

Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет информатики и
радиоэлектроники»
Факультет компьютерных систем и сетей
Кафедра информатики

Курсовой проект по дисциплине:
«Программирование»

Пояснительная записка к курсовой работе

Тема работы:
«Бот-тестировщик»

В.	Исполнитель студентка гр. 653501	_____	Белясова А.
		(подпись дата)	
	Руководитель	_____	Козуб В.Н.
		(подпись дата)	

		(оценка)	

Минск

Оглавление

Введение	3
Формулировка цели, мотивы и платформа	3
Поставленные задачи и план их решения	5
Основная часть.....	6
Сравнение с аналогами.....	6
Интерфейс и дизайн.....	9
Техническая реализация.....	15
Создание бота в Telegram	15
Telegram Bot API.....	17
Python	18
pyTelegramBotAPI.....	19
Redis – СУБД.....	22
Логика бота.....	24
Заключение.....	26
Литература.....	27

Введение

Формулировка цели, мотивы и платформа

Целью данной курсовой работы было создание бота для платформы Telegram, который тестирует знания пользователей в области грамматики английского языка.

Telegram – это бесплатный кроссплатформенный мессенджер, позволяющий обмениваться текстовыми сообщениями и медиафайлами различных форматов. Он абсолютно поддерживаемый, может быть беспрепятственно запущен на любом устройстве. Главной особенностью данной сервисной службы является скорость, доступность, безопасность и надежность. Вам абсолютно не стоит беспокоиться за сохранность собственных данных, так как Telegram придерживается политики строгой секретности в отношении личной информации пользователей, интересы которых ставит выше всего. Для этих целей была разработана новая технология шифрования переписки MTProto, которая дополняется использованием проприетарной серверной части с закрытым кодом, доступ к которой не имеет никто, кроме Telegram.

Через Telegram можно отправлять не только сообщения, фото, видео, но и файлы любых форматов (doc, zip, mp3 и т. д.), а также создавать группы до 5000 человек и каналы трансляции новостей для неограниченной по размеру аудитории. Аккаунт пользователя привязывается к его контактному номеру, что позволяет быстро находить людей из телефонной книги. Поиск также возможен по username. Это делает Telegram похожим на SMS и e-mail, повышая таким образом степень удобства.

Telegram во многом отличается от других популярных мессенджеров, таких как WhatsApp. Главные отличия – это то, что, как уже говорилось, Telegram абсолютно бесплатный и, как утверждает его основатель Павел Дуров, таким и останется; не содержит реклам; использует облако в качестве хранилища, что позволяет синхронизировать данные на любом устройстве и при отсутствии необходимости не хранить медиафайлы непосредственно на устройстве, а в облаке. Самым же главным достоинством Telegram является предоставление удобной платформы с открытым кодом разработчикам для создания своих приложений, в число которых входят боты.

Что же такое Telegram боты? Боты – это сторонние приложения, которые работают внутри Telegram. По сути, они являются особыми

аккаунтами, которым не нужен телефон для регистрации. Пользователи могут взаимодействовать с ботами, отправляя им сообщения, команды и inline-запросы. Боты могут:

- получать настроенные уведомления и новости (Forbes Bot, TechCrunch Bot)
- интегрироваться с другими сервисами (Image Bot, Gif Bot, Wiki Bot, etc.)
- создавать игры для одного и нескольких пользователей (Game Bot, Gamee)
- организовывать социальные сервисы (BotOrHot)
- делать что угодно еще

Возможность создания ботов была быстро подхвачена многими сервисными службами (Viber, VK, Facebook). На данный момент – это один из главных трендов среди разработчиков. В помощь энтузиастам Telegram предоставил API для создания собственных приложений-клиентов и Bot API – специально для ботов. Процесс написания своего бота является достаточно трудоёмким и познавательным, в особенности для тех, кто впервые сталкивается с такими задачами, как отправление Интернет-запроса, j-son objects, WebHook, работа с СУБД, использование API и т. д. Кроме того, это очень интересно. Ведь Telegram дает вам полную свободу в реализации любой идеи. Именно поэтому данная тема была выбрана для курсовой работы.

Поставленные задачи и план их решения

Создание бота – это задача, решение которой выстраивается в несколько этапов. Необходимо:

- создать бота, зарегистрировав его через BotFather и с его помощью настроив основную информацию (имя, описание, возможности (inline?))
- выбрать язык программирования, возможности которого в наибольшей степени удовлетворяют поставленной задаче
- так как в качестве языка был выбран не изученный ранее Python, то возникла новая цель – его изучение
- определение принципов работы с Bot API
- написание скрипта
- внедрение (собственно хостинг бота на сервере)

Во время написания кода был сформирован совсем другой класс задач, связанных непосредственно со спецификой выполняемых ботом функций. Нужно было разработать схему будущей программы и выяснить все необходимые для её написания средства. В ходе проектирования была обнаружена потребность в сохранении данных о пользователе. Для решения данной проблемы наиболее эффективным подходом является использование базы данных, что и было решено реализовать. Соответственно, + 1 задача – поиск подходящей СУБД и обучение работе с ней.

Бот был назван GrammarCheckBot (@teachmeenglish_bot) в соответствии со своим непосредственным назначением.

Основная часть

Сравнение с аналогами

В ходе поиска аналогов для GrammarCheckBot был обнаружен только один – Andy English Bot.

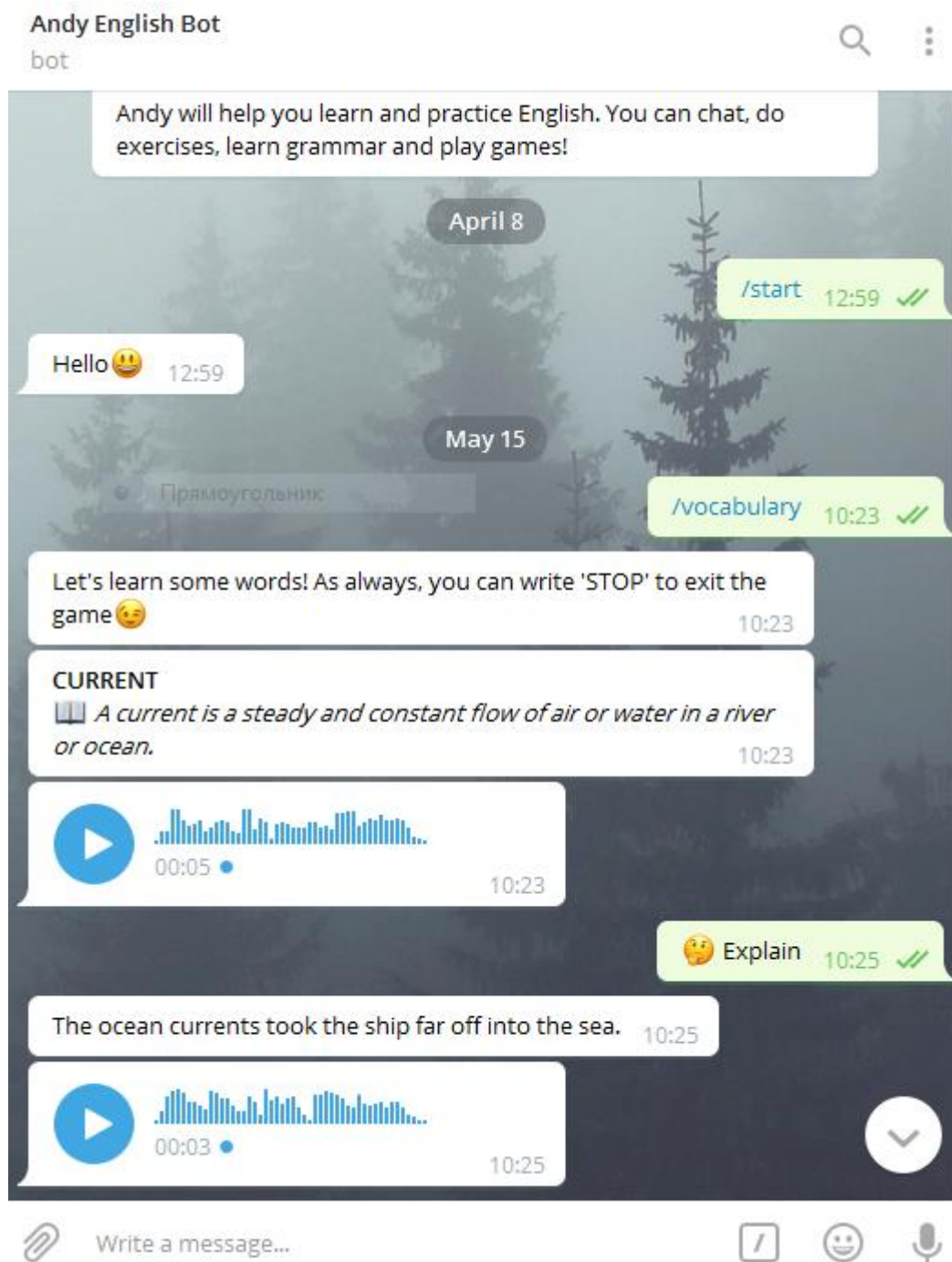


Рисунок 1. Andy Bot

Этот бот обладает следующим функционалом.









-  **/stats** your statistics
-  **/vocabulary** learn new words
-  **/grammar** practice grammar
-  **/play** play language games
-  **/talk** just talk, like people do
-  **/help** ask for help
-  **/settings** change learning schedule
-  **/stop** cancel any activity

Рисунок 2. Интерфейс Andy

Andy-бот предоставляет пользователю возможность изучать новые слова по определению, играть в «угадай слово по emoji», а также позволяет настраивать расписание, указывая как часто бот должен начинать общение самостоятельно. Это самые увлекательные возможности данного бота, всё остальное же такого восторга не вызывает. Как можно видеть, в списке есть команда «/grammar», которая отвечает за практику грамматики. Но на самом деле бот непосредственно не реализует этой функции, а лишь косвенно способствует этому: он предоставляет ссылки на скачивание соответствующих приложений из AppStore и PlayMarket.



Рисунок 3. "Grammar Check"

Таким образом, данный бот не обладает необходимой функциональностью, чтобы проводить тестирование пользователей непосредственно внутри чата.

Если вы хотите пройти тест, вы вынуждены совершить лишние действия, к тому же потратить память на своем устройстве на скачивание приложения. Этот недостаток восполняет GrammarCheckBot, который по специальной команде начинает тестирование, что очень просто и удобно. Плюс это не требует дополнительных затрат памяти, так как все необходимые данные хранятся в базе данных бота. Но в то же время стоит отметить, что GrammarCheckBot обладает значительно меньшим функционалом, чем Andy English Bot.

Таким образом, два данных бота взаимно незаменимы, так как имеют некоторые недостатки относительно друг друга и реализуют разные интерфейсы, хотя в целом имеют общую цель: обучение английскому языку. Значит, по функционалу GrammarCheckBot непосредственных аналогов не имеет.

Интерфейс и дизайн

Стоит отметить, что в процессе создания бота, всё-таки, есть те аспекты, которые Telegram разработчику контролировать не позволяет. В их число входит дизайн. Хотя, в таких случаях тоже есть выход – Telegram API, который даёт возможность создавать индивидуально-настроенные телеграм-клиенты. Но те, кого устраивает стандартный внешний вид мессенджера, могут ничего не менять. GrammarCheckBot выдержан в классическом стиле Telegram.

Помимо дизайна, нельзя варьировать основной интерфейс диалога: нельзя настраивать возможности панели ввода, изменять содержимое основного меню.

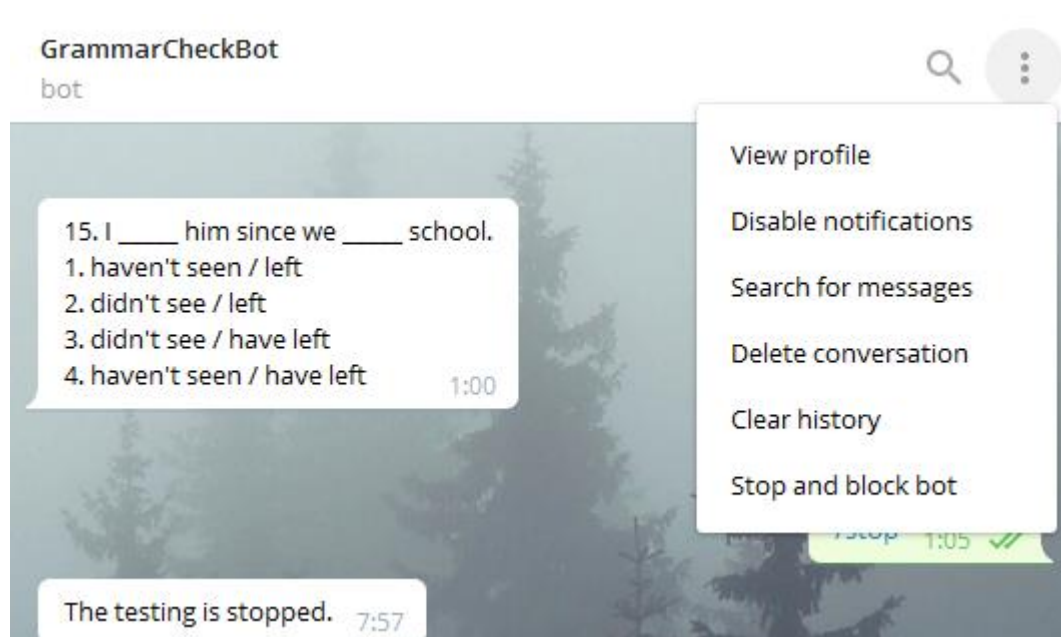


Рисунок 4. Стандартный вид меню в Telegram

Но всё же интерфейс бота зависит от его назначения. Его функционал сокрыт в небольшой кнопке на панели ввода, при нажатии на которую открываются все команды с их описанием.

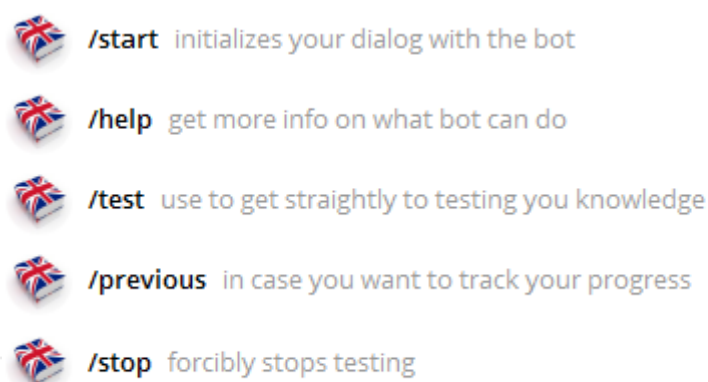


Рисунок 5. Доступные команды для бота GrammarCheckBot

Встроенной возможностью Telegram является функция Share, которая позволяет делиться ботами. При этом пользователь, которому бот был отправлен, получает специальное сообщение.

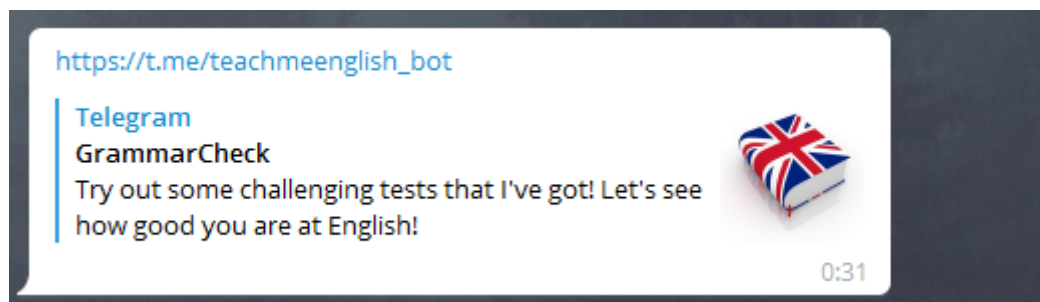


Рисунок 6. Bot's about info

Когда вы в первый раз открываете диалог с ботом, вы видите следующее.

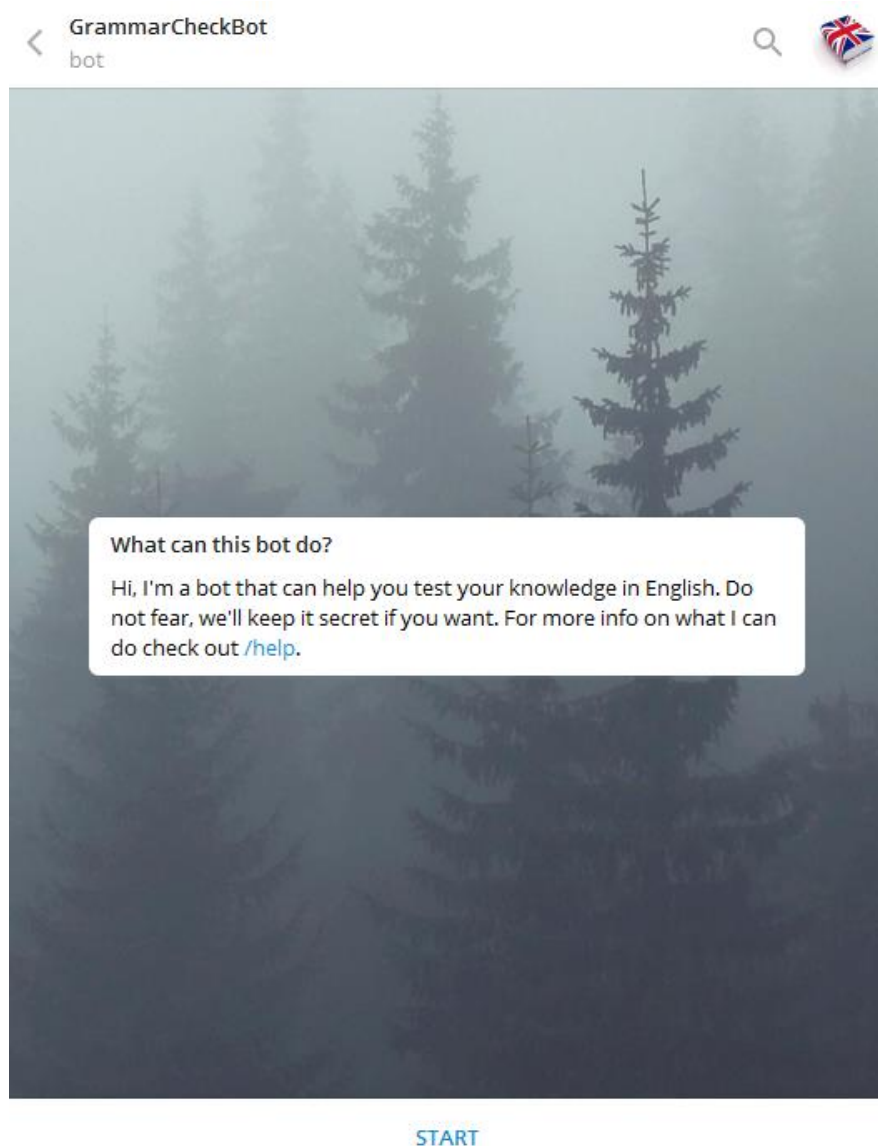


Рисунок 7. Начало диалога с ботом

Чтобы начать диалог, нужно нажать кнопку «start». Тогда будет отправлена команда 'start', которая инициализирует данные этого диалога у бота.

Чтобы получить подробное описание всех возможностей и самого бота, необходимо воспользоваться командой 'help'.

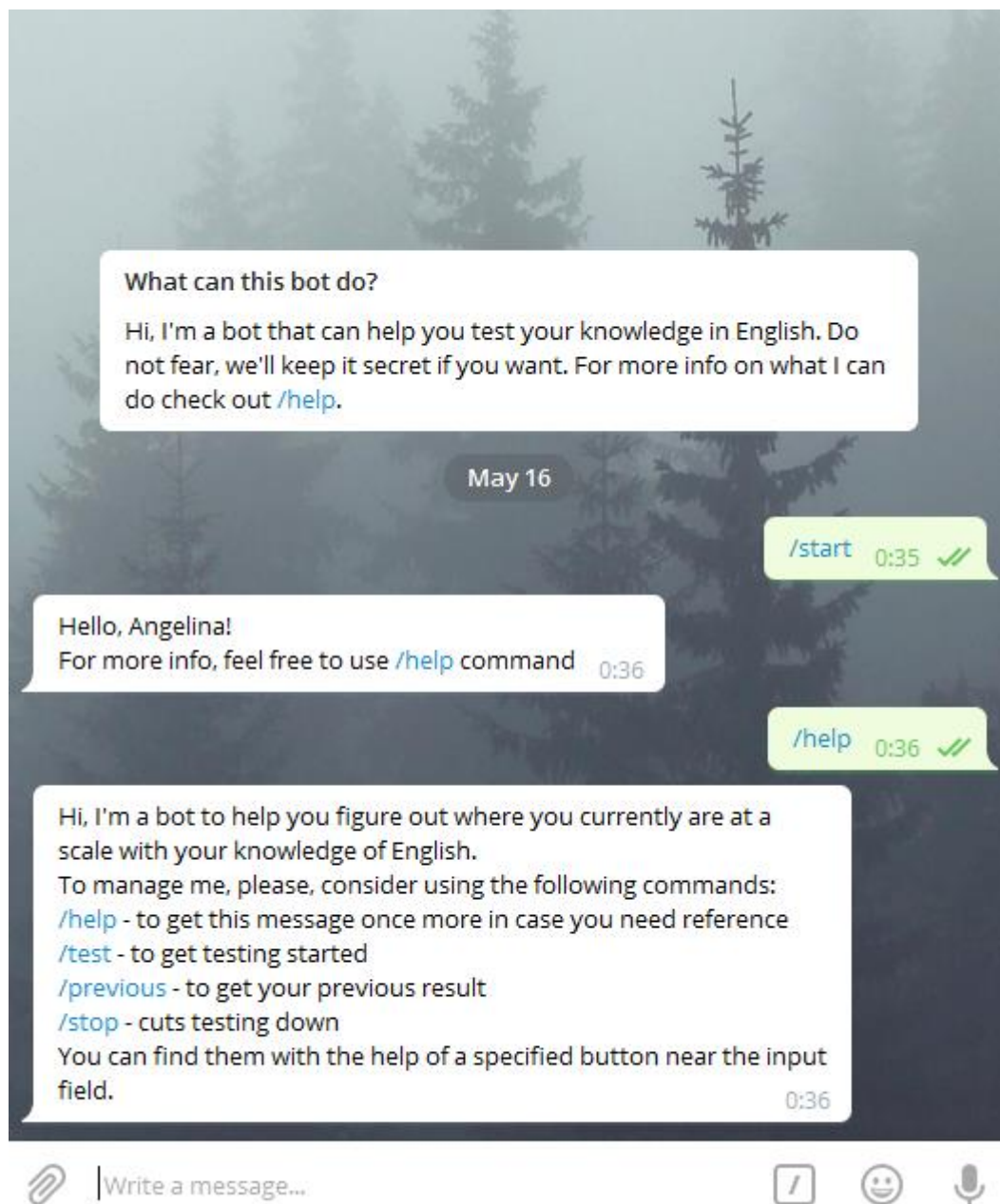


Рисунок 8. Демонстрация команд start и help

Другие команды из списка (см. рис. 2) – 'test', 'previous', 'stop'.

Команда 'test' запускает механизм тестирования. Команда 'stop' прекращает тест, если он имеет место.

Перед началом теста необходимо выбрать его уровень. Как только тест запущен, бот отправляет пользователю по одному вопросу с ожиданием

ответа. Всего вопросов в каждом из 3 тестов – 20. Для удобства бот посылает вместе с сообщением специальную клавиатуру.

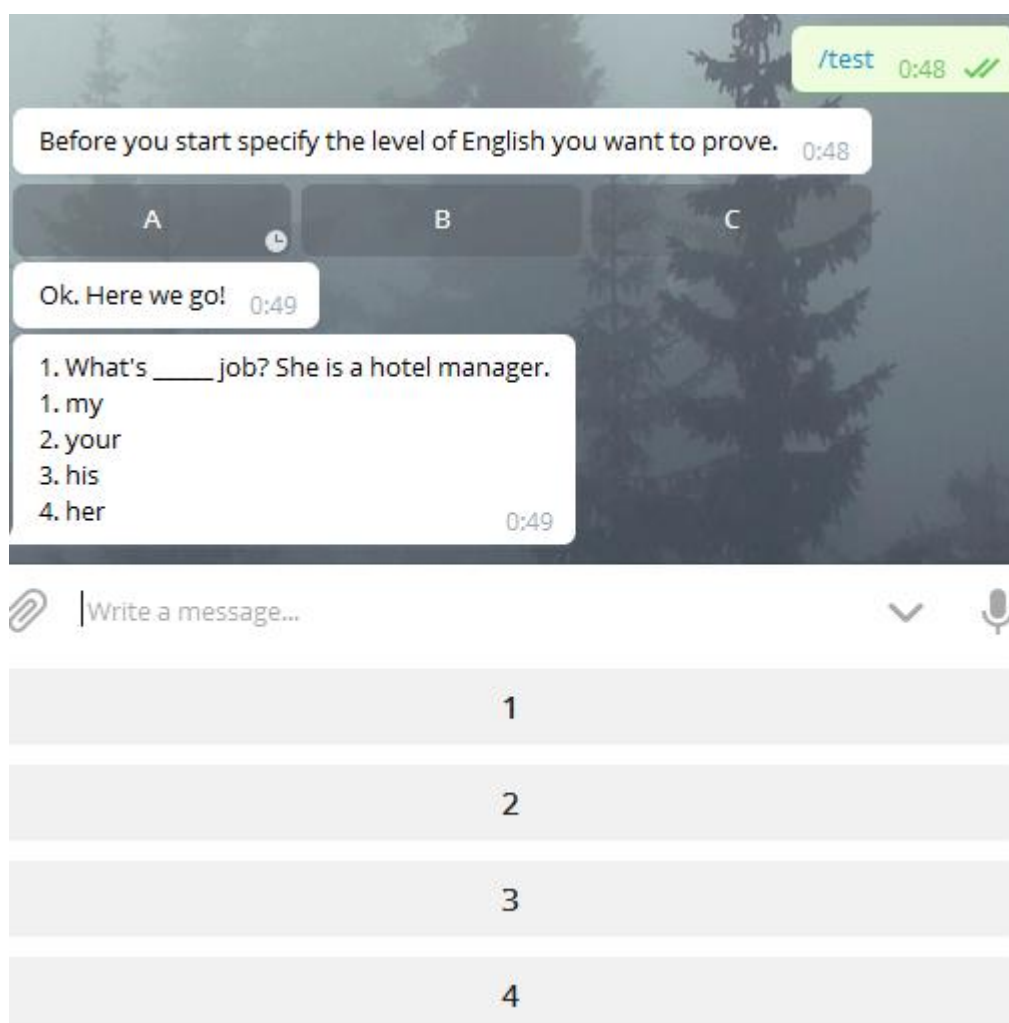


Рисунок 9. Начало теста

В случае неверного ввода ответа, бот сообщает об этом и просит ввести еще раз.

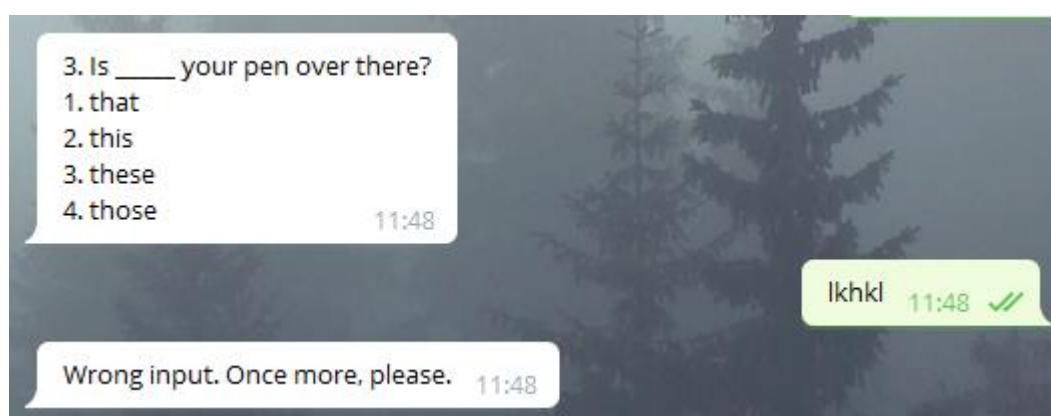


Рисунок 10. Неверный ввод

В случае полного прохождения теста пользователь получает свой результат, который содержит количество правильных ответов, их процент, а

также уровень, 1 или 2, в зависимости от того, превышает ли количество правильных ответов половину заданий. Также предлагается узнать правильные ответы. Хотя это нежелательно, так как лучше попытаться самому разобраться в своих ошибках и пройти тест заново.

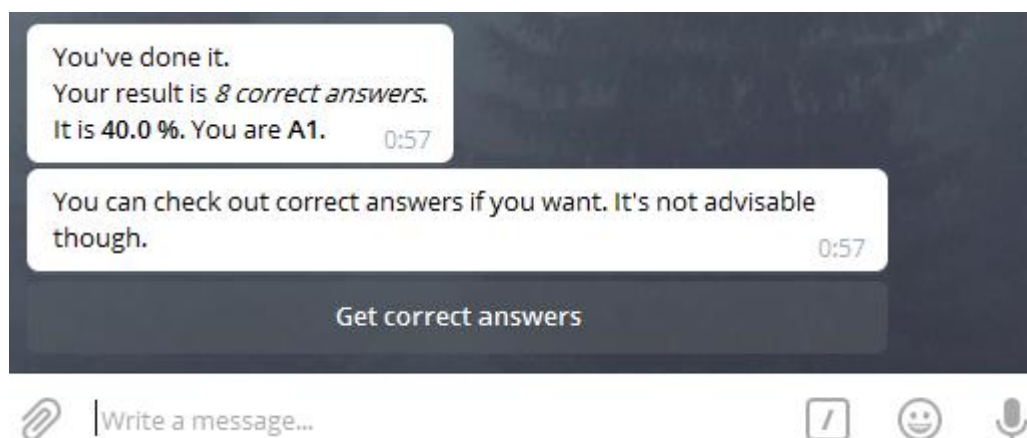


Рисунок 11. Тест пройден, результаты

В случае же, когда вы нажмете на кнопку «Узнать ответы», вы получите следующее сообщение.

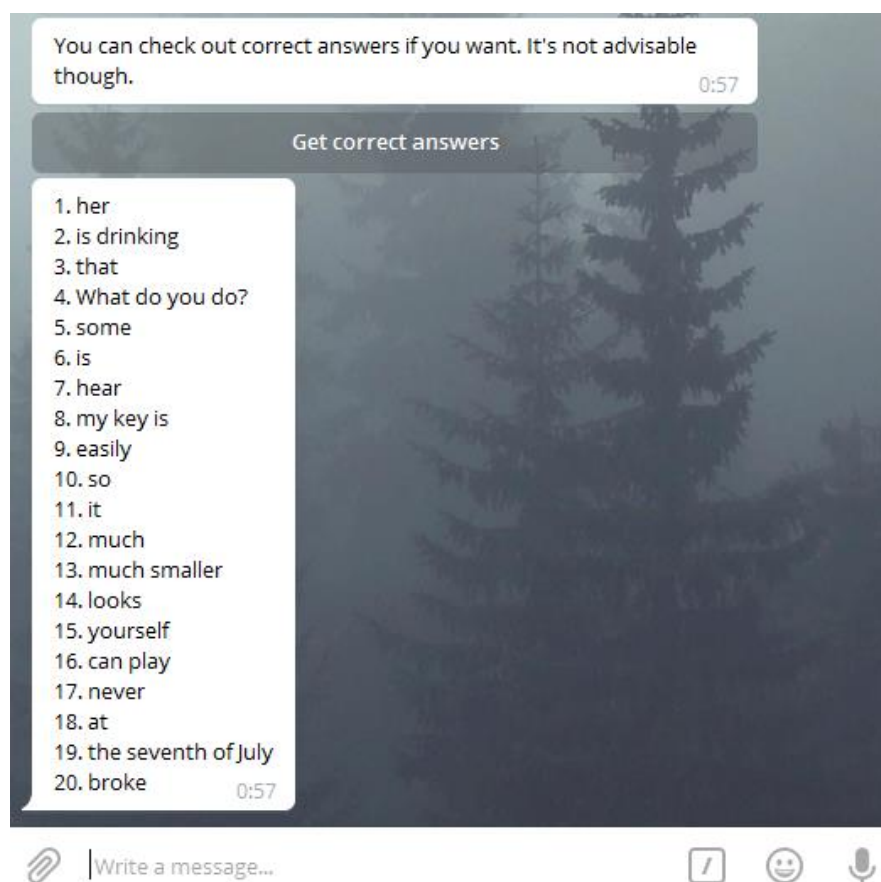


Рисунок 12. Правильные ответы

Есть еще одна команда – ‘previous’. Она возвращает пользователю список последних пройденных тестов с количеством правильных ответов на них.

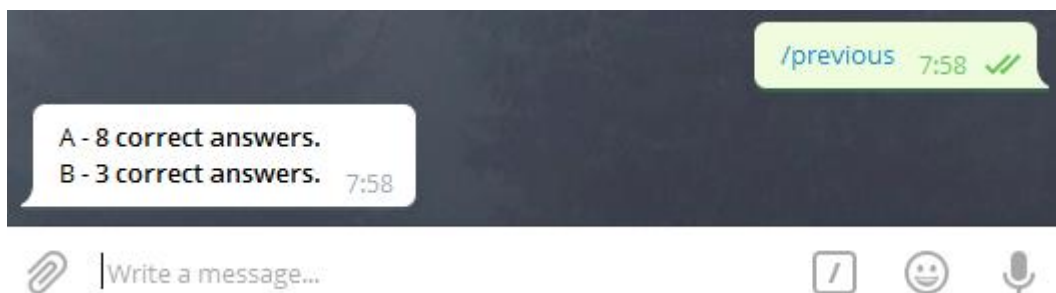


Рисунок 13. Предыдущие результаты

Техническая реализация

Создание бота в Telegram

Для того чтобы создать бота в Telegram, вам необходимо воспользоваться услугами специального бота, имя которого BotFather. Этот бот предоставляет интерфейс для удобного и быстрого управления ботами.

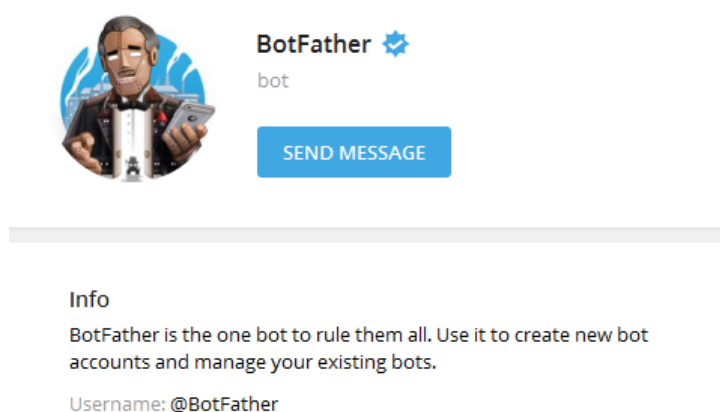


Рисунок 14. BotFather

Он предлагает следующий интерфейс.

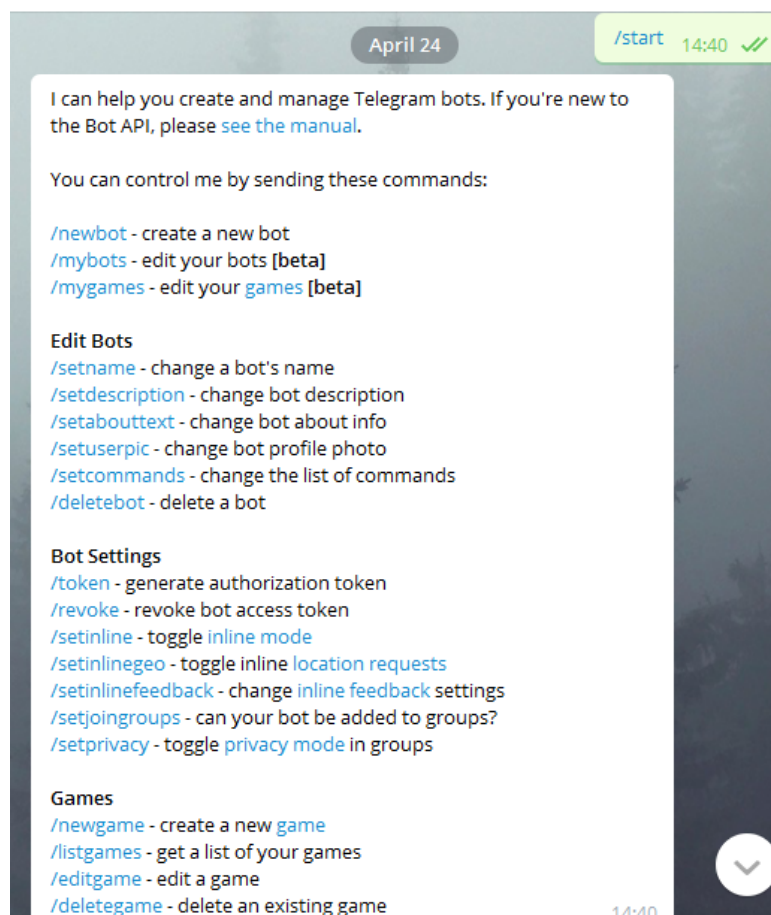


Рисунок 15. Интерфейс FatherBot

Как уже можно понять, бот создается с помощью команды 'newbot'. Когда вы вызываете эту команду, происходит следующее:

- бот-отец просит у вас имя для вашего бота, которое увидит пользователь в названии диалога
- затем бот-отец просит придумать @username для бота, который должен быть уникальным (в случае совпадения с username уже существующего бота FatherBot сообщит об этом и попросит придумать новый)
- в случае успеха вы получаете токен для бота – уникальный ключ, предоставляющий доступ к управлению ботом – и несколько дополнительных наставлений от бота-отца (токен на рис. 13 не будет действительный на момент сдачи записки)

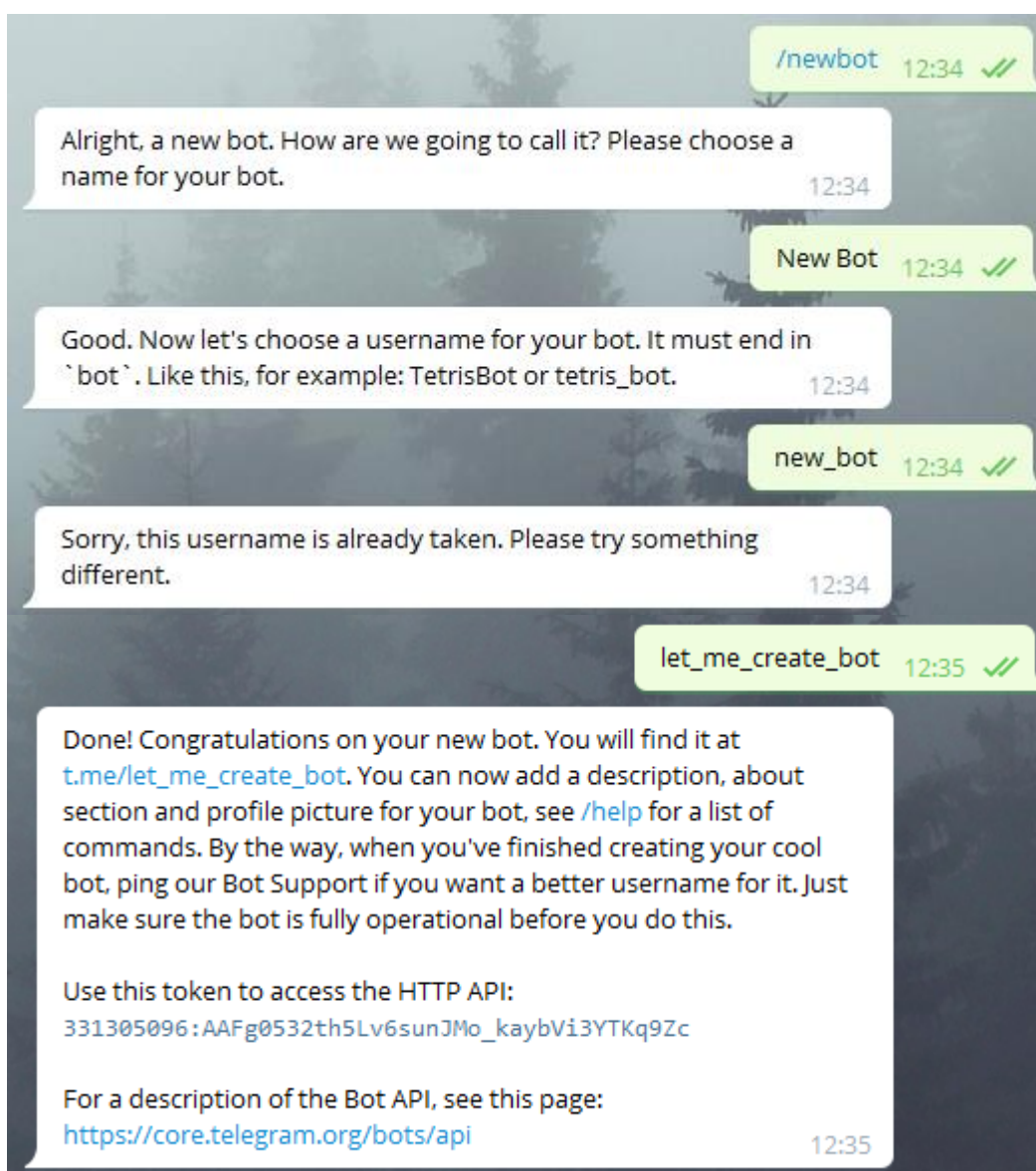


Рисунок 16. Создание бота

Всё, бот создан. Теперь надо научиться управлять им.

Telegram Bot API

Этот API позволяет разработчикам управлять ботами с помощью HTTP-запросов, которые представлены в следующей форме:

https://api.telegram.org/bot<token>/METHOD_NAME

В эту строку надо вставить непосредственно токен бота и имя метода, который бот должен выполнить, а также передать в метод все необходимые параметры. Существует 4 способа передачи параметров, один из них — использование URL-строки. Тогда после имени метода надо поставить знак вопроса и перечислить параметры, разделяя их знаком амперсанта.

Например, запрос «отправить сообщение» будет выглядеть так:

https://api.telegram.org/bot<token>/sendMessage?chat_id={id}&text={text}

Ответ на запрос содержит объект в формате JSON, в котором всегда есть булево поле 'ok', которое содержит true, если запрос выполнен успешно, и дополнительно есть поле 'description', в котором в случае ошибки содержится ее описание.

Telegram Bot API включает множество методов и типов данных, которые дают возможность разработчикам максимально удобно и полно реализовывать свои идеи.

Доступно более 20 типов (User, Chat, Message, Update и т.д.), столько же методов (sendMessage, sendFile, getChat и т.д.), а также 2 способа обновления сообщений: getUpdates и setWebhook. GrammarCheckBot был реализован с помощью первого метода, так как он более прост для выполнения, хотя является менее эффективным, чем второй. getUpdates получает все уведомления, доступные для данного бота с сервера Telegram. Получается, что придется с определенной частотой опрашивать сервер на наличие обновлений, что понижает скорость реакции бота. Метод setWebhook устанавливает вебхук (особый механизм получения обновлений) на сайт, url которого вы передаете в качестве параметра в метод. Это значит, что при наличии новых обновлений Telegram сам будет оповещать вас об этом, посылая обновления на тот самый сайт, который затем уведомляет вас о полученных данных. Как можно видеть, это более удобно, но от нас требуется наличие сайта, а для него необходим сервер. Так что, установить вебхук отнимает больше времени, чем GetUpdates, поэтому было решено использовать последний. Хотя вебхук предпочтительнее.

Более подробная информация о Telegram Bot API находится на официальном сайте Telegram.¹

¹ <https://core.telegram.org/bots/api>

Python

Важной составляющей реализации является язык программирования, от него зависит удобство программиста. Каждый язык — это совокупность средств, с помощью которых конструируется программа. Чем выше уровень языка, тем эффективнее идет работа по написанию кода в отношении времени.

GrammarCheckBot было решено написать на Python. Выбор был сделан в пользу этого языка по причине присущему ему ряда преимуществ.²

- Python - интерпретируемый язык программирования. С одной стороны, это позволяет значительно упростить отладку программ, с другой - обуславливает сравнительно низкую скорость выполнения.
- Динамическая типизация. В python не надо заранее объявлять тип переменной, что очень удобно при разработке.
- Хорошая поддержка модульности. Вы можете легко написать свой модуль и использовать его в других программах.
- Встроенная поддержка Unicode в строках. В Python необязательно писать всё на английском языке, в программах вполне может использоваться ваш родной язык.
- Автоматическая сборка мусора, отсутствие утечек памяти.
- Интеграция с C/C++, если возможностей python недостаточно.
- Понятный и лаконичный синтаксис, способствующий ясному отображению кода. Удобная система функций позволяет при грамотном подходе создавать код, в котором будет легко разобраться другому человеку в случае необходимости. Также вы сможете научиться читать программы и модули, написанные другими людьми.

Все это способствовало достаточно быстрому освоению нового языка и его свободному использованию при написании программы.

Для того чтобы начать писать код на Python необходим интерпретатор. Был установлен интерпретатор Python v. 3.6, который удобен для написания небольших модулей в целях изучения языка. Логика бота была написана в специальной IDE от компании JetBrains — PyCharm. Этот продукт бесплатный для студентов и очень удобный в использовании.

² <https://pythonworld.ru/>

pyTelegramBotAPI

Существует множество способов для использования Telegram Bot API. Их можно разделить на прямые и косвенные. Прямые способы реализуются через http-запросы, а косвенные – немного опосредованные. Они выполняются при помощи специальных написанных для Python модулей. Telegram предлагает несколько:

- Telepot
- twx.botapi
- pyTelegramBotAPI

Было решено использовать модуль для написания бота, потому что пользоваться модулями удобнее, чем работать непосредственно с Интернет-запросами. И способ написания бота никак не влияет на его возможности в конечном итоге, ведь все эти модули предоставляют тот же набор функций, что и Telegram Bot API, и реализуют их непосредственно через запросы, предоставляя разработчикам возможность взаимодействовать с API на более высоком уровне.

Из приведенных выше для работы был выбран модуль pyTelegramBotAPI, потому что по нему доступна более подробная документация и он включает методы для удобной работы с получением уведомлений.

Чтобы установить модуль, можно воспользоваться утилитой pip:

```
path>pip install pyTelegramBotAPI
```

Для начала работы необходимо импортировать данный модуль в программу.

```
import telebot
```

Инициализация бота происходит следующим образом:

```
bot = telebot.TeleBot(config.token)
```

“config.py” — это файл, в котором объявлены важные константы программы, такие как token, шаблоны ключей для базы данных.

pyTelegramAPI содержит реализацию всех функций из Bot API с абсолютно идентичным поведением. Он поддерживает все те же типы данных. Но названия функций претерпели небольшие изменения, чтобы соответствовать основному стайлгайду языка (PEP8). Например, sendMessage -> send_message. Набор параметров у методов же сохранился.

Чтобы заставить бота работать и получать обновления, необходимо воспользоваться функцией `polling()`.

```
bot.polling(none_stop=True)
```

На нижнем уровне эта функция создает новый поток, в котором вызывается функция получения обновлений. Полученные обновления рассылаются функциям `listeners` (принимают сообщения и обрабатывают их по заданному шаблону) и обработчикам сообщений.

В основе работы модуля лежат так называемые `handlers`, которые в зависимости от переданных в них параметров реагируют на отдельные классы сообщений и отвечают на них, выполняя функцию указанную после.

Вот пример `message_handler`, который отвечает на команду 'help' сообщением с полезной информацией, текст которого хранится в модуле "texts.py".

```
@bot.message_handler(commands=['help'])
def handle_help(message):
    bot.send_message(message.chat.id, texts.help_message)
```

Для бота были также использованы `callback handlers`, которые реагируют на нажатия клавиш встроенной в сообщения клавиатуры (`inline keyboard`). Работает он по принципу `message handler`, только обрабатывает не сообщения, а `callback query`.

Вот пример. Этот `handler` посылает пользователю правильные ответы на вопросы.

```
@bot.callback_query_handler(func=lambda call: "answers" in
call.data)
def get_answers_handler(call):
    level = call.data.strip("answers")

    i = 0
    text = ""
    while i < 20:
        i += 1
        correct =
database.get(config.correct_answer.format(level=level,
number=i))
        text += "{ }. ".format(i) +
database.hget(config.answers.format(level=level, number=i),
correct).decode() + "\n"

    bot.send_message(call.message.chat.id, text)
```

Как можно видеть из рис. 9, были использованы `custom keyboards`, а также возможности форматирования текста.

Чтобы отправить custom keyboard необходимо создать объект соответствующего типа (вся информация о типах содержится в модуле `types` модуля `telebot`), а также объекты, которые соответствуют клавишам и добавить их на клавиатуру, а потом отправить её пользователю вместе с сообщением, указав дополнительный параметр `reply_markup`. Например, для отправления Reply Keyboard нужно сделать следующее.

```
keyboard_markup =
types.ReplyKeyboardMarkup(resize_keyboard=True)
i = 0
while i != len(answers_str):
    keyboard_markup.add(types.KeyboardButton("{}".format((i +
1))))
    i += 1

bot.send_message(chat_id, message_text,
reply_markup=keyboard_markup)
```

Для форматирования текста можно использовать два типа разметок `html` и `Markdown`.

Markdown style

To use this mode, pass *Markdown* in the *parse_mode* field when using `sendMessage`. Use the following syntax in your message:

```
*bold text*
_italic text_
[text](http://www.example.com/)
`inline fixed-width code`
```text
pre-formatted fixed-width code block
```
```

Рисунок 17. Markdown Style

HTML style

To use this mode, pass *HTML* in the *parse_mode* field when using `sendMessage`. The following tags are currently supported:

```
<b>bold</b>, <strong>bold</strong>
<i>italic</i>, <em>italic</em>
<a href="http://www.example.com/">inline URL</a>
<code>inline fixed-width code</code>
<pre>pre-formatted fixed-width code block</pre>
```

Рисунок 18. HTML Style

Все это были основные средства Bot API, использованные для создания бота.

Redis – СУБД

Задачей бота GrammarCheckBot является тестирование знаний пользователя. Это подразумевает хранение вопросов и ответов в каком-либо формате. Способов для решения этой задачи можно придумать много, но самым удобным из них является использование базы данных, ведь эта структура хранения информации является максимально удобной для использования.

В качестве системы управления базой данных была выбрана СУБД Redis по двум главным причинам: она бесплатная и она является широкофункциональной. Данная система поддерживает многочисленные типы данных, имеет большим количеством методов, позволяющих размещать, получать и работать с элементами в базе данных, которую она устанавливает на компьютер. Она беспрепятственно скачивается через консоль на Linux, потому что изначально адаптирована для этого класса ОС. Для скачивания Redis на Windows был скачан установщик с github ресурса.³

Чтобы активировать систему, необходимо запустить локальный сервер (redis-server.exe), который также устанавливается системой. Затем все взаимодействие с БД происходит через клиент (redis-client.exe). Все данные после сохранения записываются в файл с расширением .rdb.

Чтобы обратиться к БД прямо из программы, нужно установить специальный модуль redis-py (устанавливается через pip, как уже показывалось ранее). Он поддерживает те же типы и интерфейс, что и Redis. Для его подключения необходима следующая строка кода:

```
import redis
```

Для работы с базой данной, как через клиент (параметры одинаковы для всех локальных БД Redis):

```
database = redis.StrictRedis(host='localhost', port=6379, db=0)
```

Для добавления элемента в базу данных нужно воспользоваться методом set.

```
set key value
```

Для получения значения ключа используется метод get.

```
get key
```

³ <https://github.com/MSSOpenTech/redis>

Это самый простой способ использования Redis. Есть более сложные структуры: хеш-таблицы, списки, множества, упорядоченные множества и т. д. Для работы с типом существует отдельный набор методов.

В процессе разработки были использованы такие типы данных, как множества и хеш-таблицы.

Для работы с множествами (неупорядоченные структуры данных, каждый элемент которых уникален) доступны методы:

`sadd key value` – добавляет значение в множество
`sismember key` – проверяет, является ли элементом множества (1 — да, 0 — нет)
`sdel key` – удаляет элемент из множества

Для работы с хеш-таблицами:

`hset name key value` – добавляет в таблицу name значение
`hdel name key` – удаляет элемент из таблицы
`hmset name [key value] x n` – добавляет несколько элементов
`hget name key` – возвращает значение элемента в таблице
`hgetall name` – возвращает список всех элементов

Для сохранения данных используется функция – `save`.

Логика бота

Программа состоит из 3 модулей: “bot.py” (логика), “texts.py” (большие сообщения), “config.py” (константы).

GrammarCheckBot, как и все другие боты Telegram, не может сам инициировать общение с пользователем, так как не имеет его id. Поэтому существует команда ‘start’, которая позволяет боту узнать о новом диалоге. Данный бот в ответ на эту команду отправляет приветствие и запоминает id пользователя, записывая его в базу данных.

```
@bot.message_handler(commands=['start'])
def handle_start(message):

    bot.send_message(message.chat.id, "Hello,
{name}!\n".format(name=message.from_user.first_name) +
                    "For more info, feel free to use /help
command")

    #adds a user's id to the database if it's there yet
    if database.sismember("users", message.from_user.id) == 0:
        database.sadd("users", message.from_user.id)
        database.save()
```

В ответ на команду ‘help’ он отправляет сообщение с необходимой информацией.

```
@bot.message_handler(commands=['help'])
def handle_help(message):

    bot.send_message(message.chat.id, texts.help message)
```

Команда ‘test’ — создает встроенную клавиатуру с названиями уровней тестов, просит пользователя сделать выбор.

```
@bot.message_handler(commands=['test'])
def handle_test(message):

    #shaping custom inline keyboard
    inline_markup = types.InlineKeyboardMarkup()
    btnn1 = types.InlineKeyboardButton("A", callback_data="A")
    btnn2 = types.InlineKeyboardButton("B", callback_data="B")
    btnn3 = types.InlineKeyboardButton("C", callback_data="C")
    inline_markup.add(btnn1, btnn2, btnn3)

    text = "Before you start specify the level of English you want to prove."
    bot.send_message(message.chat.id, text, reply_markup=inline_markup)
```

Затем обрабатывается callback query, вызванный нажатием на встроенную клавиатуру. В базу записывается уровень текущего теста, номер текущего вопроса и счет. Посылается первый вопрос теста.


```
@bot.callback_query_handler(func=lambda call: True)
def call_handler(call):

    #announcing test's start
    bot.send_message(call.message.chat.id, "Ok. Here we go!")

    #getting some helpful buffers on the go and sending the first question of
    the test (level)
    database.set("{user_id}.question".format(user_id=call.from_user.id), 1)
    database.set("{user_id}.level".format(user_id=call.from_user.id),
call.data)
    database.set("{user_id}.score".format(user_id=call.from_user.id), 0)
    send_question(call.message.chat.id, call.data)
    database.save()
```

Функция отправки вопроса посылает вдобавок клавиатуру. Если введен ответ, не принадлежащий списку предложенных, то бот не переходит на следующий вопрос.

Дальше человек отправляет ответы, а бот следит за текущим вопросом. Если все вопросы закончились, то он очищает поле в базе, хранящее текущий вопрос и уровень. Подсчитывает результаты и выводит их. А также предлагает узнать ответы на вопросы (новый callback query).

```
#getting final score
score =
int(database.get("{user_id}.score".format(user_id=message.from_user.id)))
percentage = score * 100 / 20

#sending result
deduction = "You are *{}*.".format(question_level, 1 if percentage < 50
else 2)
text = texts.result.format(score, percentage, deduction)
keyboard = types.ReplyKeyboardRemove(selective=False)
bot.send_message(message.chat.id, text, reply_markup=keyboard,
parse_mode="Markdown")
```

Команда 'previous' — проверяет базу и узнает последние результаты ранее пройденных тестов.

```
@bot.message_handler(commands=['previous'])
def handle_previous(message):

    #if there is no info about passed tests
    if len(database.hgetall(message.from_user.id)) == 0:
        bot.send_message(message.chat.id, "You've not passed a test yet.")

    #gets all not-empty fields containing previous results and forms a string
    levels = ["A", "B", "C"]
    score = ""
    for level in levels:
        result =
database.hget("{user_id}".format(user_id=message.from_user.id), level)
        if result is not None:
            score += level + " - " + "*{} correct
answers.*".format(result.decode()) + '\n'
    bot.send_message(message.chat.id, score, parse_mode="Markdown")
```

Заключение

В результате работы был создан бот, который позволяет пользователям пройти тест по английскому языку прямо внутри чата и получить оценку своих знаний. Данный бот на данный момент не имеет точных аналогов, а также, как и планировалось, дополняет интерфейс многофункционального бота Andy English Bot, который не предоставляет непосредственной возможности тестирования. Теперь любой пользователь может беспрепятственно пользоваться данным ботом без необходимости скачивать сторонние приложения или переходить на вебсайт, особенно с мобильного устройства. Интерфейс данного бота прост, оттого и удобен.

В итоге, были реализованы все задачи, за исключением отправления скрипта бота на сервер, чтобы тот мог функционировать 24/7. Но это может быть в будущем решено. Есть много сервисов, которые обеспечивают данные услуги за небольшую цену, а тем более – бесплатные. Например, платформа HeroKu, которая предоставляет 600 часов работы приложения каждый месяц без оплаты.

В дальнейшем хотелось бы дополнить функционал бота, сделать его более интерактивным. Исходя из перспективы на будущее, бот сохраняет id всех пользователей, когда-либо инициировавших общение с ним, чтобы в случае пополнения базы данных тестов уведомлять их об этом. Он также сохраняет ответы пользователей на тест, чтобы потом, при обнаружении ошибки в тесте, пересмотреть результаты его прохождения.

Данная курсовая работа послужила толчком к изучению новых технологий, таких как Интернет-запросы, API, Python, базы данных, что дало достаточно знаний для успешного выполнения поставленных задач. Помимо этого, все это ляжет в основу продолжения процесса самообразования, поможет увереннее смотреть на сложные задачи и неизвестные понятия.

Большим подспорьем во время обучения стали широкие возможности поисковых сервисов. В современном мире, особенно будучи частью информационного общества, необходимо уметь находить и использовать все те знания, которые вас окружают. Этому сейчас способствуют многие сервисы и службы. Нужно лишь только обладать любопытством, целеустремленностью и желанием учиться. Это есть залог развития. Развитие есть движение. Движение есть прогресс.

Литература

Telegram

Bots

<https://core.telegram.org/bots>

Bot API

<https://core.telegram.org/bots/api>

Modules

<https://core.telegram.org/bots/samples>

pyTelegramAPI

<https://github.com/eternnoir/pyTelegramBotAPI>

Telegram Group for Bot Developers

<https://t.me/pythontelegrambotgroup>

First Steps

<https://groosha.gitbooks.io/telegram-bot-lessons/content/chapter8.html>

Python

Course

<https://programming086.blogspot.com.by/2015/12/python-2015.html>

Documentation

<https://docs.python.org/3.6/tutorial/index.html>

Beginner's guide

<https://pythonworld.ru/>

Nice tutorial

<http://www.python-course.eu>

Redis

Tutorial (interactive client)

<http://try.redis.io/>

Official documentation

<https://redis.io/>

Setup

<https://github.com/MSOpenTech/redis>

Module

<https://pypi.python.org/pypi/redis>

Commands

<https://redis.io/commands>

Types

<https://redis.io/topics/data-types-intro>

Supplementary useful resources

A guy with a boat

https://www.youtube.com/channel/UCoXbChoGT-_tVQVK3nWgY1g

Requests

<https://www.youtube.com/watch?v=iMBuy0INnHQ&list=PLu5rqx-k8WFEVP8MWyZj2AOYt36ZajJAw&index=2&t=2394s>

Wikipedia

Tests

<http://www.study.ru/test/testlist.php?id=172>