Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Космических и информационных технологий
институт

Вычислительная техника
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой ВТ
_____ О. В. Непомнящий
подпись инициалы, фамилия
« ____ »_____ 20__ г.

БАКАЛАВРСКАЯ РАБОТА

09.03.01 Информатика и вычислительная техника
код и наименование направления

Телеграм бот – изучение английского языка
тема

Пояснительная записка

Руководитель	_____	доцент, канд.физ-мат.наук	К. В. Коршун
	подпись, дата	должность, ученая степень	инициалы, фамилия
Выпускник	_____		В. А. Галкин
	подпись, дата		инициалы, фамилия
Нормоконтролер	_____	доцент, канд.физ-мат.наук	К. В. Коршун
	подпись, дата	должность, ученая степень	инициалы, фамилия

Красноярск 2021

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Космических и информационных технологий
институт

Вычислительная техника
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой ВТ
_____ О. В. Непомнящий
подпись инициалы, фамилия
« ___ »_____ 20__ г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
в форме бакалаврской работы

Студенту Галкину Вячеславу Анатольевичу
фамилия, имя, отчество

Группа КИ17-07Б Направление(специальность) 09.03.01
номер код

Информатика и вычислительная техника
наименование

Тема выпускной квалификационной работы Телеграм бот – изучение английского языка

Утверждена приказом по университету № от

Руководитель ВКР К. В. Коршун, доцент, канд.физ-мат.наук
инициалы, фамилия, должность, учёное звание и место работы

Исходные данные для ВКР: Спроектировать чат-бота, изучить существующие технологии разработки ботов, разработать общую архитектуру системы, выбрать аппаратное и программное обеспечение системы, разработать конфигурацию аппаратного и программного обеспечения, разработать и реализовать прототип системы, провести тестирование работоспособности.

Перечень разделов ВКР: Аналитическая часть, разработка общей архитектуры системы, разработка чат-бота, проверка работоспособности системы.

Перечень графического материала: презентация в формате Power Point.

Руководитель ВКР _____ К. В. Коршун
Подпись инициалы, фамилия

Задание принял к исполнению _____ В. А. Галкин
Подпись инициалы, фамилия

«___»_____2021 г.

РЕФЕРАТ

Выпускная квалификационная работа по теме «Телеграм бот – Изучение английского языка» содержит 41 страницу текстового документа, 1 таблицу, 12 изображений, 9 использованных источников.

TELEGRAM, ЧАТ-БОТ, PYTHON, SQLITE, СФУ

Целью выпускной квалификационной работы является разработка бота помощника для мессенджера Telegram.

Основные задачи:

- анализ выбранной предметной области;
- выбор технологий и среды разработки;
- разработка чат-бота на платформе Telegram;
- тестирование чат-бота;

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 3
1 Обзор предметной области 4
1.1 Мессенджеры 4
1.2 Чат-боты 6
1.3 Пример чат-бота 6
1.4. Telegram Bot API 8
2 Выбор технологий и среды разработки 10
2.1 Язык программирования Python 10
2.1.1. Framework Aiogram 11
2.2 База данных SQLite 11
2.3 Среда разработки PyCharm 13
3. Реализация чат-бота 14
3.1 Регистрация чат-бота для Telegram 14
3.2. Описания функциональности чат-бота 15
3.3. Описание прецедентов 15
3.4. Описание серверной части 17
3.4.1. База данных 17
3.4.2. Описание серверной части чат-бота 19
4. Запуск и тестирование 21
ЗАКЛЮЧЕНИЕ 24
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 25
ПРИЛОЖЕНИЕ А 27
ПРИЛОЖЕНИЕ Б 33
ПРИЛОЖЕНИЕ В 35
ПРИЛОЖЕНИЕ Г 40
ПРИЛОЖЕНИЕ Г 41

ВВЕДЕНИЕ

Сейчас уже трудно представить себе жизнь без интернета. В нем ежесекундно происходят миллионы событий начиная от простого поиска информации, общения и просмотра новостей заканчивая глобальными событиями по типу аукционов или введения государственных интернет услуг. В настоящее время трудно найти кампанию не имеющую свою группу в социальных сетях или собственный сайт. Но в данной работе нас будет интересовать общение, которое из реального мира постепенно переходит в виртуальное благодаря доступности мессенджеров и простоте их использования.

Актуальность выпускной квалификационной работы обусловлена высокой популярностью мессенджеров и таких средств автоматизации как чат-боты среди пользователей сети Интернет. Чат-боты позволяют упростить ежедневные рутинные задачи, такие как получение информации о погоде, пробках, последних новостях и другие. Главным достоинством относительно классических приложений является возможность совмещения всех возможностей на платформе одного мессенджера.

Данная работа посвящена созданию бота способного обучить английскому языку. На основании этого были приняты дальнейшие задачи выпускной квалификационной работы:

- анализ выбранной предметной области;
- выбор технологий и среды разработки;
- разработка чат-бота на платформе Telegram;
- тестирование чат бота;

1 Обзор предметной области

1.1 Мессенджеры

Свое развитие интернет-сервисы для общения начали с чатов, потом мессенджеры, потом социальные сети, но с недавних пор мессенджеры вновь возглавили список самых перспективных сервисов. Это подтверждено исследованиями компаний «We Are Social» и «Hootsuite» в 2021 году: количество пользователей мессенджеров в мире увеличилось на 13%. [1] Одновременно с этим среднее количество мессенджеров, используемых одним пользователем, увеличилось за 2021 год с 3 до 4 штук. Исследования компании «Билайн» показали, что мессенджеры используются преимущественно взрослыми, так на долю лиц до 18 лет приходится 7,7% от общего количества аудитории, пользователи в возрасте от 18 до 25 лет составляют 9,8%, от 25 до 35 лет – 32,3%, от 35 до 45 лет – 2б,8%. При этом 13,1% пользователей мессенджеров находятся в возрасте от 45 до 55 лет, а 7,6% — от 55 до б4 лет. И всего 2,7% аудитории мессенджеров находится в возрасте старше 64 лет. [2]

Причина повторной волны популярности мессенджеров – самоизоляция которая заставила большее количество людей использовать мессенджеры для общения с близкими и коллегами, что сильно повлияло на рост трафика внутри приложений, за год он увеличился более чем в четыре раза. [2]

Telegram является приложением созданным на языке программирования C++. Клиентские приложения могут быть установлены на различные платформы: Windows, Linux, MacOS, IOS, Android так же существует веб-версия. Пользователи могут обмениваться сообщениями и различными файлами. Для работы приложение использует серверную часть с закрытым кодом, который расположен на серверах, которые находятся в разных концах планеты.

Главной особенностью Telegram считается специально разработанный протокол шифрования MTProto и возможность создания секретных чатов.

Секретные чаты требуются в тех случаях, когда посланное сообщение должно быть максимально защищено и даже быть удалено спустя некоторое время, например, при передачи каких-нибудь паролей или важных файлов содержащих интеллектуальную собственность. В этих чатах реализовано «Оконченное шифрование» которое гарантирует невозможность перехватки сообщений даже если вас взломают, ибо секретные чаты привязаны к самим устройствам. Сообщения из таких чатов не поддаются пересылке и не оставляют следов на сервере Telegram. Так же существуют некоторые особенности у этих чатов, а именно:

В том случае если ваш партнер или друг, с которым вы общались в секретном чате сменит режим чата на обычный то сообщения начнут шифроваться стандартным методом шифрования и это произойдет со всеми сообщениями в чате что были не удалены.

Если ваш собеседник сделает фото экрана, на котором будет секретный чат то уведомление о фото придет на ваше устройство.

Когда вы выйдете из аккаунта, но забудете закрыть секретный чат он удалится автоматически без возможности восстановления.

В данный момент Telegram может работать на многих устройствах: телефонах, компьютерах и даже доступен в веб версии. Так же он не обошел стороной и Linux так как изначальные пользователи были из сферы ИТ и больше пользовались Linux для своей работы. Сейчас Telegram переведен на многие языки: русский, французский, немецкий, итальянский, английский, польский и т. д.

Какими преимуществами обладает Telegram над другими мессенджерами:

- Имеет Porbale и веб-версию

- открытость – использование открытого протокола MTProto и API, бесплатных для всех;
- скорость отправки сообщений быстрее остальных мессенджеров;
- ограничение на размер отправляемых файлов до 2 ГБ.
- В Telegram отсутствует реклама;
- все сообщения будут зашифрованы, а сообщения из секретных чатов удалены без следов на сервере Telegram;

1.2 Чат-боты

Чат-боты — это программа в основе, которой ИИ, которая способна принимать от пользователя различную информацию: сообщение, команды, файлы и т. п. после обработать их и выдать результат.

В основном выделяют чат-ботов двух типов:

1) Декларативные чат-боты, ориентированные на задания

Задействуют заранее прописанные инструкции и используются в основном для обслуживания и поддержки в формате 24\7.

2) Предиктивные чат-боты на основе данных, работающие в режиме диалога

Задействуют машинное обучение и являются более сложной формой декларативных чат-ботов. Они способны учитывать контекст отправленного пользователем сообщения и отправить более персонализированный ответ.

Чат-боты могут применятся для самых различных задач: простые ответы на вопросы по командам, различные тесты, отправка файлов, проверка файлов на вирусы, построение маршрутов по карте, вызов такси и т. п.

1.3 Пример чат-бота

Для примера возьмем простого чат-бота «Skeddy»

Этот чат-бот был создан для людей, которые часто что-то забывают. Вот представьте себе ситуацию, когда вы находитесь на работе и тут резко вспоминаете что дома, закончилось молоко, но из-за наряженной работы часто заходя вечером в магазин вы просто забывали об этом и утром не могли выпить чашку кофе с молоком. Тут на помощь приходит чат-бот «Skeddy»

С его помощь вы запросто можете создать себе заметку или напоминание, привязанное ко времени. Почему же это так удобно? Потому что мессенджеры в наше время крайне популярны и есть у большинства, а также удобности способствует то как именно задаются напоминания. От вас требуется только найти этого чат-бота и нажать кнопку старт. После можно просто писать ему обычные фразы, например, «Купить молоко в 22 часа» или «Выключить плиту через 20 минут» и это удобно, когда под рукой нет таймера, а встроенный таймер в телефоне слишком тихий.

Так же можно использовать настоящее расписание, которое будет напоминать вам сделать что-либо каждый месяц или неделю и т. п. в определенное время нужно лишь ввести: «Передать показания каждый первый день месяца в 9 утра».

Примеры сообщений показаны на рисунке 1.

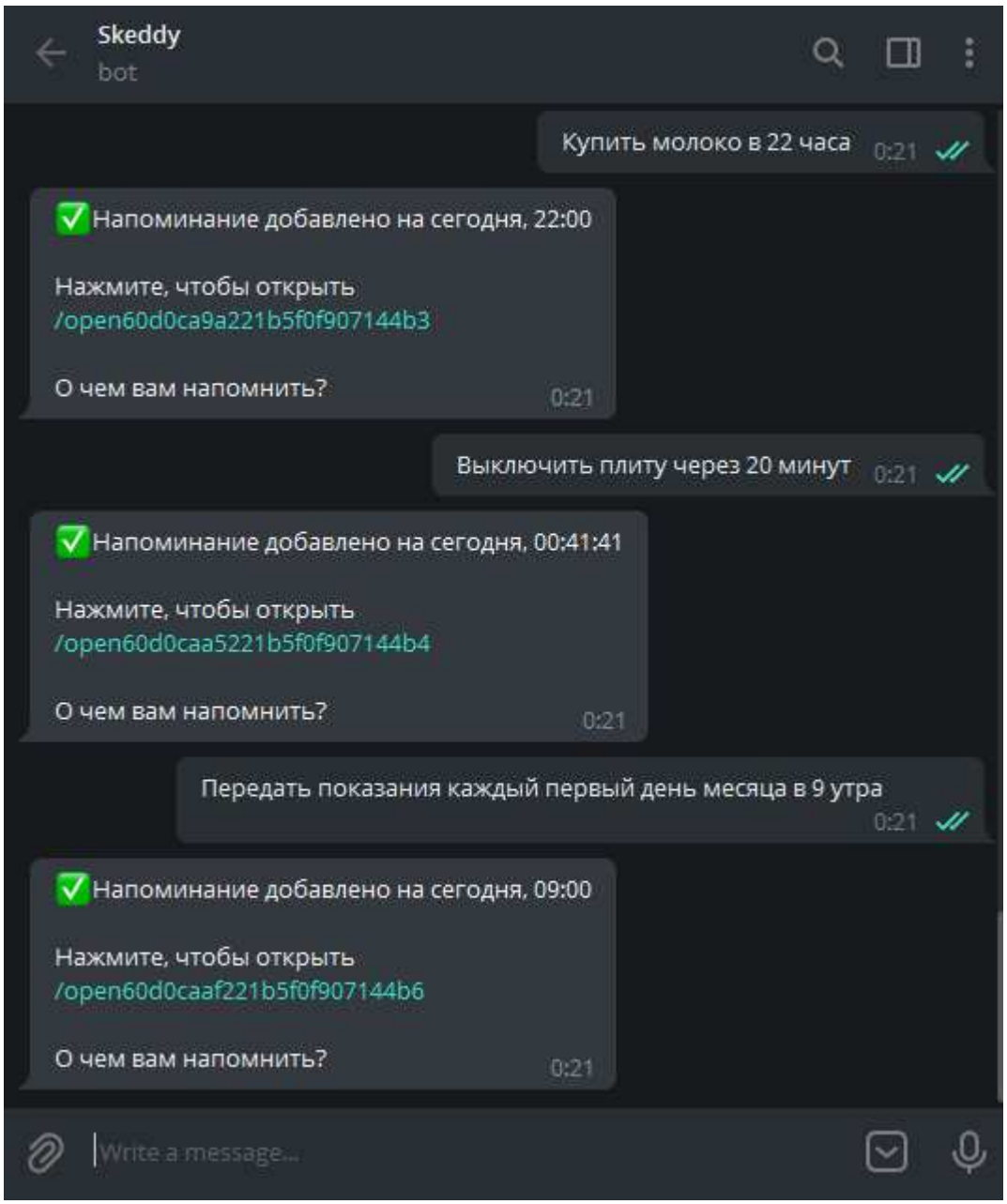

Рисунок 1 – пример создания заметок в чат-боте «Skeddy»

1.4. Telegram Bot API

Bot API дает возможность подключить бота к системе Telegram. Telegram-боты-это обычные учетные записи, но для их настройки не требуется дополнительный номер телефона. Управлять этими ботами можно используя HTTPS – запросы. Но при этом существует два метода получения данных\отправления данных.

Long Pooling и Webhook – позволяют запросить обновления от бота.

- Первый метод постоянно через определенный промежуток времени опрашивает сервера Telegram о наличии обновлений.
- Второй метод сервер сам оповещает приложение о обновлениях

Чтобы управлять ботом с помощью Telegram Bot API требуется получить специальный токен который является уникальным для каждого бота. Его получить, зарегистрировав своего нового чат-бота в особом чат-боте «BotFarher». Полную документацию можно найти в списке источников под номером 9. [9]

Пример диаграммы Telegram Bot API представлен на рисунке 2

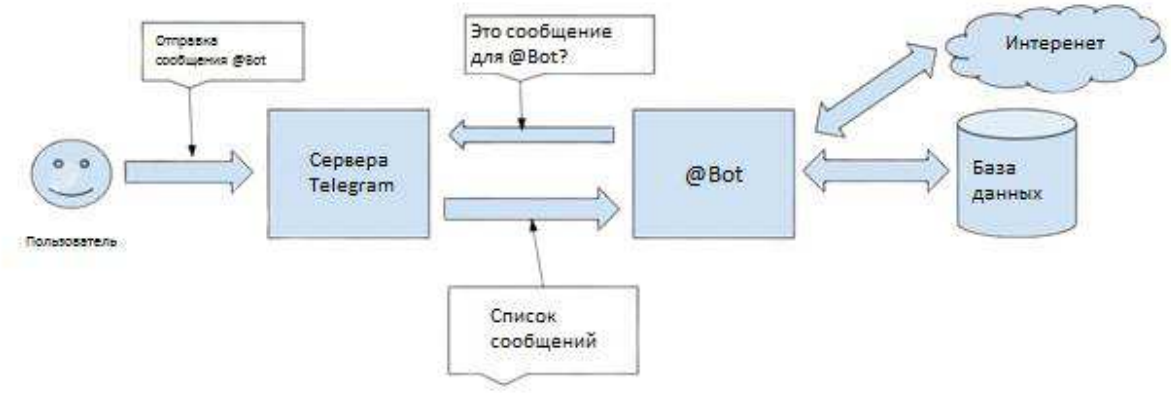

Рисунок 2 – пример диаграммы Telegram Bot API

В настоящее время Telegram Bot API реализован на различных платформах и языках.

2 Выбор технологий и среды разработки

Во время поиска подходящих решений для реализации чат-бота был сделан вывод, что для разработки бота требуется знать хотя бы один язык серверного программирования, например: Python, Ruby, Node.JS или PHP. Нужно было определиться какой именно язык изучать и использовать. Так же немаловажно уметь работать с REST (Representational State Transfer) API (Application Programming Interface), который предоставляют мессенджеры, а именно Telegram Bot API.

2.1 Язык программирования Python

В качестве языка программирования был выбран язык Python.

Python имеет структурированный и хорошо читаемый код а так же обладает высокой производительностью программных решений. На Python благодаря широкому выбору встроенных библиотек и полезных функций возможно написать большинство программ от самых простых скриптов до систем управления.

Из-за простоты кода, строгой динамической типизации, автоматического управления памятью Python обладает низким порогом вхождения и даже не самый опытный программист с легкостью сможет написать нужную ему не большую программу этому так же поспособствует огромное количество информации находящихся в открыты источниках, а благодаря хорошей читаемости в будущем будет намного проще корректировать и дорабатывать программу

Одним из преимуществ Python является то что он реализован на разных платформах и операционных системах.

Минусом Python является низкая скорость выполнения программ так как она будет интерпретируема, однако данный минус можно не учитывать при написании программ для которых скорость выполнения не будет считаться критичной.

Поскольку мы будем использовать язык программирования Python в качестве фреймворка для Telegram Bot API мы возьмем Aiogram.

2.1.1. Framework Aiogram

Aiogram - это простой и полностью асинхронный фреймворк для API Telegram Bot, написанный на Python 3.7 с asyncio и aiohttp.

Данный фреймворк позволяет более просто использовать Bot API, не тратя огромные ресурсы времени на изучении что позволяет сконцентрировать свое внимание на разработке чат-бота.

Чтобы начать использовать фреймворк aiogram нужно скачать и установить его командой меденжера пакетов PIP:

«$ pip install -U aiogram»

После чего в самой программе бота подключить следующие модули:

«from aiogram import Bot, Dispatcher, executor, types»

Затем требуется инициализировать полученные от «BotFather» токен, «Bot» и «Dispatcher»

И после всех этих действий можно приступать к программированию своего собственного чат-бота [5].

2.2 База данных SQLite

SQLite — это компактная интегрированная реляционная база данных. Исходный код библиотеки был передан в общественное достояние. Это чисто реляционная база данных [6].

Именно то что в SQLite существует большой набор различных инструментов для реализаций в отличии от сетевых систем, а также присутствует прямое обращение к файлам от чего количество таблиц и сами баз ограничивается только жестким диском пользователя и ко всему выше сказанному база данных сразу встроена в виде модуля в языке программирования Python сподвигнуло меня использовать SQLite в своей реализации чат-бота.

Как позже выяснилось из-за своей компактности и простоты SQLite идеально подходит для создания небольших проектов для демонстрации возможностей. Существует только один минус: максимальный размер одной базы данных составляет 2ГБ, однако это мешает крайне редко.

Сегодня большая часть разработчиков использует SQLite Поддерживаемые типы данных:

– NULL: значение NULL.

– INTEGER: целое число со знаком, хранящееся в 1, 2, 3, 4, 6 или 8 байтах.

– REAL: число с плавающей запятой, хранящееся в 8-байтовом формате IEEE.

– TEXT: текстовая строка, закодированная UTF-8, UTF-16BE или UTF-16LE.

– BLOB: тип данных, хранящихся в той же форме, в которой они были получены.

Преимущества:

– Файловая: вся база данных хранится в одном файле, что облегчает перемещение.

– Стандартизированное: SQLite использует SQL. [6]

Недостатки:

– Отсутствие пользовательского управления: другие более обширные базы данных позволяют пользователям управлять связями в большом количестве таблиц в соответствии с привилегиями. У SQLite такой функции нет.

– Невозможность дополнительной настройки: У SQLite невозможно поднять производительность.

2.3 Среда разработки PyCharm

PyCharm – интегрированная среда разработки для Python. Разрабатывалась компанией JetBrains. За основу взята другая интегрированная среда разработки IntelliJ IDEA. Существует в двух версиях: бесплатной (Community Edition) и платной (Professional Edition). Также существует версия Educational Edition – переработанная среда разработки для обучения программированию. В отличие от обычных IDE данная версия имеет встроенные примеры и задачи для обучения, и упрощенный интерфейс [7].

В PyCharm было реализовано множество функций для лучшего усвоения написанного кода начиная от банального подсвечивания кода заканчивая авто-форматированием. PyCharm возможно запустить на различных платформах например: Windows, Linux и Mac OS. Достигнуть этой кроссплатформенности смогли путем написания среды разработки на двух языках Python и Java

Данная среда разработки была выбрана благодаря раннему опыту работы с PyCharm что должно помочь в более быстрой и качественной реализации поставленных задач. Так же кампания JetBrains судя по отчетам за год набирает популярность расширяя свою аудиторию и штат сотрудников.

3. Реализация чат-бота

3.1 Регистрация чат-бота для Telegram

Для начала разработки нужно пройти регистрацию у особого чат-бот «BotFather».

Требуется использовать команду «/newbot» чтобы зарегистрировать нового чат-бота. Затем «BotFather» запросит написать вас имя и имя пользователя, а затем сгенерирует уникальный токен авторизации для нашего нового чат-бота.

Уникальный токен авторизации имеет вид «110201543:AAHdqTcvCH1vGWJxfSeofSAs0K5PALDsaw» и должен быть защищен вами. В случае если ваш токен был скомпрометирован или вы потеряли его существует возможность создать новый токен введя команду «/token»[12].

Так же у чат-бота присутствуют пострегистрационные настройки описанные в таблице 1

Таблица 1 – Доступные команды для изменения чат-ботов

Команда	Описание
/setname	Позволяет сменить имя чат-бота
/setdescription	Устанавливает описание бота, которое будет показано, при первом открытии чат бота
/setabouttext	Присваивает текст в поле «О чат-боте»
/setuserpic	Присваивает выбранную картинку
/setcommands	Позволяет создать список доступных команд
/deletebot	Удалаяет указанного чат-бота

После выполнения настроек с помощью чат-бота «BotFather» можно приступить к разработке программной части чат-бота

3.2. Описания функциональности чат-бота

В первую очередь чат-бот должен будет помочь людям, которые только начали изучать английский язык и им критически не хватает базы слов для комфортного продолжения изучения.

Для удобства использования и автоматического расширения было принято решение использовать интерфейсы со встроенный клавиатурой

Планируется реализовать следующие меню:

- Главное меню
- Меню настроек
- Меню выбора блока для изучения
- Меню выбора верного слова
- Меню подтверждения
- Меню завершения блока

Чат-бот при первом запуске пользователем предлагает ввести команду «/start» после чего пользователю будет показано меню, в котором он должен выбрать тему для изучения и подтвердить начало изучения.

В случае успешного завершения теста без единой ошибки пользователь получит сообщение об успешном завершении темы в ином случае вернется в меню выбора темы.

3.3. Описание прецедентов

На рисунке 3 представлена диаграмма прецедентов

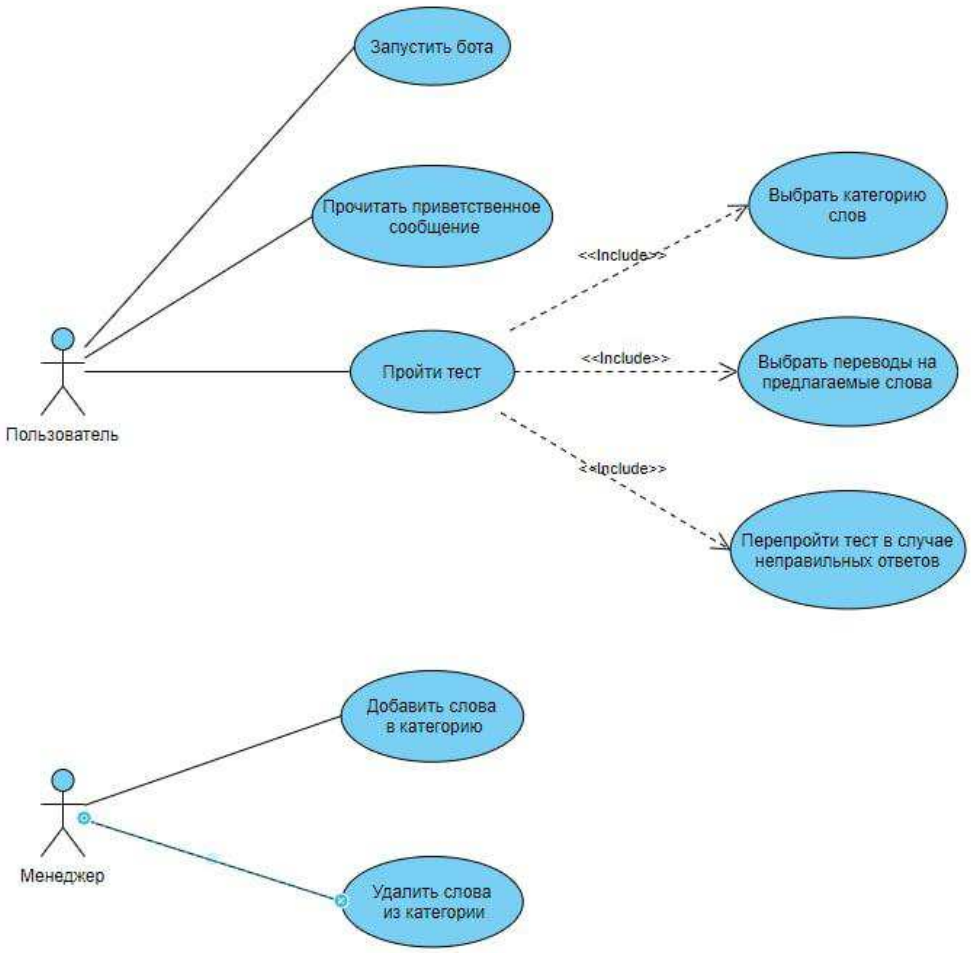

Рисунок 3 – Диаграмма прецедентов

Текстовое описание прецедентов:

Прецедент «Запустить бота» активируется командой «/start» и позволяет пользователям ознакомиться с представленным чат-ботом и выбрать дальнейшие действия.

Прецедент «Пройти тест» включает в себя несколько прецедентов и позволяет пользователю успешно завершить тест или вернуться назад к выбору темы для прохождения.

Прецедент «Выбрать категорию слов» при запуске посылает запрос в базу данных и выводит пользователю на выбор список доступных тем для изучения в данный момент.

Прецедент «Выбрать переводы на предлагаемые слова» запускается после выбора темы и предлагает пользователю английское слово и несколько вариантов перевода. Если пользователь выбирает все переводы верно посылает сообщение о успешном завершении темы и записывает информацию в базу данных.

Прецедент «Перепройти тест в случае неправильных ответов» если пользователь допускает ошибку при прохождении теста его возвращает в меню выбора темы для прохождения.

3.4. Описание серверной части

3.4.1. База данных

Для реализации поставленной задачи потребовалось база данных (БД). В проекте используется БД SQLite. База данных представляет собой 6 таблиц.

Диаграмма представлена на рисунке 4

Рисунок 4 – Диаграмма базы данных

Каждая таблица несет в себе следующую информацию:

Blocks – Содержит в себе номера и названия блоков для изучения

EWords – Содержит в себе Английские слова и номер привязанного блока

RWords – Содержит в себе Русские слова

Translations – Содержит в себе связь между уникальными идентификаторами Английских и Русских слов

User_Block – Содержит в себе информацию о пройденых блоках пользователем

Users – Содержит в себе информацию о пользователях начавшим работы с чат-ботом

Для создания и заполнения базы данных было исрользованно программное обеспечение «DB Brower for SQLite». Таким образом для

заполнения или правки таблиц менеджеру не нужно будет изучать SQL запросы и это повысит доступность чат-бота.

3.4.2. Описание серверной части чат-бота

От серверной части чат-бота требуется решить следующие задачи:

– Получать и обрабатывать сообщение от пользователя

– Получать информацию из базы данных выполняя запросы с необходимыми параметрами

– Составлять и отправлять ответ в виде текста пользователю

Кнопки inline-клавиатуры и их обратная информация позволяют чат-боту получить команду для дальнейших действий. После того как пользователь нажмет кнопку на inline-клавиатуре серверная часть сравнит полученный ответ с вариантами, заложенными в программе и выдаст результат

В документации Telegram Bot API сказано, что каждая клавиатура должна иметь один обязательный параметр – имя кнопки (text), и шесть необязательных – ссылка (url), обратные данные (callback_data), возможность встроенного запроса (switch_inline_query), возможность вывода клавиатуры из другого чата (switch_inline_query_current_chat), вызов описания запущенной игры (callback_game) и кнопка с возможностью покупки (pay).

Пример inline - клавиатуры представлен на рисунке 5

Рисунок 5 – Пример inline – клавиатуры

Для начала реализуем функцию приема начального сообщения и отправку ответа:

```
async def start_message
```

После этого создадим функцию обработки нажатия кнопки «Выбор блока» которая создаст нам новое меню со всеми блоками запросив информацию о существующих и доступных на данный момент блоков:

```
async def process_callback_choose_block
```

Затем реализуем обработку кнопок блоков в которой будем обращаться к базе данных для получения информации о количестве слов в блоке:

async def process_callback_blocks

После начнем работу над функцией обрабатывающей кнопки варианта ответа которая так же будет запрашивать один правильный и 3 не правильных варианта ответа из базы данных:

```
async def process_callback_check
```

В заключении нужно добавить обработку кнопки «Что пройдено» в которой будет обращение к базе данных за информацией о пройденых блоках пользователем.

```
async def process_callback_done
```

Таким образом была реализована серверная часть чат-бота с требуемыми функциями.

4. Запуск и тестирование

При вводе команды «/start» пользователю предлагается начать выбор блока или же перейти в настройки (рисунок 6)

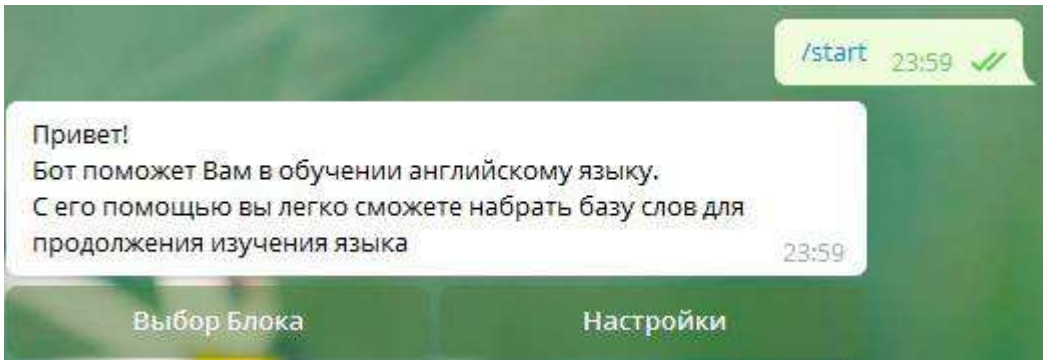

Рисунок 6 – Главное меню

Далее было реализовано меню выбора блока для изучения. Оно позволяет выбрать интересующую нас тему для изучения слов. Среди кнопок клавиатуры можно найти все доступные темы на данный момент (рисунок 7).

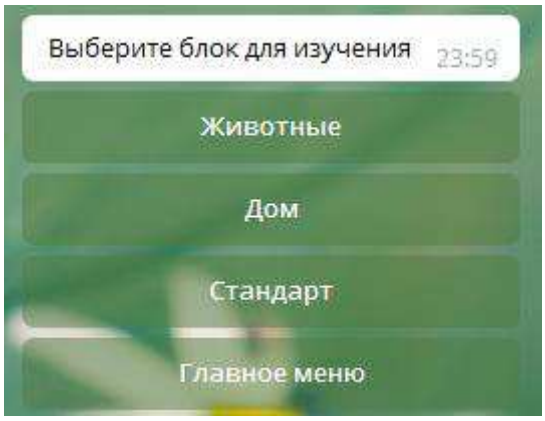

Рисунок 7 – Меню выбора блока для изучения

После выбора блока для изучения последует его описание и подтверждение о начале прохождения (рисунок 8).

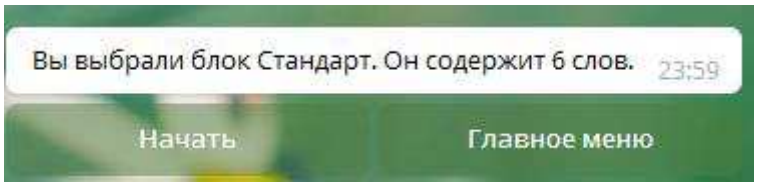

Рисунок 8 – Меню подтверждения

В случае нажатии кнопки «Начать» будет создано новое меню предлагающее вам слово из блока и 4 варианта перевода из которых верный будет только один (рисунок 9).

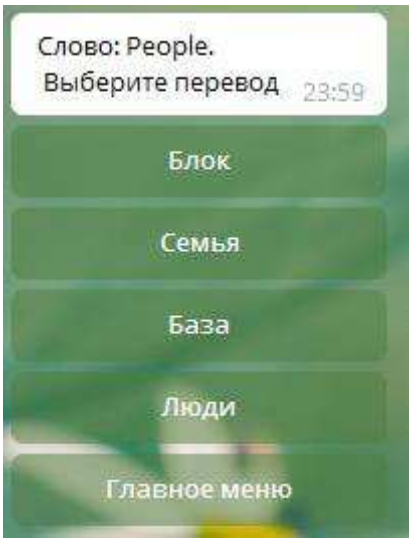

Рисунок 9 – Меню выбора верного слова

После прохождения блока с правильным результатом 100% результатом будет получено сообщение о успешном завершении блока с возможностью возврата в главное меню или выбора следующего блока (рисунок 10)

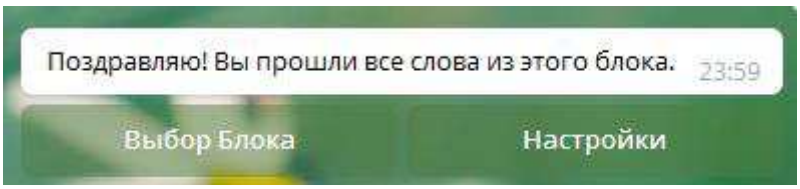

Рисунок 10 – Меню завершения блока

При нажатии «Настройки» будет получена клавиатура настроек (рисунок 11).

Рисунок 11 – Меню настройки

После выбора варианта «Что пройдено» можно получить сообщение о пройденых блоках и с возможностью вернуться в главное меню (рисунок 12)

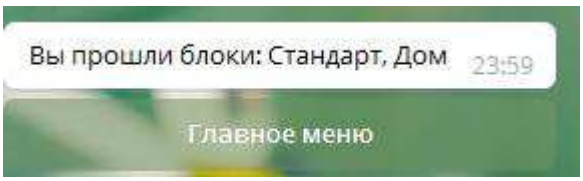

Рисунок 12 – Сообщение о пройденных блоках

На этом реализация клиентской части чат-бота была завершена.

ЗАКЛЮЧЕНИЕ

В рамках выпускной квалификационной работы были выполнены поставленные задачи. Изучены различные базы данных и выбрана более подходящая для них, найдена подходящая среда разработки и язык программирования, реализован и протестирован чат-бот.

В будущем планируется увеличение функциональности чат-бота, а именно:

- Напоминание о необходимости ежедневного\еженедельного прохождения различных тем
- Возможность добавления пользовательских слов и тем
- Возможность изучения грамматики

Конечным результатом выпускной квалификационной работы является реализованный чат-бот в мессенджере Telegram который поможет в изучении английского языка путем наращивания вашей собственной базы слов

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Telegram: новый мессенджер от Павла Дурова [Электронный ресурс] / republic.ru Режим доступа: https://republic.ru/posts/l/978067 (дата обращения 16.06.2021)
2. Оценки рынка мессенджеров [Электронный ресурс] / tadviser.ru Режим доступа: https://www.tadviser.ru/index.php/Статья:Мессенджеры_(Instant_Messenger,_IM) (дата обращения 16.06.2021)
3. SQLite, MySQL и PostgreSQL: сравниваем популярные реляционные СУБД [Электронный ресурс]. – URL: https://tproger.ru/translations/sqlitemysql-postgresql-comparison/ (Дата обращения 16.06.2021).
4. 10 самых популярных мессенджеров в мире [Электронный ресурс] / kanobu.ru Режим доступа: https://kanobu.ru/articles/10-samyih-populyarnyih-messendzherov-v-mire-376082/ (дата обращения 16.06.2021)
5. Aiogram [Электронный ресурс] / aiogram Режим доступа https://docs.aiogram.dev/en/latest/quick_start.html (Дата обращения 21.06.2021)
6. PyCharm Community Edition [Электронный ресурс] / blog.jetbrains.com Режим доступа: https://blog.jetbrains.com/pycharm/2017/09/pycharm-community-edition-and-professional-edition-explained-licenses-and-more/ (Дата обращения 16.06.2021)
7. JetBrains Toolbox [Электронный ресурс] / blog.jetbrains.com Режим доступа: https://blog.jetbrains.com/blog/2016/03/09/jetbrains-toolbox-release-and-versioning-changes/ (Дата обращения 16.06.2021)SQL Lite [Электронный ресурс]. – URL:

https://lecturesdb.readthedocs.io/databases/sqlite.html (Дата обращения 16.06.2021).

8. Bots: An introduction for developers [Электронный ресурс] / core.telegram.org Режим доступа: https://core.telegram.org/bots#6-botfather (Дата обращения 16.06.2021)
9. Telegram APIs [Электронный ресурс] / core.telegram.org Режим доступа: https://core.telegram.org (Дата обращения 16.06.2021)

ПРИЛОЖЕНИЕ А

Файл «main.py»

```
import logging

from bd import get_dict_blocks, get_blocks_list,
get_blocks_list1, get_rus_name_and_id, get_eng_words,
set_user, \
    set_user_block, get_done_blocks
import config

from aiogram import Bot, types
from aiogram.dispatcher import Dispatcher
from aiogram.utils import executor
from aiogram.contrib.fsm_storage.memory import
MemoryStorage
from aiogram.contrib.middlewares.logging import
LoggingMiddleware

from config import BOT_TOKEN

from messages import MESSAGES, get_block_message_start
import keyboard as kb
url =
"https://api.telegram.org/bot1320775247:AAHT5VzNVr4tnri
NowopkLnl3vBWLx4oWHw/getUpdates"

logging.basicConfig(format=u'%(filename)+13s [
LINE:%(lineno)-4s] %(levelname)-8s [%(asctime)s]
%(message)s',
                    level=logging.DEBUG)
```

```
bot = Bot(token=BOT_TOKEN)
dp = Dispatcher(bot, storage=MemoryStorage())

dp.middleware.setup(LoggingMiddleware())
GROUP_ID = -448197357

block_list = get_blocks_list()
block_list1 = get_blocks_list1()
# eng_words = []
count = 0
check = -1

async def update_text_blocks(message: types.Message,
dict):
    await message.edit_text(MESSAGES['blocks_message'],
reply_markup=kb.getMarkup(dict))

async def update_text_menu(message: types.Message):
    await message.edit_text(MESSAGES['enter'],
reply_markup=kb.inline_kb_menu)

async def update_text_blocks_start(message:
types.Message, word_count, rus_name, block):
    await
message.edit_text(get_block_message_start(word_count,
rus_name), reply_markup=kb.getStartBlockMarkup(block))

async def update_text_words(message: types.Message):
```

```
    await message.edit_text("Слово: " +
config.eng_words[count] + ".\n Выберите перевод",

reply_markup=kb.getRusWordsMarkup(config.eng_words[coun
t]))

async def update_text_final_words(message:
types.Message):
    await message.edit_text("Поздравляю! Вы прошли все
слова из этого блока.", reply_markup=kb.inline_kb_menu)

async def update_text_settings(message: types.Message):
    await message.edit_text("Настройка",
reply_markup=kb.getSettingsMarkup())

async def update_text_done_blocks(message:
types.Message, blocks):
    result = "Вы прошли блоки: "
    for s in blocks:
        result = result + s + ", "
    result = result[:len(result) - 2]
    await message.edit_text(result,
reply_markup=kb.getMenuOnlyMarkup())


@dp.callback_query_handler(lambda c: c.data ==
'choose_block')
async def process_callback_choose_block(callback_query:
types.CallbackQuery):
```

```
    # await
bot.send_message(callback_query.from_user.id, 'Нажата
первая кнопка!')
    await update_text_blocks(callback_query.message,
get_dict_blocks())
    print(callback_query.from_user.id)
    await bot.answer_callback_query(callback_query.id)

@dp.callback_query_handler(lambda c: c.data == 'menu')
async def process_callback_menu(callback_query:
types.CallbackQuery):
    await update_text_menu(callback_query.message)
    await bot.answer_callback_query(callback_query.id)

@dp.callback_query_handler(lambda c: c.data in
block_list)
async def process_callback_blocks(callback_query:
types.CallbackQuery):
    print(callback_query.data)
    word_count, rus_name =
get_rus_name_and_id(callback_query.data)
    await
update_text_blocks_start(callback_query.message,
word_count, rus_name, callback_query.data)
    await bot.answer_callback_query(callback_query.id)


@dp.callback_query_handler(lambda c: c.data in
block_list1)
```

```
async def process_callback_start_block(callback_query:
types.CallbackQuery):
    global count
    count = 0
    config.eng_words =
get_eng_words(callback_query.data)
    print(config.eng_words)
    await update_text_words(callback_query.message)
    await bot.answer_callback_query(callback_query.id)

@dp.callback_query_handler(lambda c: c.data == "0" or
c.data == "1" or c.data == "2" or c.data == "3" or
c.data == "10")
async def process_callback_check(callback_query:
types.CallbackQuery):
    global count
    count = count + 1
    if count == len(config.eng_words):
        await
update_text_final_words(callback_query.message)
        set_user_block(callback_query.from_user.id)
    elif callback_query.data == "10":
        await update_text_words(callback_query.message)
    else:
        await
update_text_blocks(callback_query.message,
get_dict_blocks())
    await bot.answer_callback_query(callback_query.id)
```

```
@dp.callback_query_handler(lambda c: c.data ==
"settings")
async def process_callback_settings(callback_query:
types.CallbackQuery):
    await update_text_settings(callback_query.message)
    await bot.answer_callback_query(callback_query.id)

@dp.callback_query_handler(lambda c: c.data == "done")
async def process_callback_done(callback_query:
types.CallbackQuery):
    done_blocks =
get_done_blocks(callback_query.from_user.id)
    await
update_text_done_blocks(callback_query.message,
done_blocks)
    await bot.answer_callback_query(callback_query.id)

@dp.message_handler(commands=['start'])
async def start_message(message: types.Message):
    set_user(message.from_user.id)
    await message.reply(MESSAGES['start'],
reply_markup=kb.inline_kb_menu, reply=False)

@dp.message_handler(commands=[MESSAGES['test_message']]
)
async def help_message(message: types.Message):
    await message.reply(MESSAGES['enter'])

if __name__ == '__main__':
    executor.start_polling(dp)
```

ПРИЛОЖЕНИЕ Б

Файл «keyboard.py»

```
import random
from aiogram.types import ReplyKeyboardRemove, \
    ReplyKeyboardMarkup, KeyboardButton, \
    InlineKeyboardMarkup, InlineKeyboardButton
from messages import MESSAGES
from bd import get_rus_right_words, get_other_rus_words

inline_btn_choose_block = InlineKeyboardButton('Выбор Блока',
callback_data='choose_block')
inline_btn_settings = InlineKeyboardButton('Настройки', callback_data='settings')
inline_btn_menu = InlineKeyboardButton('Главное меню', callback_data='menu')
inline_kb_menu = InlineKeyboardMarkup().add(inline_btn_choose_block,
inline_btn_settings)

def getMarkup(blockDict):
    print(blockDict)
    markup = InlineKeyboardMarkup()
    for key, value in blockDict.items():
        markup.add(InlineKeyboardButton(value, callback_data=key))
    markup.add(inline_btn_menu)
    return markup

def getStartBlockMarkup(block):
    markup = InlineKeyboardMarkup()
    block = block + '1'
    print(block)
    text = "Начать"
```

```
    markup.add(InlineKeyboardButton(text, callback_data=block),
inline_btn_menu)
    return markup

def getRusWordsMarkup(word):
    right_words = get_rus_right_words(word)
    other_words = get_other_rus_words(word)
    sum_words = [right_words[0], other_words[0], other_words[1], other_words[2]]
    random.shuffle(sum_words)
    answerId = sum_words.index(right_words[0])
    markup = InlineKeyboardMarkup(row_width=2)
    for idx, val in enumerate(sum_words):
        if idx == answerId:
            markup.add(InlineKeyboardButton(val, callback_data="10"))
        else:
            markup.add(InlineKeyboardButton(val, callback_data=str(idx)))
    markup.add(inline_btn_menu)
    return markup

def getSettingsMarkup():
    markup = InlineKeyboardMarkup()
    markup.add(InlineKeyboardButton("Что пройдено", callback_data="done"),
               InlineKeyboardButton("Напоминания", callback_data="remember"),
               inline_btn_menu)
    return markup
def getMenuOnlyMarkup():
    markup = InlineKeyboardMarkup()
    markup.add(inline_btn_menu)
    return markup
```

ПРИЛОЖЕНИЕ В

Файл «bd.py»

```
import sqlite3
import logging
import random
import config

def get_dict_blocks():
    con = sqlite3.connect("DataBase.db")
    cur = con.cursor()
    cur.execute("select Name, R_Name from Blocks")
    result = {}
    for row in cur:
        result[row[0]] = row[1]
    con.close()
    return result

def get_blocks_list():
    con = sqlite3.connect("DataBase.db")
    cur = con.cursor()
    cur.execute("select Name from Blocks")
    result = []
    for row in cur:
        result.append(row[0])
    con.close()
    return result

def get_blocks_list1():
```

```
    con = sqlite3.connect("DataBase.db")
    cur = con.cursor()
    cur.execute("select Name from Blocks")
    result = []
    for row in cur:
        result.append(row[0] + "1")
    con.close()
    return result

def get_rus_name_and_id(name):
    con = sqlite3.connect("DataBase.db")
    cur = con.cursor()
    cur.execute("select R_Name from Blocks where Name=\"" + name + "\"")
    row = cur.fetchone()
    rus_name = row[0]
    cur.execute("select count(*) from EWords where Block_id=(select Id from
Blocks where Name=\"" + name + "\")")
    row = cur.fetchone()
    word_count = row[0]
    con.close()
    return word_count, rus_name

def get_eng_words(name1):
    name = name1.replace("1", "")
    con = sqlite3.connect("DataBase.db")
    cur = con.cursor()
    cur.execute("select Word from EWords where Block_id= (select Id from Blocks
where Name=\"" + name + "\")")
    result = []
```

```
    for row in cur:
        result.append(row[0])
    con.close()
    random.shuffle(result)
    return result

def get_rus_right_words(word):
    con = sqlite3.connect("DataBase.db")
    cur = con.cursor()
    cur.execute(
        "select Word from Translations inner join RWords on Translations.RId =
RWords.Id where EId = (select Id from EWords where Word = \"" + word + "\")")
    result = []
    for row in cur:
        result.append(row[0])
    con.close()
    random.shuffle(result)
    return result

def get_other_rus_words(word):
    con = sqlite3.connect("DataBase.db")
    cur = con.cursor()
    cur.execute(
        "select Word from Translations inner join RWords on Translations.RId =
RWords.Id where EId != (select Id from EWords where Word = \"" + word + "\")")
    result = []
    for row in cur:
        result.append(row[0])
    con.close()
```

```
    random.shuffle(result)
    return result

def get_name_block_by_word(word):
    con = sqlite3.connect("DataBase.db")
    cur = con.cursor()
    cur.execute("select Name from Block where Id = (select Block_id from EWords
where Word = \"" + word + "\")")
    row = cur.fetchone()
    result = row[0]
    con.close()
    return result

def set_user(id):
    con = sqlite3.connect("DataBase.db")
    cur = con.cursor()
    print(id)
    cur.execute("insert or ignore into Users(Telegram_id) values(" + str(id) + ")")
    con.commit()
    con.close()

def set_user_block(id):
    print(config.eng_words)
    con = sqlite3.connect("DataBase.db")
    cur = con.cursor()
    cur.execute("select Block_id from EWords where Word = \"" +
config.eng_words[0] + "\"")
    row = cur.fetchone()
    block_id = row[0]
```

```
    user_id = []
    block_completed = []
    cur.execute("select Block_id from User_Block where User_id = " + str(id))
    for row in cur:
        block_completed.append(row[0])

    if block_id not in block_completed:
        con.cursor().execute("insert into User_Block(User_id, Block_id) values(" +
str(id) + ", " + str(block_id) + ")")
        print("sssss", id, block_id)
        con.commit()

    con.close()

def get_done_blocks(id):
    con = sqlite3.connect("DataBase.db")
    cur = con.cursor()
    cur.execute("SELECT R_Name from Blocks inner join User_Block on
User_Block.Block_id = Blocks.Id where User_Block.User_id = " + str(id))
    result = []
    for row in cur:
        result.append(row[0])
    return result
```

ПРИЛОЖЕНИЕ Г

Файл «messages.py»

```
enter_message = 'Бот поможет Вам в обучении английскому языку. \nС его помощью вы легко сможете набрать базу слов для продолжения изучения языка' \

start_message = 'Привет!\n' + enter_message
invalid_key_message = 'Ключ "{key}" не подходит.\n' + enter_message
state_reset_message = 'Состояние успешно сброшено'
current_state_message = 'Текущее состояние - "{current_state}", что удовлетворяет условию "один из {states}"'
blocks_message = 'Выберите блок для изучения'
blocks_message_start = 'Вы выбрали блок  ... Он содержит ... слов'
blocks_message_words = 'Тут слово из БД'

MESSAGES = {
    'start': start_message,
    'enter': enter_message,
    'invalid_key': invalid_key_message
    'state_reset': state_reset_message,
    'current_state': current_state_message,
    'blocks_message': blocks_message,
    'blocks_message_start': blocks_message_start,
    'blocks_message_words': blocks_message_words,
}

def get_block_message_start(word_count, rus_name):
    return "Вы выбрали блок " + rus_name + ". Он содержит " + str(word_count) + " слов."
```

ПРИЛОЖЕНИЕ Г

Файл «config.py»

```
BOT_TOKEN = 'you’r bot_token here'


CHANNEL_NAME = '@you’r user_name here'


eng_words = []
```

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Космических и информационных технологий
институт

Вычислительная техника
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой ВТ
_____ О. В. Непомнящий
подпись инициалы, фамилия
« 25 » 06 2021 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.01 Информатика и вычислительная техника
код и наименование направления

Телеграм бот – изучение английского языка
тема

Пояснительная записка

Руководитель _____ 18.06.21 доцент, канд.физ-мат.наук К. В. Коршун
подпись, дата должность, ученая степень инициалы, фамилия

Выпускник _____ 18.06.21 В. А. Галкин
подпись, дата инициалы, фамилия

Нормоконтролер _____ 18.06.21 доцент, канд.физ-мат.наук К. В. Коршун
подпись, дата должность, ученая степень инициалы, фамилия

Красноярск 2021