2]**}"  
  
  
def** choice\_task\_from\_Present\_Perfect():  
 **global** word\_to\_task  
 **global** answer\_to\_task  
 random\_id = random.randint(0, 10)  
 info = cursor.execute(**'SELECT \* FROM present\_perfect WHERE id=?'**, (random\_id, ))  
 record = info.fetchone()  
 word\_to\_task = **f"{**record[1]**}"** answer\_to\_task = **f"{**record[2]**}"  
  
  
def** choice\_task\_from\_Past\_Perfect():  
 **global** word\_to\_task  
 **global** answer\_to\_task  
 random\_id = random.randint(0, 10)  
 info = cursor.execute(**'SELECT \* FROM past\_perfect WHERE id=?'**, (random\_id, ))  
 record = info.fetchone()  
 word\_to\_task = **f"{**record[1]**}"** answer\_to\_task = **f"{**record[2]**}"  
  
  
def** choice\_task\_from\_modal\_verbs():  
 **global** word\_to\_task  
 **global** answer\_to\_task  
 random\_id = random.randint(0, 10)  
 info = cursor.execute(**'SELECT \* FROM modal\_varbs WHERE id=?'**, (random\_id, ))  
 record = info.fetchone()  
 word\_to\_task = **f"{**record[1]**}"** answer\_to\_task = **f"{**record[2]**}"  
  
  
def** choice\_task\_from\_Infinitive():  
 **global** word\_to\_task  
 **global** answer\_to\_task  
 random\_id = random.randint(0, 10)  
 info = cursor.execute(**'SELECT \* FROM infinitive WHERE id=?'**, (random\_id, ))  
 record = info.fetchone()  
 word\_to\_task = **f"{**record[1]**}"** answer\_to\_task = **f"{**record[2]**}"  
  
  
def** choice\_task\_from\_Articles():  
 **global** word\_to\_task  
 **global** answer\_to\_task  
 random\_id = random.randint(0, 10)  
 info = cursor.execute(**'SELECT \* FROM articles WHERE id=?'**, (random\_id, ))  
 record = info.fetchone()  
 word\_to\_task = **f"{**record[1]**}"** answer\_to\_task = **f"{**record[2]**}"  
  
  
def** choice\_task\_from\_Gerund():  
 **global** word\_to\_task  
 **global** answer\_to\_task  
 random\_id = random.randint(0, 10)  
 info = cursor.execute(**'SELECT \* FROM gerund WHERE id=?'**, (random\_id, ))  
 record = info.fetchone()  
 word\_to\_task = **f"{**record[1]**}"** answer\_to\_task = **f"{**record[2]**}"  
  
  
def** choice\_task\_from\_Plural():  
 **global** word\_to\_task  
 **global** answer\_to\_task  
 random\_id = random.randint(0, 10)  
 info = cursor.execute(**'SELECT \* FROM plural WHERE id=?'**, (random\_id, ))  
 record = info.fetchone()  
 word\_to\_task = **f"{**record[1]**}"** answer\_to\_task = **f"{**record[2]**}"***#проверка ответа***def** check\_answer(message):  
 **if** answer\_to\_task == message.text:  
 bot.send\_message(message.chat.id, **"\_\_Ответ верен\_\_"**)  
 **if** state == state\_bot.State\_bot.command\_dictionary\_one:  
 cursor.execute(**'UPDATE users SET count\_user = count\_user + 50 WHERE user\_id = ?'**, (message.from\_user.id,))  
 **elif** state == state\_bot.State\_bot.command\_dictionary\_two:  
 cursor.execute(**'UPDATE users SET count\_user = count\_user + 70 WHERE user\_id = ?'**, (message.from\_user.id,))  
 **elif** state == state\_bot.State\_bot.command\_dictionary\_three:  
 cursor.execute(**'UPDATE users SET count\_user = count\_user + 90 WHERE user\_id = ?'**, (message.from\_user.id,))  
 **elif** state == state\_bot.State\_bot.command\_Present\_Simple **or** state == state\_bot.State\_bot.command\_Past\_Simple **or** state == state\_bot.State\_bot.command\_Present\_Continuous **or** state\_bot.State\_bot.command\_Past\_Continuous **or** state\_bot.State\_bot.command\_Present\_Perfect **or** state\_bot.State\_bot.command\_Past\_Perfect **or** state\_bot.State\_bot.command\_Modal\_Verbs **or** state\_bot.State\_bot.command\_Infinitive **or** state\_bot.State\_bot.command\_Articles **or** state\_bot.State\_bot.command\_Gerund **or** state\_bot.State\_bot.command\_Plural:  
 cursor.execute(**'UPDATE users SET count\_user = count\_user + 100 WHERE user\_id = ?'**, (message.from\_user.id,))  
 **else**:  
 bot.send\_message(message.chat.id, **f"\_\_Ответ не верен\_\_\n\n Верный ответ: {**answer\_to\_task**}"**)  
  
  
*#создание клавиатур***def** make\_keyboard\_menu(message):  
 markup = types.ReplyKeyboardMarkup(resize\_keyboard=**True**)  
 button\_1 = types.KeyboardButton(text=**"Список тем"**)  
 button\_2 = types.KeyboardButton(text=**"Работа со словарём"**)  
 button\_3 = types.KeyboardButton(text=**"Таблица лидеров"**, )  
 markup.add(button\_1, button\_2, button\_3)  
 bot.send\_message(message.chat.id, text=**"Выбери пункт меню:"**, reply\_markup=markup)  
  
  
**def** make\_keyboard\_table\_liders(message):  
 keybord = types.ReplyKeyboardMarkup(resize\_keyboard=**True**)  
 button\_1 = types.KeyboardButton(text=**"Вывести лучшую десятку"**)  
 button\_2 = types.KeyboardButton(text=**"Вывести своё место"**)  
 button\_3 = types.KeyboardButton(text=**"Назад"**)  
 keybord.add(button\_1, button\_2, button\_3)  
 bot.send\_message(message.chat.id, text=**"Выбери пункт:"**, reply\_markup=keybord)  
  
  
**def** make\_keyboard\_dictionary(message):  
 keybord = types.ReplyKeyboardMarkup(resize\_keyboard=**True**)  
 button\_1 = types.KeyboardButton(text=**"1 уровень сложности"**)  
 button\_2 = types.KeyboardButton(text=**"2 уровень сложности"**)  
 button\_3 = types.KeyboardButton(text=**"3 уровень сложности"**)  
 button\_4 = types.KeyboardButton(text=**"Правила режима \"Работа со словарём\""**)  
 button\_5 = types.KeyboardButton(text=**"Назад"**)  
 keybord.add(button\_1, button\_2, button\_3, button\_4, button\_5)  
 bot.send\_message(message.chat.id, text=**"Выбери пункт:"**, reply\_markup=keybord)  
  
  
**def** make\_keyboard\_topic\_selection(message):  
 keybord = types.ReplyKeyboardMarkup(resize\_keyboard=**True**)  
 button\_1 = types.KeyboardButton(text=**"Времена"**)  
 button\_2 = types.KeyboardButton(text=**"Модальные глаголы"**)  
 button\_3 = types.KeyboardButton(text=**"Инфинитив"**)  
 button\_4 = types.KeyboardButton(text=**"Артикли"**)  
 button\_5 = types.KeyboardButton(text=**"Герундий"**)  
 button\_6 = types.KeyboardButton(text=**"Множественное число"**)  
 button\_7 = types.KeyboardButton(text=**"Правила"**)  
 button\_8 = types.KeyboardButton(text=**"Назад"**)  
 keybord.add(button\_1, button\_2, button\_3, button\_4, button\_5, button\_6, button\_7, button\_8)  
 bot.send\_message(message.chat.id, text=**"Выбери пункт:"**, reply\_markup=keybord)  
  
  
**def** make\_keyboard\_modal\_verbs(message):  
 keybord = types.ReplyKeyboardMarkup(resize\_keyboard=**True**)  
 button\_1 = types.KeyboardButton(text=**"Задание на \"Модальные глаголы\""**)  
 button\_2 = types.KeyboardButton(text=**"Правило для темы \"Модальные глаголы\""**)  
 button\_3 = types.KeyboardButton(text=**"Назад"**)  
 keybord.add(button\_1, button\_2, button\_3)  
 bot.send\_message(message.chat.id, text=**"Выбери пункт:"**, reply\_markup=keybord)  
  
  
**def** make\_keyboard\_articles(message):  
 keybord = types.ReplyKeyboardMarkup(resize\_keyboard=**True**)  
 button\_1 = types.KeyboardButton(text=**"Задание на \"Артикли\""**)  
 button\_2 = types.KeyboardButton(text=**"Правило для темы \"Артикли\""**)  
 button\_3 = types.KeyboardButton(text=**"Назад"**)  
 keybord.add(button\_1, button\_2, button\_3)  
 bot.send\_message(message.chat.id, text=**"Выбери пункт:"**, reply\_markup=keybord)  
  
  
**def** make\_keyboard\_infinitive(message):  
 keybord = types.ReplyKeyboardMarkup(resize\_keyboard=**True**)  
 button\_1 = types.KeyboardButton(text=**"Задание на \"Инфинитив\""**)  
 button\_2 = types.KeyboardButton(text=**"Правило для темы \"Инфинитив\""**)  
 button\_3 = types.KeyboardButton(text=**"Назад"**)  
 keybord.add(button\_1, button\_2, button\_3)  
 bot.send\_message(message.chat.id, text=**"Выбери пункт:"**, reply\_markup=keybord)  
  
  
**def** make\_keyboard\_gerund(message):  
 keybord = types.ReplyKeyboardMarkup(resize\_keyboard=**True**)  
 button\_1 = types.KeyboardButton(text=**"Задание на \"Герундий\""**)  
 button\_2 = types.KeyboardButton(text=**"Правило для темы \"Герундий\""**)  
 button\_3 = types.KeyboardButton(text=**"Назад"**)  
 keybord.add(button\_1, button\_2, button\_3)  
 bot.send\_message(message.chat.id, text=**"Выбери пункт:"**, reply\_markup=keybord)  
  
  
**def** make\_keyboard\_plural(message):  
 keybord = types.ReplyKeyboardMarkup(resize\_keyboard=**True**)  
 button\_1 = types.KeyboardButton(text=**"Задание на \"Множественное число\""**)  
 button\_2 = types.KeyboardButton(text=**"Правило для темы \"Множественное число\""**)  
 button\_3 = types.KeyboardButton(text=**"Назад"**)  
 keybord.add(button\_1, button\_2, button\_3)  
 bot.send\_message(message.chat.id, text=**"Выбери пункт:"**, reply\_markup=keybord)  
  
  
**def** make\_keyboard\_present\_simple(message):  
 keybord = types.ReplyKeyboardMarkup(resize\_keyboard=**True**)  
 button\_1 = types.KeyboardButton(text=**"Задание на \"Present Simple\""**)  
 button\_2 = types.KeyboardButton(text=**"Правило для темы \"Present Simple\""**)  
 button\_3 = types.KeyboardButton(text=**"Назад"**)  
 keybord.add(button\_1, button\_2, button\_3)  
 bot.send\_message(message.chat.id, text=**"Выбери пункт:"**, reply\_markup=keybord)  
  
  
**def** make\_keyboard\_past\_simple(message):  
 keybord = types.ReplyKeyboardMarkup(resize\_keyboard=**True**)  
 button\_1 = types.KeyboardButton(text=**"Задание на \"Past Simple\""**)  
 button\_2 = types.KeyboardButton(text=**"Правило для темы \"Past Simple\""**)  
 button\_3 = types.KeyboardButton(text=**"Назад"**)  
 keybord.add(button\_1, button\_2, button\_3)  
 bot.send\_message(message.chat.id, text=**"Выбери пункт:"**, reply\_markup=keybord)  
  
  
**def** make\_keyboard\_present\_continuous(message):  
 keybord = types.ReplyKeyboardMarkup(resize\_keyboard=**True**)  
 button\_1 = types.KeyboardButton(text=**"Задание на \"Present Continuous\""**)  
 button\_2 = types.KeyboardButton(text=**"Правило для темы \"Present Continuous\""**)  
 button\_3 = types.KeyboardButton(text=**"Назад"**)  
 keybord.add(button\_1, button\_2, button\_3)  
 bot.send\_message(message.chat.id, text=**"Выбери пункт:"**, reply\_markup=keybord)  
  
  
**def** make\_keyboard\_past\_continuous(message):  
 keybord = types.ReplyKeyboardMarkup(resize\_keyboard=**True**)  
 button\_1 = types.KeyboardButton(text=**"Задание на \"Past Continuous\""**)  
 button\_2 = types.KeyboardButton(text=**"Правило для темы \"Past Continuous\""**)  
 button\_3 = types.KeyboardButton(text=**"Назад"**)  
 keybord.add(button\_1, button\_2, button\_3)  
 bot.send\_message(message.chat.id, text=**"Выбери пункт:"**, reply\_markup=keybord)  
  
  
**def** make\_keyboard\_present\_perfect(message):  
 keybord = types.ReplyKeyboardMarkup(resize\_keyboard=**True**)  
 button\_1 = types.KeyboardButton(text=**"Задание на \"Present Perfect\""**)  
 button\_2 = types.KeyboardButton(text=**"Правило для темы \"Present Perfect\""**)  
 button\_3 = types.KeyboardButton(text=**"Назад"**)  
 keybord.add(button\_1, button\_2, button\_3)  
 bot.send\_message(message.chat.id, text=**"Выбери пункт:"**, reply\_markup=keybord)  
  
  
**def** make\_keyboard\_past\_perfect(message):  
 keybord = types.ReplyKeyboardMarkup(resize\_keyboard=**True**)  
 button\_1 = types.KeyboardButton(text=**"Задание на \"Past Perfect\""**)  
 button\_2 = types.KeyboardButton(text=**"Правило для темы \"Past Perfect\""**)  
 button\_3 = types.KeyboardButton(text=**"Назад"**)  
 keybord.add(button\_1, button\_2, button\_3)  
 bot.send\_message(message.chat.id, text=**"Выбери пункт:"**, reply\_markup=keybord)  
  
  
**def** make\_keyboard\_time(message):  
 keybord = types.ReplyKeyboardMarkup(resize\_keyboard=**True**)  
 button\_1 = types.KeyboardButton(text=**"Present Simple"**)  
 button\_2 = types.KeyboardButton(text=**"Past Simple"**)  
 button\_3 = types.KeyboardButton(text=**"Present Continuous"**)  
 button\_4 = types.KeyboardButton(text=**"Past Continuous"**)  
 button\_5 = types.KeyboardButton(text=**"Present Perfect"**)  
 button\_6 = types.KeyboardButton(text=**"Past Perfect"**)  
 button\_7 = types.KeyboardButton(text=**"Правила"**)  
 button\_8 = types.KeyboardButton(text=**"Назад"**)  
 keybord.add(button\_1, button\_2, button\_3, button\_4, button\_5, button\_6, button\_7,button\_8)  
 bot.send\_message(message.chat.id, text=**"Выбери пункт:"**, reply\_markup=keybord)  
  
  
*#сортировка бд и вывод запроса***def** conclusion\_user\_position(message):  
 count = 1  
 output\_line = **"-----Ваше место-----\n"** cursor.execute(**'SELECT \* FROM users ORDER BY count\_user DESC'**)  
 requests\_top\_users = cursor.fetchall()  
 **for** row **in** requests\_top\_users:  
 **if** row[0] == message.from\_user.id:  
 output\_line += **f" {**count**}. "** + str(row[3]) + **" "** + str(row[4]) + **"\n"** bot.send\_message(message.chat.id, output\_line)  
 count += 1  
  
  
**def** conclusion\_top\_ten\_users(message):  
 cursor.execute(**'SELECT \* FROM users ORDER BY count\_user DESC'**)  
 records = cursor.fetchall()  
 i = 1  
 output\_line = **"-----Таблица лидеров-----\n"  
 for** row **in** records:  
 **if** i <= 10:  
 output\_line += **f" {**i**}. "** + str(row[3]) + **" "** + str(row[4]) + **"\n"** i += 1  
 bot.send\_message(message.chat.id, output\_line)  
  
  
bot.polling(none\_stop=**True**)

ПРИЛОЖЕНИЕ Б

**Код базы данных программы**

-- Таблица: articles

— Таблица: articles  
CREATE TABLE articles (id INTEGER PRIMARY KEY UNIQUE NOT NULL, task TEXT NOT NULL, answer TEXT NOT NULL);  
INSERT INTO articles (id, task, answer) VALUES (1, '…Smiths have a dog and a cat. a) … b) The c) A', 'b');  
INSERT INTO articles (id, task, answer) VALUES (2, 'He knows how to work on … computer. a) a b) an c) …', 'a');  
INSERT INTO articles (id, task, answer) VALUES (3, 'She was the first woman to swim across … English Channel. a) a b) … c) the', 'c');  
INSERT INTO articles (id, task, answer) VALUES (4, 'Go down … Kingston Street and turn left into Oxford Street. a) the b) a c) …', 'c');  
INSERT INTO articles (id, task, answer) VALUES (5, 'I don’t like milk in … tea. a) … b) the с) а', 'a');  
INSERT INTO articles (id, task, answer) VALUES (6, 'At the end of… busy day, sleep is the best way to restore your energy. a) the b) a c) …', 'b');  
INSERT INTO articles (id, task, answer) VALUES (7, 'We’ll go for a walk if … weather is fine. a) a b) … c) the', 'c');  
INSERT INTO articles (id, task, answer) VALUES (8, 'Could you give me … information I asked for in my letter? a) the b) … c) a', 'a');  
INSERT INTO articles (id, task, answer) VALUES (9, 'I spent … very interesting holiday in England. a) the b) a c) …', 'b');  
INSERT INTO articles (id, task, answer) VALUES (10, '…war is a terrible thing. a) The b) … с) А', 'b');

-- Таблица: dictionary\_first\_level

CREATE TABLE dictionary\_first\_level (id INTEGER PRIMARY KEY UNIQUE NOT NULL, word TEXT NOT NULL, translation TEXT NOT NULL);

INSERT INTO dictionary\_first\_level (id, word, translation) VALUES (1, 'read', 'читать');

INSERT INTO dictionary\_first\_level (id, word, translation) VALUES (2, 'father', 'папа');

INSERT INTO dictionary\_first\_level (id, word, translation) VALUES (3, 'mother', 'мама');

INSERT INTO dictionary\_first\_level (id, word, translation) VALUES (4, 'red', 'красный');

INSERT INTO dictionary\_first\_level (id, word, translation) VALUES (5, 'apple', 'яблоко');

INSERT INTO dictionary\_first\_level (id, word, translation) VALUES (6, 'head', 'голова');

INSERT INTO dictionary\_first\_level (id, word, translation) VALUES (7, 'dog', 'собака');

INSERT INTO dictionary\_first\_level (id, word, translation) VALUES (8, 'cat', 'кот');

INSERT INTO dictionary\_first\_level (id, word, translation) VALUES (9, 'pig', 'свинья');

INSERT INTO dictionary\_first\_level (id, word, translation) VALUES (10, 'wolf', 'волк');

-- Таблица: dictionary\_second\_level

CREATE TABLE dictionary\_second\_level (id INTEGER PRIMARY KEY UNIQUE NOT NULL, word TEXT NOT NULL, translation TEXT NOT NULL);

INSERT INTO dictionary\_second\_level (id, word, translation) VALUES (1, 'sway', 'колебание');

INSERT INTO dictionary\_second\_level (id, word, translation) VALUES (2, 'to invent', 'изобретать');

INSERT INTO dictionary\_second\_level (id, word, translation) VALUES (3, 'district', 'район');

INSERT INTO dictionary\_second\_level (id, word, translation) VALUES (4, 'devotion', 'преданность');

INSERT INTO dictionary\_second\_level (id, word, translation) VALUES (5, 'isolation', 'изоляция');

INSERT INTO dictionary\_second\_level (id, word, translation) VALUES (6, 'verefication', 'проверка');

INSERT INTO dictionary\_second\_level (id, word, translation) VALUES (7, 'emotion', 'эмоция');

INSERT INTO dictionary\_second\_level (id, word, translation) VALUES (8, 'bad luck', 'неудача');

INSERT INTO dictionary\_second\_level (id, word, translation) VALUES (9, 'luck', 'удача');

INSERT INTO dictionary\_second\_level (id, word, translation) VALUES (10, 'condition', 'условие');

-- Таблица: dictionary\_third\_level

CREATE TABLE dictionary\_third\_level (id PRIMARY KEY UNIQUE NOT NULL, word TEXT NOT NULL, translation TEXT NOT NULL);

INSERT INTO dictionary\_third\_level (id, word, translation) VALUES (1, 'performance', 'выступление');

INSERT INTO dictionary\_third\_level (id, word, translation) VALUES (2, 'opportunity', 'возможность');

INSERT INTO dictionary\_third\_level (id, word, translation) VALUES (3, 'success', 'успех');

INSERT INTO dictionary\_third\_level (id, word, translation) VALUES (4, 'illegitimate', 'противозаконный');

INSERT INTO dictionary\_third\_level (id, word, translation) VALUES (5, 'clothes', 'одежда');

INSERT INTO dictionary\_third\_level (id, word, translation) VALUES (6, 'review', 'рецензия');

INSERT INTO dictionary\_third\_level (id, word, translation) VALUES (7, 'withdraw', 'снимать');

INSERT INTO dictionary\_third\_level (id, word, translation) VALUES (8, 'cheque', 'чек');

INSERT INTO dictionary\_third\_level (id, word, translation) VALUES (9, 'thistle', 'чертополох');

INSERT INTO dictionary\_third\_level (id, word, translation) VALUES (10, 'appointment', 'встреча');

-- Таблица: gerund

CREATE TABLE gerund (id INTEGER UNIQUE PRIMARY KEY NOT NULL, task TEXT NOT NULL, answer TEXT NOT NULL);

INSERT INTO gerund (id, task, answer) VALUES (1, 'The doctor recommended \_\_\_\_ a pill twice a day. (take)', 'taking');

INSERT INTO gerund (id, task, answer) VALUES (2, 'Sometimes I imagine \_\_\_\_ a lottery. (win)', 'winning');

INSERT INTO gerund (id, task, answer) VALUES (3, 'I think \_\_\_\_ by car is very convenient. (travel)', 'traveling');

INSERT INTO gerund (id, task, answer) VALUES (4, 'She avoids \_\_\_\_ fast food. (eat)', 'eating');

INSERT INTO gerund (id, task, answer) VALUES (5, 'They consider \_\_\_\_ the job. (change)', 'changing');

INSERT INTO gerund (id, task, answer) VALUES (6, 'He continues \_\_\_\_ that book I gave him. (read)', 'reading');

INSERT INTO gerund (id, task, answer) VALUES (7, 'We go \_\_\_\_ every morning. (jog)', 'jogging');

INSERT INTO gerund (id, task, answer) VALUES (8, 'She suggested \_\_\_\_ Paris at the weekend. (visit)', 'visiting');

INSERT INTO gerund (id, task, answer) VALUES (9, 'I can''t stand \_\_\_\_ in a line in the supermarket. (wait)', 'waiting');

INSERT INTO gerund (id, task, answer) VALUES (10, 'The film was so funny we couldn''t help \_\_\_\_ . (laugh)', 'laughing');

-- Таблица: infinitive

CREATE TABLE infinitive (id INTEGER PRIMARY KEY UNIQUE NOT NULL, task TEXT NOT NULL, answer TEXT NOT NULL);

INSERT INTO infinitive (id, task, answer) VALUES (1, 'He can (to run/run) \_\_\_\_  very fast.', 'run');

INSERT INTO infinitive (id, task, answer) VALUES (2, 'Let me (to help/help) \_\_\_\_  you do the housework.', 'help');

INSERT INTO infinitive (id, task, answer) VALUES (3, 'The teacher told Ann (to do/do) \_\_\_\_ the task again.', 'to do');

INSERT INTO infinitive (id, task, answer) VALUES (4, 'You always make me (to laugh/laugh) \_\_\_\_', 'laugh');

INSERT INTO infinitive (id, task, answer) VALUES (5, ' I would rather (to take/take) \_\_\_\_ a taxi to go to the city center.', 'take');

INSERT INTO infinitive (id, task, answer) VALUES (6, 'She prefers (to drive/driving) \_\_\_\_ rather on go by bus.', 'to drive');

INSERT INTO infinitive (id, task, answer) VALUES (7, 'You ought (to take/taking) care of your health.', 'to take');

INSERT INTO infinitive (id, task, answer) VALUES (8, 'I’d rather (to eat/eat) \_\_\_\_ at home than go to the cafe.', 'eat');

INSERT INTO infinitive (id, task, answer) VALUES (9, 'Do you want me (to go/go) \_\_\_\_ with you?', 'to go');

INSERT INTO infinitive (id, task, answer) VALUES (10, 'Everyone must (to wear/wear) \_\_\_\_ a mask in a public place. ', 'wear');

-- Таблица: modal\_verbs

CREATE TABLE modal\_verbs (id INTEGER PRIMARY KEY UNIQUE NOT NULL, task TEXT NOT NULL, answer TEXT NOT NULL);

INSERT INTO modal\_verbs (id, task, answer) VALUES (1, 'If you studied well, you \_\_\_\_\_\_\_\_\_\_\_\_ worry at the exam.', 'shouldn’t');

INSERT INTO modal\_verbs (id, task, answer) VALUES (2, 'When I was young, I \_\_\_\_\_\_\_\_\_\_\_\_ swim very fast.', 'could');

INSERT INTO modal\_verbs (id, task, answer) VALUES (3, 'It’s cold outside. You \_\_\_\_\_\_\_\_\_\_\_\_ wear a coat.', 'should');

INSERT INTO modal\_verbs (id, task, answer) VALUES (4, 'She \_\_\_\_\_\_\_\_\_\_\_\_ speak Italian because she lived in Italy 2 years ago.', 'can');

INSERT INTO modal\_verbs (id, task, answer) VALUES (5, 'If you feel bad, you \_\_\_\_\_\_\_\_\_\_\_\_ see a doctor.', 'should');

INSERT INTO modal\_verbs (id, task, answer) VALUES (6, 'You \_\_\_\_\_\_\_\_\_\_\_\_ be late for the lessons.', 'shouldn’t');

INSERT INTO modal\_verbs (id, task, answer) VALUES (7, 'I’m sorry, I \_\_\_\_\_\_\_\_\_\_\_\_ talk right now.', 'can’t');

INSERT INTO modal\_verbs (id, task, answer) VALUES (8, 'There isn’t much time. We \_\_\_\_\_\_\_\_\_\_\_\_ hurry.', 'must');

INSERT INTO modal\_verbs (id, task, answer) VALUES (9, 'I \_\_\_\_\_\_\_\_\_\_\_\_ carry this bag, it’s too heavy.', 'can’t');

INSERT INTO modal\_verbs (id, task, answer) VALUES (10, 'Lily \_\_\_\_\_\_\_\_\_\_\_\_ dance very well. She goes to a dancing school.', 'can');

-- Таблица: past\_continuous

CREATE TABLE past\_continuous (id INTEGER PRIMARY KEY UNIQUE NOT NULL, task TEXT NOT NULL, answer TEXT NOT NULL);

INSERT INTO past\_continuous (id, task, answer) VALUES (1, 'Around me people (to talk) Russian, Italian and English.', 'were talking');

INSERT INTO past\_continuous (id, task, answer) VALUES (2, 'Arnold (to talk) to some of the other guests on the terrace when Hardy came.', 'was talking');

INSERT INTO past\_continuous (id, task, answer) VALUES (3, 'Alex (to look) at his watch.', 'was looking');

INSERT INTO past\_continuous (id, task, answer) VALUES (4, 'All night long the stars (to glitter).', 'were glittering');

INSERT INTO past\_continuous (id, task, answer) VALUES (5, 'Elizabeth (to eat) and didn’t raise her head.', 'was eating');

INSERT INTO past\_continuous (id, task, answer) VALUES (6, 'He drank some of the wine and ate several chunks of bread while he (to wait) for his dinner to come up.', 'was waiting');

INSERT INTO past\_continuous (id, task, answer) VALUES (7, 'The family (to prepare) for the party.', 'was preparing');

INSERT INTO past\_continuous (id, task, answer) VALUES (8, 'She (to argue) that only Belinda knew how to treat men.', 'was arguing');

INSERT INTO past\_continuous (id, task, answer) VALUES (9, 'A few minutes later Edward (to hurry) through the streets to his bus stop.', 'was hurrying');

INSERT INTO past\_continuous (id, task, answer) VALUES (10, 'They moved across the room, which (to start) to fill up, to a vacant corner.', 'was starting');

-- Таблица: past\_perfect

CREATE TABLE past\_perfect (id INTEGER PRIMARY KEY UNIQUE NOT NULL, task TEXT NOT NULL, answer TEXT NOT NULL);

INSERT INTO past\_perfect (id, task, answer) VALUES (1, 'There \_\_\_\_ many apples on the ground. They \_\_\_\_ off the tree. (were, fell / were, had fallen / had been, fell)', 'were, had fallen');

INSERT INTO past\_perfect (id, task, answer) VALUES (2, 'Irene \_\_\_\_ to school yesterday. She \_\_\_\_ the test. (didn’t come, had already passed / hadn’t come, already passed / didn’t come, already passed)', 'didn’t come, had already passed');

INSERT INTO past\_perfect (id, task, answer) VALUES (3, 'Ben \_\_\_\_ home when I \_\_\_\_. (just got, had called / had just got, called / just got, called)', 'had just got, called');

INSERT INTO past\_perfect (id, task, answer) VALUES (4, 'They \_\_\_\_ hungry. They \_\_\_\_ lunch. (weren''t, had just had / hadn''t been, just had / weren''t, just had)', 'weren’t, had just had');

INSERT INTO past\_perfect (id, task, answer) VALUES (5, 'When I first \_\_\_\_ Jill, she \_\_\_\_ around the world. (had meet, already traveled / met, already traveled / met, had already traveled)', 'met, had already traveled');

INSERT INTO past\_perfect (id, task, answer) VALUES (6, 'Somebody \_\_\_ a song which I \_\_\_\_ before. (sang, never heard / sang, had never heard / had sung, never heard)', 'sang, had never heard');

INSERT INTO past\_perfect (id, task, answer) VALUES (7, 'When Gene \_\_\_\_ at the party, Lisa \_\_\_\_ home. (arrived, had already gone / had arrived, already went / arrived, already went)', 'arrived, had already gone');

INSERT INTO past\_perfect (id, task, answer) VALUES (8, 'The car \_\_\_\_ very dirty. Nobody \_\_\_\_ it for weeks. (was, cleaned / was, had cleaned / had been, cleaned)', 'was, had cleaned');

INSERT INTO past\_perfect (id, task, answer) VALUES (9, 'Andy \_\_\_\_ Claire yesterday. He \_\_\_\_ her in years. (saw, hadn’t seen / had seen, saw / saw, didn’t see)', 'saw, hadn’t seen');

INSERT INTO past\_perfect (id, task, answer) VALUES (10, 'Last year we \_\_\_\_ in the USA for the first time. We \_\_\_\_ there before. (were, have never been / had been, never were / were, had never been)', 'were, had never been');

-- Таблица: past\_simple

CREATE TABLE past\_simple (id INTEGER PRIMARY KEY UNIQUE NOT NULL, task TEXT NOT NULL, answer TEXT NOT NULL);

INSERT INTO past\_simple (id, task, answer) VALUES (1, 'I (to invite) \_\_ your friend to the party.', 'invited');

INSERT INTO past\_simple (id, task, answer) VALUES (2, 'Paul (to find) \_\_ a good and inexpensive hotel.', 'found');

INSERT INTO past\_simple (id, task, answer) VALUES (3, 'We (to understand) \_\_ each other.', 'understood');

INSERT INTO past\_simple (id, task, answer) VALUES (4, 'Did you (to see) \_\_ my new bike?', 'see');

INSERT INTO past\_simple (id, task, answer) VALUES (5, 'I (not to know) \_\_ this.', 'didn''t know');

INSERT INTO past\_simple (id, task, answer) VALUES (6, 'Why did you (to ask) \_\_ me this question?', 'ask');

INSERT INTO past\_simple (id, task, answer) VALUES (7, 'She (not to like) \_\_ New York.', 'didn''t like');

INSERT INTO past\_simple (id, task, answer) VALUES (8, 'I (to think) \_\_ you (not to be) \_\_ busy.', 'thought, weren''t');

INSERT INTO past\_simple (id, task, answer) VALUES (9, 'How (can) \_\_ you (to forget) \_\_ about her birthday?', 'could, forget');

INSERT INTO past\_simple (id, task, answer) VALUES (10, 'At 7:50 I (to sleep) \_\_, so I (to be) \_\_ able to be at the crime scene.', 'was sleeping, wasn''t');

-- Таблица: plural

CREATE TABLE plural (id INTEGER PRIMARY KEY UNIQUE NOT NULL, task TEXT NOT NULL, answer TEXT NOT NULL);

INSERT INTO plural (id, task, answer) VALUES (1, 'knife (knifs/knives)', 'knives');

INSERT INTO plural (id, task, answer) VALUES (2, 'toy (toyes/toys)', 'toys');

INSERT INTO plural (id, task, answer) VALUES (3, 'mouse (mice/mouses)', 'mice');

INSERT INTO plural (id, task, answer) VALUES (4, 'man (men/mans)', 'men');

INSERT INTO plural (id, task, answer) VALUES (5, 'child (childs/children)', 'children');

INSERT INTO plural (id, task, answer) VALUES (6, 'lady (ladys/ladies)', 'ladies');

INSERT INTO plural (id, task, answer) VALUES (7, 'leaf (leafs/leaves)', 'leaves');

INSERT INTO plural (id, task, answer) VALUES (8, 'watch (watches/watchs)', 'watches');

INSERT INTO plural (id, task, answer) VALUES (9, 'tomato (tomatos/tomatoes)', 'tomatoes');

INSERT INTO plural (id, task, answer) VALUES (10, 'party (parties/partys)', 'parties');

-- Таблица: present\_continuous

CREATE TABLE present\_continuous (id INTEGER PRIMARY KEY UNIQUE NOT NULL, task TEXT NOT NULL, answer TEXT NOT NULL);

INSERT INTO present\_continuous (id, task, answer) VALUES (1, 'I (to write) an essay now.', 'am writing');

INSERT INTO present\_continuous (id, task, answer) VALUES (2, 'Look! Michael (to dance) now.', 'is dancing');

INSERT INTO present\_continuous (id, task, answer) VALUES (3, 'The children (to eat) cakes now.', 'are eating');

INSERT INTO present\_continuous (id, task, answer) VALUES (4, 'I \_\_\_\_ her favoutire song (to sing).', 'am singing');

INSERT INTO present\_continuous (id, task, answer) VALUES (5, 'She \_\_\_\_, she \_\_\_\_ (not to speak, to text)', 'isn''t speaking, is texting');

INSERT INTO present\_continuous (id, task, answer) VALUES (6, 'That cow \_\_\_\_ something (to chew).', 'is chewing');

INSERT INTO present\_continuous (id, task, answer) VALUES (7, '\_\_\_\_ that spider \_\_\_\_ at me? (to look)', 'is, looking');

INSERT INTO present\_continuous (id, task, answer) VALUES (8, 'We \_\_\_\_, we \_\_\_\_ (to walk, not to run).', 'are walking, are not running');

INSERT INTO present\_continuous (id, task, answer) VALUES (9, 'How \_\_\_\_ you \_\_\_\_ now? (to feel)', 'are, feeling');

INSERT INTO present\_continuous (id, task, answer) VALUES (10, 'Tommy \_\_\_\_ a book now. Tommy and Mark \_\_\_\_ chess (not to read, to play).', 'isn''t reading, are playing');

-- Таблица: present\_perfect

CREATE TABLE present\_perfect (id INTEGER PRIMARY KEY UNIQUE NOT NULL, task TEXT NOT NULL, answer TEXT NOT NULL);

INSERT INTO present\_perfect (id, task, answer) VALUES (1, 'He \_\_\_\_\_ (finish) training.', 'has finished');

INSERT INTO present\_perfect (id, task, answer) VALUES (2, 'She \_\_\_\_\_ (score) twenty points in the match.', 'has scored');

INSERT INTO present\_perfect (id, task, answer) VALUES (3, 'We \_\_\_\_\_ (watch) all the Champions League matches this season.', 'have watched');

INSERT INTO present\_perfect (id, task, answer) VALUES (4, 'That''s amazing! She \_\_\_\_\_ (run) fifteen kilometers this morning!', 'has run');

INSERT INTO present\_perfect (id, task, answer) VALUES (5, 'She \_\_\_\_\_ (buy) some really nice rollerblades!', 'has bought');

INSERT INTO present\_perfect (id, task, answer) VALUES (6, 'Oh, no! I \_\_\_\_\_ (lose) my money!', 'have lost');

INSERT INTO present\_perfect (id, task, answer) VALUES (7, 'My mum \_\_\_\_\_ (write) shopping list. It''s on the kitchen table.', 'has written');

INSERT INTO present\_perfect (id, task, answer) VALUES (8, 'Dad, you \_\_\_\_\_ (eat) my biscuit!', 'have eaten');

INSERT INTO present\_perfect (id, task, answer) VALUES (9, 'I’m tired. I \_\_\_\_\_ (watch) three X-Files videos.', 'have watched');

INSERT INTO present\_perfect (id, task, answer) VALUES (10, 'Hurry up! They \_\_\_\_\_ (start) the film!', 'have started');

-- Таблица: present\_simple

CREATE TABLE present\_simple (id INTEGER PRIMARY KEY UNIQUE NOT NULL, task TEXT NOT NULL, answer TEXT NOT NULL);

INSERT INTO present\_simple (id, task, answer) VALUES (1, 'They \_\_\_\_\_ football at the institute. (to play)', 'play');  
INSERT INTO present\_simple (id, task, answer) VALUES (2, 'She \_\_\_\_\_ emails. (not / to write)', 'does not write');  
INSERT INTO present\_simple (id, task, answer) VALUES (3, 'My mother \_\_\_\_ fish. (not / to like)', 'does not like');  
INSERT INTO present\_simple (id, task, answer) VALUES (4, 'His brother \_\_\_\_\_ in an office. (to work)', 'works');  
INSERT INTO present\_simple (id, task, answer) VALUES (5, 'She \_\_\_ very fast. (cannot / to read)', 'cannot read');  
INSERT INTO present\_simple (id, task, answer) VALUES (6, 'His wife \_\_\_\_\_ a motorbike. (not / to ride)', 'does not ride');  
INSERT INTO present\_simple (id, task, answer) VALUES (7, ' I ... a student.(to be)', 'am');  
INSERT INTO present\_simple (id, task, answer) VALUES (8, 'Max ... an office-worker.(to be)', 'is');  
INSERT INTO present\_simple (id, task, answer) VALUES (9, 'We ... late, sorry!(to be)', 'are');  
INSERT INTO present\_simple (id, task, answer) VALUES (10, 'Alice \_\_\_\_ a sister.(to have)', 'has');

-- Таблица: users

CREATE TABLE users (user\_id INTEGER UNIQUE NOT NULL, user\_name TEXT NOT NULL, user\_surname TEXT, username STRING, count\_user INTEGER);  
INSERT INTO users (user\_id, user\_name, user\_surname, username, count\_user) VALUES (648921357, 'п', 'р', 'Андрей', 1);  
INSERT INTO users (user\_id, user\_name, user\_surname, username, count\_user) VALUES (123454668, 'f', 'g', 'Стас', 15);  
INSERT INTO users (user\_id, user\_name, user\_surname, username, count\_user) VALUES (935843820, 'Andrey', NULL, 'Иван', 5350);  
INSERT INTO users (user\_id, user\_name, user\_surname, username, count\_user) VALUES (323412985, 'Татьяна', 'Распопина', 'Lol', 600);  
INSERT INTO users (user\_id, user\_name, user\_surname, username, count\_user) VALUES (650450315, 'Иван', 'Пронин', 'lol', 0);  
INSERT INTO users (user\_id, user\_name, user\_surname, username, count\_user) VALUES (456548412, 'Данил', 'Пронин', 'Кнехт', 1050);  
INSERT INTO users (user\_id, user\_name, user\_surname, username, count\_user) VALUES (561515565, 'Данил', 'р', 'Данил', 600);  
INSERT INTO users (user\_id, user\_name, user\_surname, username, count\_user) VALUES (659595623, 'Настя', 'а', 'Настя', 900);  
INSERT INTO users (user\_id, user\_name, user\_surname, username, count\_user) VALUES (789451214, 'Алёна', 'о', 'Алёна', 300);  
INSERT INTO users (user\_id, user\_name, user\_surname, username, count\_user) VALUES (789845132, 'Петр', 'е', 'Петр', 450);  
INSERT INTO users (user\_id, user\_name, user\_surname, username, count\_user) VALUES (458484515, 'Игорь', 'н', 'Игорь', 1540);  
INSERT INTO users (user\_id, user\_name, user\_surname, username, count\_user) VALUES (409337883, 'IwantBlini', NULL, 'Сергей', 0);

COMMIT TRANSACTION;

PRAGMA foreign\_keys = on;