МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

РАБОТА ПРОВЕРЕНА

Рецензент
Ведущий программист
ООО «ВОРТЕКСКОД»

_____ П.А.Михайлов

“___”_____ 2020 г.

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой,
д.ф.-м.н., профессор

_____ Л.Б. Соколинский

“___”_____ 2020 г.

Разработка чат-бота для изучения академического английского языка

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 02.03.02.2020.308-017.ВКР

Научный руководитель,
ст. преподаватель кафедры СП
_____ К.Ю. Никольская

Автор работы,
студент группы КЭ-402
_____ М.А. Лех

Ученый секретарь
(нормоконтролер)
_____ И.Д. Володченко
“___”_____ 2020 г.

Челябинск-2020

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

УТВЕРЖДАЮ
Зав. кафедрой СП
_____ Л.Б. Соколинский
03.02.2020

ЗАДАНИЕ
на выполнение выпускной квалификационной работы бакалавра
студенту группы КЭ-402
Лех Маргарите Андреевне,
обучающейся по направлению
02.03.02 «Фундаментальная информатика и информационные технологии»

1. Тема работы (утверждена приказом ректора от 24.04.2020 № 627)
Разработка чат-бота для изучения академического английского языка.
2. Срок сдачи студентом законченной работы: 05.06.2020.
3. Исходные данные к работе
3.1. API для чат-ботов. [Электронный ресурс] URL: https://vk.com/dev/bots_docs (дата обращения 11.05.2020).
3.2. Видеокурс по боту для вк. [Электронный ресурс] URL: https://www.youtube.com/watch?v=1H31pfxNGdc&list=PL6eAa1p_LgucLFlOzgpWLNHZoxVVK1-0A (дата обращения 11.05.2020).
4. Перечень подлежащих разработке вопросов
4.1.Выполнить анализ предметной области и произвести обзор существующих решений.
4.2.Спроектировать чат-бот.
4.3.Реализовать чат-бот.
4.4.Провести тестирование разработанного чат-бота.
5. Дата выдачи задания: 03.02.2020.

Научный руководитель
ст. преподаватель кафедры СП К.Ю.Никольская

Задание принял к исполнению М.А. Лех

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

ОСНОВНЫЕ ОПРЕДЕЛЕНИЯ

Чат-бот – программа, которая общается с пользователем на естественном языке, понимает и выполняет требования пользователя с помощью подручных средств [1].

Мессенджер – программа, мобильное приложение или веб-сервис для мгновенного обмена сообщениями.

Социальная сеть – интерактивные онлайн-сервисы, созданные для общения и обмена информацией группами пользователей.

Академический английский – это раздел языка, который используется для написания научных статей, книг, а также ведения прочей официальной документации и официального общения в рамках учебного заведения.

АКТУАЛЬНОСТЬ ТЕМЫ ИССЛЕДОВАНИЯ

Английский язык активно изучается как иностранный во многих учебных заведениях по всему миру. Студенту, обучающемуся за границей или же просто желающему найти дополнительную информацию для своей научной работы, необходимо знать академический английский язык. Именно академический английский необходим для более глубокого понимания интересующего материала. Этот раздел языка, используемый для описания абстракций научного материала, включает различные грамматические конструкции и лексикон, относящийся к самым разным сферам образования. Именно эти две составляющие считаются отличительными особенностями академического английского языка, которыми необходимо овладеть обучающемуся [2].

Одним из ключевых факторов в успешном овладении любого иностранного языка, не только английского, является погружение в языковую среду. Это в свою очередь подразумевает создание условий, при которых человек так или иначе сталкивается с необходимым для изучения и закрепления учебным материалом. Однако, далеко не у всех есть возможность обучаться за границей или же контактировать с англоязычной средой

на постоянной основе. С распространением Интернета и социальных сетей общение и обучение стало возможным с помощью онлайн приложений [3].

Чат-бот является довольно перспективным обучающим приложением, интерес к которым стабильно растет, как видно из графика, представленного на рисунке 1 [4].

Рис. 1. График запросов по чат-ботам с июня 2015 г. по апрель 2020 г.

Среди ключевых особенностей чат-бота, обуславливающих их популярность, можно выделить описанные ниже возможности.

1. Постоянная доступность. Пользователь может использовать чат-бот в любое удобное ему время.

2. Мгновенный отклик. Пользователю не нужно ожидать продолжительное время, для того чтобы получить ответ на свой запрос.

3. Независимость от платформы. Чат-боты, представленные в социальных сетях, предоставляют одинаковый функционал как в браузерах персональных компьютеров, так и в мобильном приложении социальной сети [5].

Стоит заметить, что из различных способов обучения новому языку, регулярное пополнение словарного запаса и запоминание ключевых грамматических конструкций и устойчивых выражений не требуют наличия живого общения с носителем языка [6]. Таким образом, даже чат-бот с самым базовым функционалом может быть полезен чат-бот при изучении академического английского. Помимо этого, чат-бот может стать вспомо-

гательным техническим средством, позволяющим пользователю изучать язык, в любое время и в любом месте.

ЦЕЛИ И ЗАДАЧИ ИССЛЕДОВАНИЯ

Целью данной работы является разработка чат-бота для изучения академического английского языка.

Для разработки чат-бота необходимо решить следующие задачи.

1. Выполнить анализ предметной области и произвести обзор существующих решений.

2. Спроектировать чат-бот.

3. Реализовать чат-бот.

4. Провести тестирование разработанного чат-бота.

СТРУКТУРА РАБОТЫ

Работа состоит из введения, четырех глав и заключения. Объем работы составляет 62 страницы, объем библиографии – 18 источников.

СОДЕРЖАНИЕ РАБОТЫ

В первой главе проводится анализ предметной области, проведен анализ существующих решений, приведены примеры аналогов.

Во второй главе рассмотрены варианты использования чат-бота и описаны его.

В третьей главе описаны средства реализации и рассмотрены методы реализации проекта чат-бота.

В четвертой главе описан процесс тестирования разработанного бота и приведен протокол тестирования некоторых аспектов его работы.

В заключении приводятся основные результаты работы.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И СУЩЕСТВУЮЩИХ РЕШЕНИЙ

1.1. Технологии разработки чат-ботов

На данный момент выделяют два типа чат-ботов, разделяемых по принципу работы. Первый тип интерпретирует введенное пользователем предложение, основываясь на наборе правил, заложенных в него при разработке. Такой чат-бот может отвечать на определенные команды. В случае если введенная команда не соответствует правилам, бот может не понять пользователя. Второй тип основан на искусственном интеллекте и технологии машинного обучения, что позволяет ему предоставлять собеседнику наиболее подходящий ответ. На самом деле, также существуют чат-боты с большим функционалом для взаимодействия, чем первый тип, но менее способные к имитации живого собеседника, чем второй тип [7].

Тем не менее, вне зависимости от типа чат-бота выделяют несколько технических методов и средств, на которые опирается большинство разработанных чат-ботов. Данные правила приводятся ниже.

1. Парсинг (синтаксический анализ информации) – это метод, подразумевающий анализ введенного пользователем текста и дальнейшее его использования с помощью ряда функций НЛП (нейролингвистическое программирование).

2. Сопоставление с образцом – это метод, использующийся во многих чат-ботах, в частности, в системах типа «вопрос-ответ», где устанавливается соответствие введенного текста с запросами на естественном языке, простыми предложениями и смысловой нагрузкой запросов.

3. AIML (Artificial Intelligence Modelling Language) – это универсальный язык разметки, основанный на XML, который представляет введенные пользователем данные как AIML-объекты. Эти объекты характеризуются с помощью тем и категорий. В зависимости от паттрена полученного объекта по определенному языковому шаблону формируется ответ. Таким обра-

зом, каждый объект AIML является своеобразным тегом, который ассоциируется с языковой командой.

4. Скрипт чата – это метод, который необходим, если не получается найти соответствия с помощью AIML. В данном случае подбирается наиболее подходящая синтаксическая структура, формирующая ответ по умолчанию. Данный подход основан на понятиях переменных, фактической информации и логических и / или.

5. SQL и реляционные базы данных как средства начали использоваться в чат-ботах для хранения предыдущих диалогов с пользователем. Это позволяет вести более продолжительные и осмысленные диалоги. Базы данных также упрощают установление соответствия введенных данных с заложенными в систему.

6. Цепи Маркова используются в чат-ботах для построения ответов в зависимости от вероятности их использования, что позволяет определить наиболее применимый к ситуации, а, следовательно, правильных ответ для пользователя. Идея цепей основана на том, что существует определенная вероятность, с которой некая буква или слово появится в одном и том же наборе текстовых данных.

7. «Подводные камни» языка – это фразы, предложения и целые абзацы, которые должны разнообразить речь чат-бота, тем самым сделав ее более убедительной и «живой». К ним относятся стандартные ответы, ошибки правописания, имитация собственной биографии, алогичные ответы.

8. Онтологии (семантические связи) – это набор взаимосвязанных концепций, необходимых для подбора синонимов, гипонимов, и иных средств естественного языка. Эти наборы могут быть представлены в виде графов, что позволяет искать взаимосвязи, основываясь на определенном наборе правил [8].

Тем не менее, стоит отметить, что большинство современных ботов – это чат-боты первого типа, которые предоставляют пользователю меню (в

текстовом виде или в виде кнопок для выбора), благодаря чему и происходит взаимодействие с пользователем. Более того, эти боты также сосредоточены на одной конкретной задаче, которая заключается в предоставлении автоматизированного сервиса на основе диалогового интерфейса [9].

1.2. Анализ аналогичных проектов

Duolingo [10]

Duolingo – это условно бесплатное приложение, доступное на мобильном телефоне либо из браузера на персональном компьютере, которое направлено на изучение различных иностранных языков. Это приложение считается одним из самых популярных приложений (в Google Play, например, у него больше 100 млн. скачиваний).

Одной из отличительных особенностей данного приложения являются направленность на развитие всех четырех языковых компетенций иностранного языка, а именно: чтение, письмо, аудирование и говорение. Овладение этими компетенциями возможно благодаря различным типа приложений:

– набор текста, озвученного встроенным в приложение ассистентом;

– составление слов в нужном порядке;

– перевод слов или фраз как с изучаемого языка на базовый, так и наоборот (в виде набора текста);

– перевод слов или фраз как с изучаемого языка на базовый, так и наоборот (в виде перетаскивания лексем, как представлено на рисунке 2);

– произнесение слов или фраз (при этом происходит распознавание речи);

– выбор пропущенного слова в предложение.

Другой отличительной особенностью является возможность изучения множества иностранных языков. Однако, в зависимости от базового языка пользователя будет варьироваться и количество возможных для изучения иностранных языков.

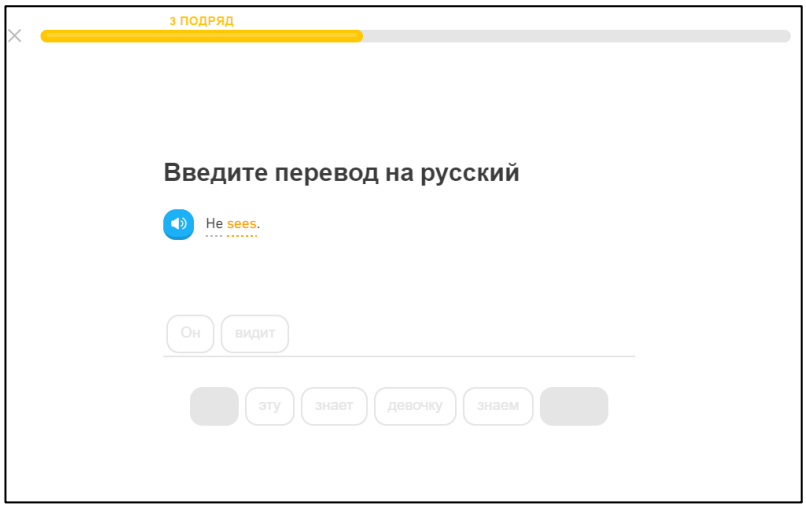

Рис. 2. Упражнение на перевод с английского на русский в веб-приложении Duolingo

Система обучения в Duolingo построена на прохождении набора заданий в рамках одного раздела, после которого открывается доступ к следующим разделам обучения (на рисунке 3). Также в качестве поощрения за пройденный материал в приложении есть система очков, которые впоследствии можно потратить на различные бонусы.

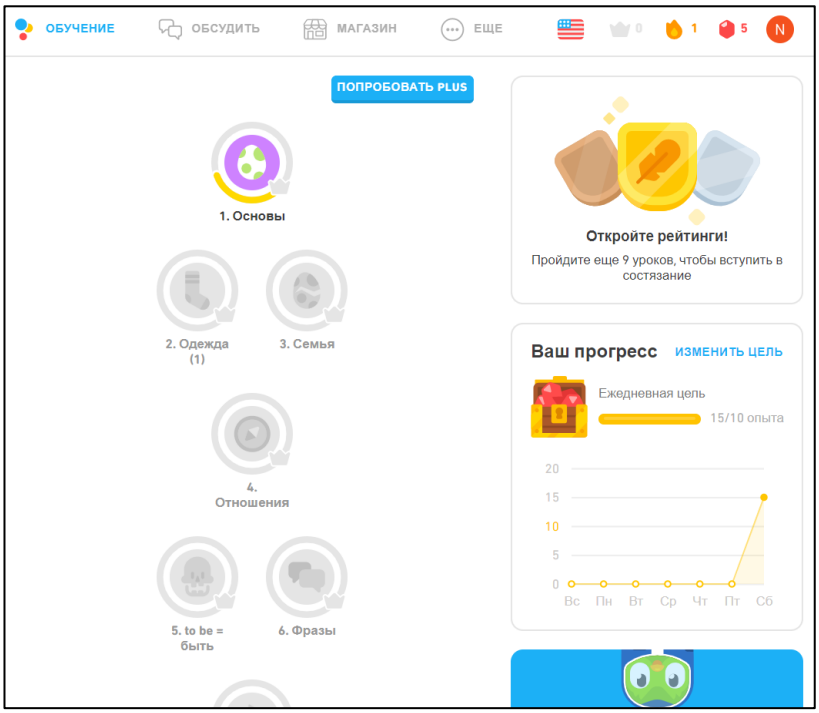

Рис. 3. Наборы заданий в разделе обучения в веб-приложении Duolingo

Кроме того, перед непосредственным обучением, пользователю предоставляется возможность пройти тест, для определения уровня. По результатам этого теста пользователю могут сразу открыться несколько разделов обучения.

Среди недостатков Duolingo можно выделить ограниченный набор заданий, фраз и слов. Хоть количество упражнений и разнообразно, многие из них основаны на повторении одних и тех же языковых конструкций, что сильно затормаживает процесс обучения. Стоит также отметить, что лексикон упражнений в основном состоит из базового набора слов и фраз, то есть не относится непосредственно к академическому английскому.

Помимо этого, определенное неудобство доставляет откат прогресса пользователя при отсутствии ежедневной активности в приложении. Этот откат может быть произведен вплоть до первого урока, даже несмотря на наличие у пользователя первоначально высокого уровня знания языка. При этом все ранее пройденные уроки могут снова заблокироваться.

Dragon-English [11]

Сайт Dragon-English предоставляет пользователю доступ к системе обучения, основанной на базе рассчитанного на 15 недель курса английского языка.

Как и Duolingo данная система обучения основана на развитие всех языковых компетенций английского языка. Однако обучение построено на иных принципах.

В начале каждого курса пользователю предоставляется видео, в котором преподаватель объясняет теоретический материал, необходимый для выполнения практических занятий. После чего пользователь может приступить к практике.

Сама практика подразумевает два типа упражнений, который, тем не менее, направлен на все четыре аспекта изучения языка. Пользователь тренирует навык письма путем написания переведенного на английский язык предложения либо заполнения пропущенных в предложении слов. В свою

очередь тренировочное предложение включает в себя как элементы грамматики, так и элементы лексики, необходимых для освоения и предоставляемые в теоретической части. Помимо этого, тренировочное предложение можно прослушать, а затем ввести в текстовое поле, тренируя навык аудирования. В случае если пользователь не может составить перевод и не понимает предложение на слух, ему может быть предоставлено частично составленное предложение, которое необходимо дополнить.

Однако овладение навыком произношения основано на самоконтроле пользователя. Он может записать собственное произношение предложения и затем прослушать его, чтобы сравнить с предоставленным оригиналом.

Введенное пользователем предложение проходит проверку и, в случае неверного ответа, ему объясняются возможные ошибки. Также пользователь может посмотреть ошибки совершенные другими пользователями. Результат проверки одного упражнения и пример следующего упражнения приведены на рисунке 4.

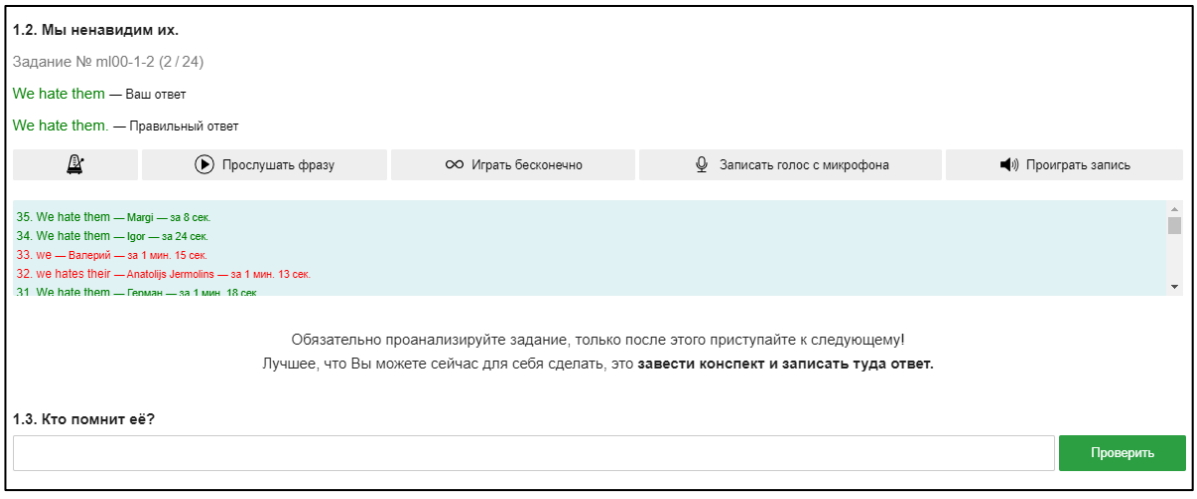

Рис. 4. Интерфейс системы обучения Dragon-English

Еще одним достоинством данной системы является усложнения как грамматики, так и изучаемого лексикона, по мере прохождения различных уроков. Однако и это система обучения прежде всего опирается на словарный запас, необходимый в повседневной жизни.

Среди недостатков данной системы обучения прежде всего стоит выделить тот факт, что она является платной, хотя пользователю и предоставляется пробная версия из пяти первых уроков.

Помимо этого, данная система обучения не имеет мобильного приложения и предоставляется только на базе веб-сайта, что не всегда может быть удобно. Интерфейс системы обучения, открытой в браузере мобильного телефона, представлен на рисунке 5.

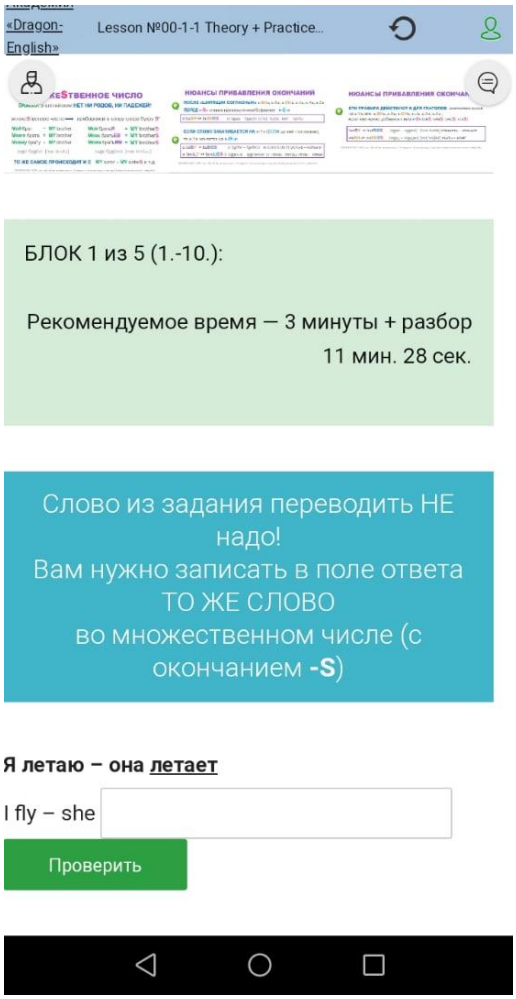

Рис. 5. Вид системы обучения Dragon-English в браузере мобильного телефона

Также стоит отметить, что эта система обучения подразумевает наличие минимального базового уровня у пользователя, желающего пройти курс.

1.3.Анализ существующих решений для реализации проекта

Facebook [12]

Социальная сеть Facebook предоставляет платформу Messenger для разработки чат-бота. Платформа идентифицирует каждого пользователя с помощью специального идентификатора PSID (идентификатор пользователя, действующий в пределах страницы – page-scoped ID), уникального в рамках конкретного канала связи. Этот идентификатор гарантирует, что пользователь будет общаться лишь с тем чат-ботом, с которым инициировал переписку. Он также позволяет компании использовать для одного клиента единый PSID по всем каналам связи.

Чат-бот в социальной сети Facebook предоставляет возможность отправлять не только текстовые сообщения и использовать функцию быстрых ответов, а также обмениваться аудио, видео, изображениями и файлами.

Платформа Messenger позволяет в любой момент перейти от переписки с чат-ботом к общению с реальным представителем компании с помощью API Handover Protocol.

На рисунке 6 представлен интерфейс чат-бота для кадрового агентства, созданный в социальной сети Facebook.

Viber [13]

Создавать функционального чат-бота в социальной сети Viber стало возможным при помощи публичного аккаунта (Viber public account), который был представлен как специальный канала связи компаний со своими клиентами.

Переписка с чат-ботом возможна в том случае, если пользователь подписан на аккаунт. Однако Viber предоставляет аккаунту возможность послать одно приветственное сообщение для пользователя тем самым инициируя диалог до того, как пользователь подпишется.

Бот способен распознавать любые типы сообщений, которыми пользователи обмениваются в Viber, а именно: текстовые сообщения, стикеры,

картинки и видео, URL-ссылки. Помимо этого, бот способен распознавать местоположение пользователя и получать контактную информацию о пользователе.

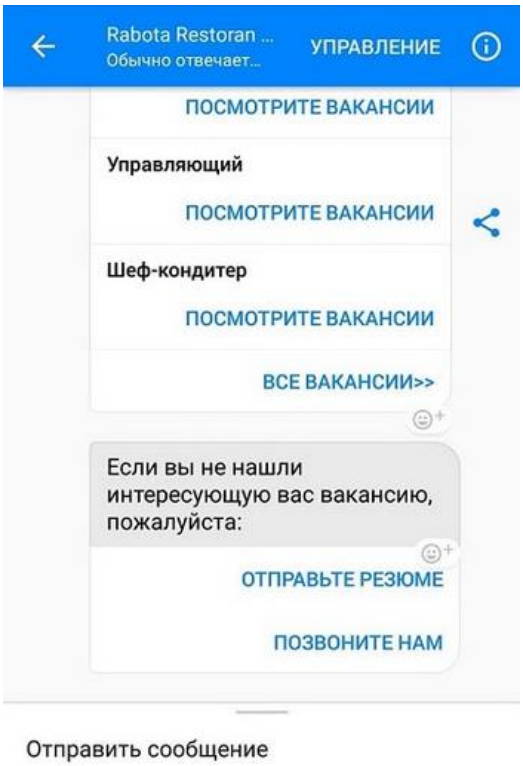

Рис. 6. Чат-бот кадрового агентства RABOTARESTORAN с кнопкой «Позвонить»

На данный момент, социальная сеть предоставляет средства разработки для создания своего чат-бота на языке программирования Python и платформе NodeJS [14].

На рисунке 7 представлен чат-бот банка «Альфа-банк», реализованный в приложении Viber.

ВКонтакте [15]

Создать чат-бота в социальной сети «ВКонтакте» возможно от имени сообщества либо пользователя (в случае если его страница является публичной). Бот может вести личную переписку с пользователем, а также участвовать в общей беседе (в том числе и с правами администратора или создателя беседы).

«ВКонтакте» предоставляет два подхода для реагирования на события в переписке – использование Callback API и подключение к Long Poll серверу. В случае с Callback API чат-бот может реагировать на любые события не только в переписке, но и в сообществе, от имени которого он общается. При использовании Long Poll сервера бот будет реагировать лишь на события, связанные с перепиской.

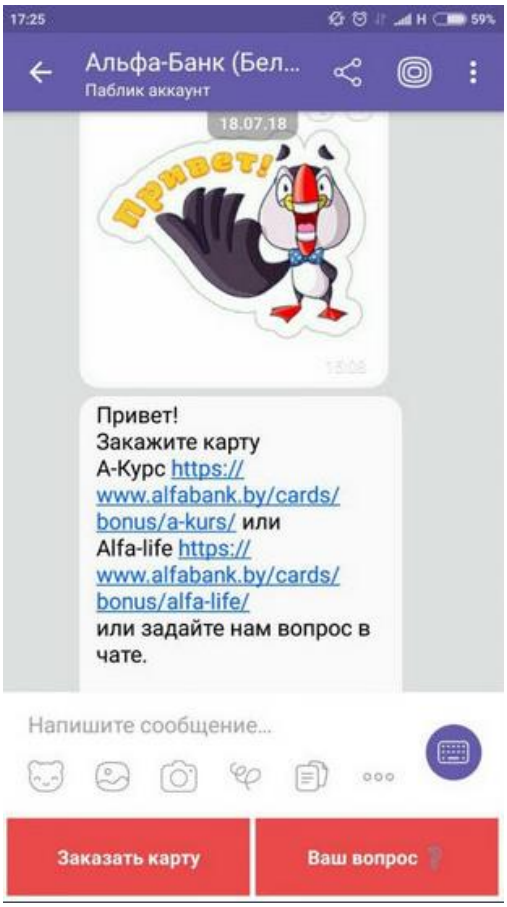

Рис. 7. Пример чат-бота от Альфа-банка с заказать карту и задать вопросы

Чат-бот способен отправлять и получать текстовые сообщения, фотографии, аудио- и видеофайлы, документы, аудиосообщения и ссылки [16]. Также в чат-бот можно добавить и кнопки для быстрых ответов [17].

Помимо этого, чат-боты, созданные в этой социальной сети, могут использовать возможности социального графа для создания подборок и рекомендаций для пользователя и являются кроссплатформенными, что

позволяет пользователям общаться с ботов как в полной, так и в мобильной версиях [18].

На рисунке 8 представлен чат-бот компании РБК, созданный в социальной сети «ВКонтакте».

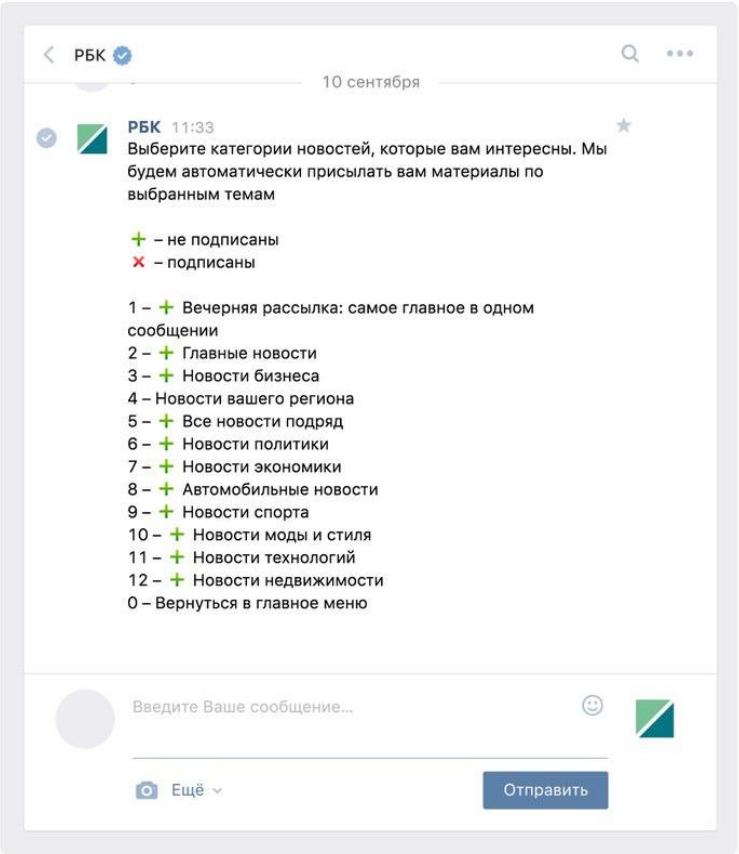

Рис. 8. Чат-бот РБК предлагает рассылки актуальных новостей

Выводы по первой главе

В результате анализа предметной области были рассмотрены существующие аналоги чат-ботов, реализующие задачу обучения иностранному языку, и были выявлены основные особенности подобных ботов. Помимо этого, были проанализированы возможности платформ, предоставляемые социальными сетями, на основе которых было решено использовать социальную сеть «ВКонтакте» для создания чат-бота.

2. ПРОЕКТИРОВАНИЕ

2.1. Функциональные требования к проектируемой системе

В ходе анализа требований к разрабатываемому чат-боту были сформулированы функциональные требования, приведенные ниже.

1. Чат-бот должен отображать инструкцию по взаимодействию с системой.

2. Чат-бот должен предоставлять тест для определения уровня английского языка пользователя.

3. Чат-бот должен предоставлять пользователю словарные слова в соответствие с выбранным пользователем разделом обучения.

4. Чат-бот должен предоставлять тестовые задания в соответствие с выбранным пользователем режимом обучения.

5. Чат-бот должен предоставлять задания на выбор пропущенных в предложении слов.

6. Чат-бот должен предоставлять пользователю статистику прохождения заданий.

2.2. Нефункциональные требования к проектируемой системе

В ходе анализа требований к разрабатываемому чат-боту были сформулированы нефункциональные требования, приведенные ниже.

1. Система чат-бота должна быть реализована на языке Python.

2. База данных чат-бота должна быть реализована при помощи СУБД PostgreSQL.

2.3. Диаграмма вариантов использования

На основе требований, предъявляемых к чат-боту, были разработаны варианты его использования, представленные на рисунке 9.

Основные актеры, взаимодействующие с системой

С системой взаимодействует пользователь, которым является любой человек, зарегистрированный в социальной сети «ВКонтакте» и использующий функционал чат-бота, например, студент.

Краткое описание вариантов использования

Пользователь может *получить инструкцию*, описывающую команды для взаимодействия с чат-ботом.

Пользователь может *пройти тест* на определение уровня английского языка. Для этого ему необходимо начать тест и ответить на вопросы теста, предложенного системой. Пользователь может в любой момент *завершить тест*. После прохождения теста, пользователь может заново начать тест.

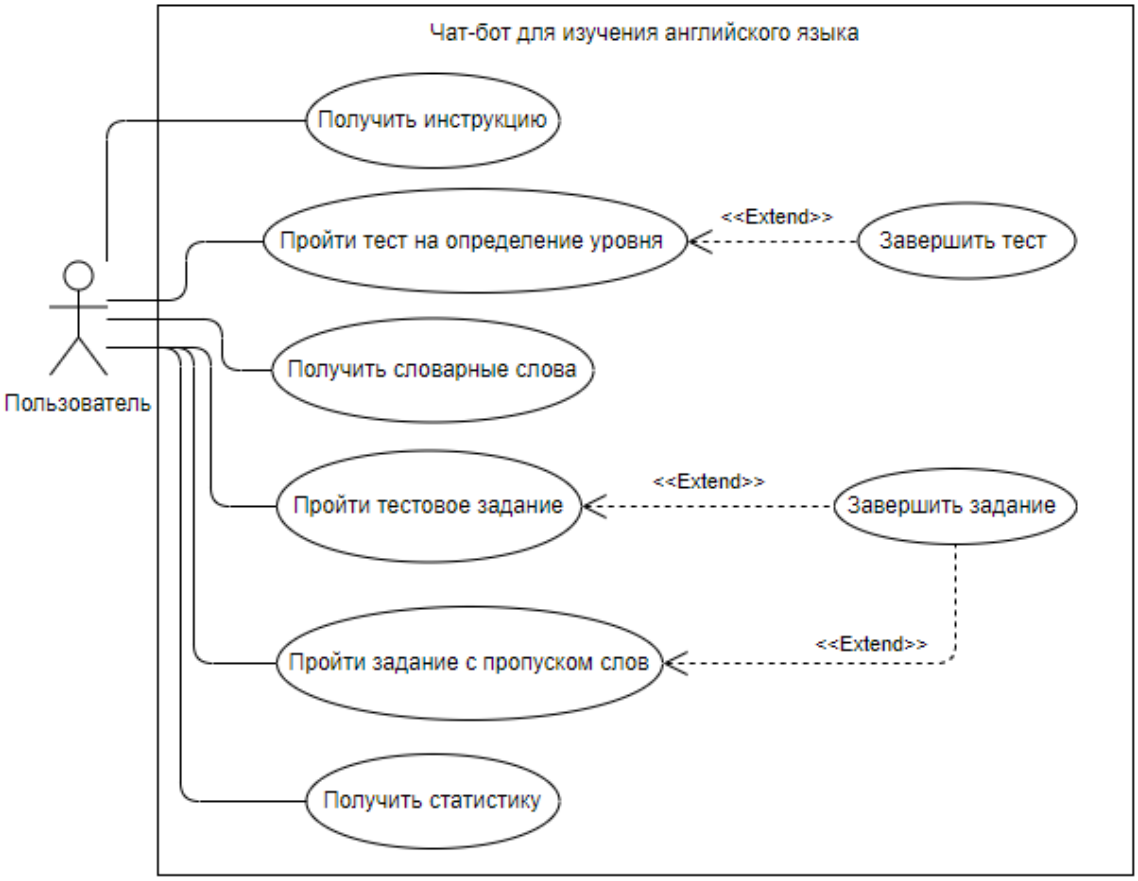

Рис. 9. Диаграмма вариантов использования чат-бота для изучения английского языка

Пользователь может *получить словарные слова*, представляющий собой список слов для изучения. Для этого пользователю необходимо выбрать свой уровень языка, а затем выбрать один из шести разделов, слова которого он хотел бы изучить.

Пользователь может *пройти тестовые задания*. Для этого ему необходимо выбрать свой уровень языка, а затем выбрать один из шести разделов, после чего указать тип тестового задания, которое он хотел бы выполнить. Пользователь может в любой момент *завершить задание* и получить результат, согласно отвеченным вопросам.

Пользователь может *пройти задания с пропуском слов*. Для этого ему необходимо выбрать свой уровень языка, а затем выбрать один из шести разделов, после чего выбрать задание на выбор пропущенных слов. Пользователь может в любой момент *завершить задание* и получить результат, согласно отвеченным вопросам.

Пользователь может *получить статистику* прохождения заданий. Для этого ему необходимо выбрать свой уровень языка, а затем выбрать один из шести интересующих его разделов.

Спецификация основных вариантов использования

Табл. 1. Спецификация вариантов использования при получении инструкции

***UseCase:* Получить инструкцию по взаимодействию**
ID: 1.
Аннотация: Получение указаний о способах взаимодействия с системой чат-бота
Главные актеры: Пользователь
Второстепенные актеры: нет
Предусловия: Главный актер инициировал диалог с чат-ботом
Основной поток: Прецедент начинается, когда главный актер вводит команду «Инструкция» или неправильную команду
Постусловия: Главный актер получил инструкцию
Альтернативные потоки: нет

Табл. 2. Спецификация вариантов использования при прохождении теста на определение уровня

***UseCase:* Пройти тест на определение уровня**
ID: 2.
Аннотация: Определение уровня английского языка, которым владеет главный актер
Главные актеры: Пользователь
Второстепенные актеры: нет
Предусловия: Пользователь инициировал диалог с ботом
Основной поток: 1. Прецедент начинается, когда Пользователь выбирает опцию «Пройти тест» 2. Пользователь выбирает опцию «начать тест» 3. Пользователь выбирает один из четырех вариантов ответа для каждого вопроса 3.1. Система определяет правильность ответа 3.2. Пользователь получает следующий вопрос 4. Пользователь завершает тест
Постусловия: Пользователь получает результаты теста
Альтернативные потоки: нет

Табл. 3. Спецификация вариантов использования при получении словарных слов

***UseCase:* Получить словарные слова**
ID: 3.
Аннотация: Получение списка словарных слов для изучения
Главные актеры: Пользователь
Второстепенные актеры: нет
Предусловия: Пользователь инициировал диалог с ботом
Основной поток: 1. Прецедент начинается, когда Пользователь выбирает опцию «Начать обучение» 2. Пользователь выбирает уровень языка и раздел для обучения

3. Пользователь выбирает опцию «Словарь» 4. Пользователь последовательно получает слова и переводы слов 5. Пользователь завершает обучение
Постусловия: Пользователь получил список слов для изучения
Альтернативные потоки: нет

Табл. 4. Спецификация вариантов использования при прохождении тестового задания

***UseCase:* Пройти тестовое задание**
ID: 4.
Аннотация: Прохождение тестового задания на знание выученных слов
Главные актеры: Пользователь
Второстепенные актеры: нет
Предусловия: Пользователь инициировал диалог с ботом
Основной поток: 1. Прецедент начинается, когда Пользователь выбирает опцию «Начать обучение» 2. Пользователь выбирает уровень языка и раздел для обучения 3. Пользователь выбирает опцию «Упражнение 1» либо «Упражнение 2» 4. Пользователь получает слово и вводит свой вариант ответа 4.1. Система проверяет правильность ответа 4.2. Пользователь получает следующее слово 5. Пользователь завершает задание
Постусловия: Пользователь получает результат тестового задания
Альтернативные потоки: нет

Табл. 5. Спецификация вариантов использования при прохождении задания с пропуском слов

***UseCase:* Пройти задание с пропуском слов**
ID: 5.
Аннотация: Прохождение задание на использование выученных слов
Главные актеры: Пользователь
Второстепенные актеры: нет
Предусловия: Пользователь инициировал диалог с ботом
Основной поток: 1. Прецедент начинается, когда Пользователь выбирает опцию «Начать обучение» 2. Пользователь выбирает уровень языка и раздел для обучения 3. Пользователь выбирает опцию «Упражнение 3» 4. Пользователь получает предложение с пропущенным словом и вводит свой вариант ответа 4.1. Система проверяет правильность ответа 4.2. Пользователь получает следующее слово 5. Пользователь завершает задание
Постусловия: Пользователь получает результат тестового задания
Альтернативные потоки: нет

Табл. 6. Спецификация вариантов использования при завершении теста

***UseCase:* Завершить тест**
ID: 6.
Аннотация: Завершение теста на определение уровня
Главные актеры: Пользователь
Второстепенные актеры: нет
Предусловия: Пользователь начал проходить тест
Основной поток: 1.Прецедент начинается, когда актер выбирает опцию «Завершить» или последний ответ в тесте 2. Пользователь возвращается в меню теста

Окончание табл. 6

Постусловия: Пользователь завершает тест
Альтернативные потоки: нет

Табл. 7. Спецификация вариантов использования при завершении задания

***UseCase:* Завершить задание**
ID: 7.
Аннотация: Завершение тестового задания или задания с пропуском слов
Главные актеры: Пользователь
Второстепенные актеры: нет
Предусловия: Пользователь начал проходить тестовое задание или задание с пропуском слов
Основной поток: 1.Прецедент начинается, когда актер выбирает опцию «Завершить» 2. Пользователь получает результат прохождения задания и возвращается в меню заданий
Постусловия: Пользователь завершает тестовое задание или задание с пропуском слов
Альтернативные потоки: нет

Табл. 8. Спецификация вариантов использования при получении статистики

***UseCase:* Получить статистику**
ID: 8.
Аннотация: Получение статистики прохождения заданий в рамках интересующего раздела
Главные актеры: Пользователь
Второстепенные актеры: нет
Предусловия: Пользователь инициировал диалог с ботом
Основной поток: 1. Прецедент начинается, когда главный актер вводит команду «Статистика»

Окончание табл. 8

2. Пользователь выбирает уровень языка и раздел для обучения
Постусловия: Пользователь получает статистику по разделу
Альтернативные потоки: нет

2.4. Общее описание архитектуры системы

Разрабатываемый чат-бот представляет собой диалоговое взаимодействие с пользователем, которое осуществляется посредством диалога в социальной сети «ВКонтакте». Пользователь участвует в беседе с чат-ботом, ведущим диалог от имени сообщества социальной сети. Чат-бот способен принимать команды пользователя, обрабатывать их и формировать ответ, в соответствии с введенной командой.

Система чат-бота представляет собой обработчика, который в зависимости от введенной пользователем команды, определяет, каким образом следует сформировать ответное сообщение, и отправляет это сообщение на сервер социальной сети.

После того, как пользователь вводит любое сообщение, система оправляет пользователю инструкцию, в которой указано, каким образом осуществляется управление чат-ботом. Пользователь может ввести определённую команду сообщением, либо сформировать сообщение с помощью кнопок клавиатуры чат-бота. Запрос с сообщением отправляется на сервер, где происходит его обработка. Сервер может обратиться к локальной базе данных для получения материалов для теста на определение уровня владения английским языком. Помимо этого, сервер может извлечь из базы данных необходимые пользователю учебные материалы.

2.5. Взаимодействие пользователя и чат-бота

На рисунке 10 представлена диаграмма последовательности, демонстрирующая процесс взаимодействия пользователя и чат-бота.

После того, как пользователь вводит сообщение, выполняется запрос к серверу социальной сети «ВКонтакте». Затем, сервер социальной сети отправляет запрос к системе чат-бота с содержанием пользовательского сообщения. Как только система получила запрос, происходит обработка запроса с целью получения из него сообщения пользователя. Полученное сообщение сопоставляется с набором команд, и распознается необходимая команда. В зависимости от результата формируется ответное сообщение. Затем, формируется запрос на сервер «ВКонтакте», содержащий ответное сообщение, после чего результат отображается пользователю в диалоге.

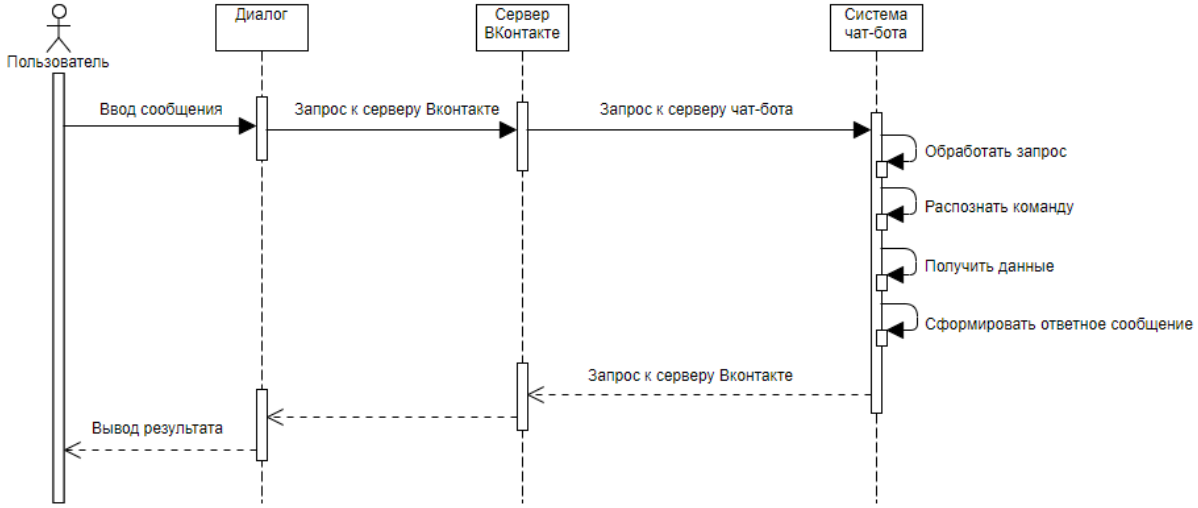

Рис. 10. Диаграмма последовательности обработки команды пользователя

2.6. Схема базы данных

На рисунке 11 представлена схема базы данных сервера. База данных включает в себя две взаимосвязанные таблицы vocabulary и gaps_exercise, таблицу данных для теста test, таблицу данных для пользователей user_accs, и таблицу данных для статистики по заданиям unit.

Таблица *vocabulary* используется для хранения набора словарных слов, соответствующих лексикону академического английского языка, и используется для изучения в словаре и тестовых заданиях. Данная таблица имеет следующие поля:

– *word_id* – уникальный идентификатор слова в таблице, представленный в формате целого числа;

– *word* – слово на английском, представленный в формате текста;

– *translat* – перевод слова на русский, представленный в формате текста;

– *unit* – номер раздела, которому соответствует слово; представлен в формате целого числа;

– *level* – значение уровня, которому соответствует слово; представлен в формате текста.

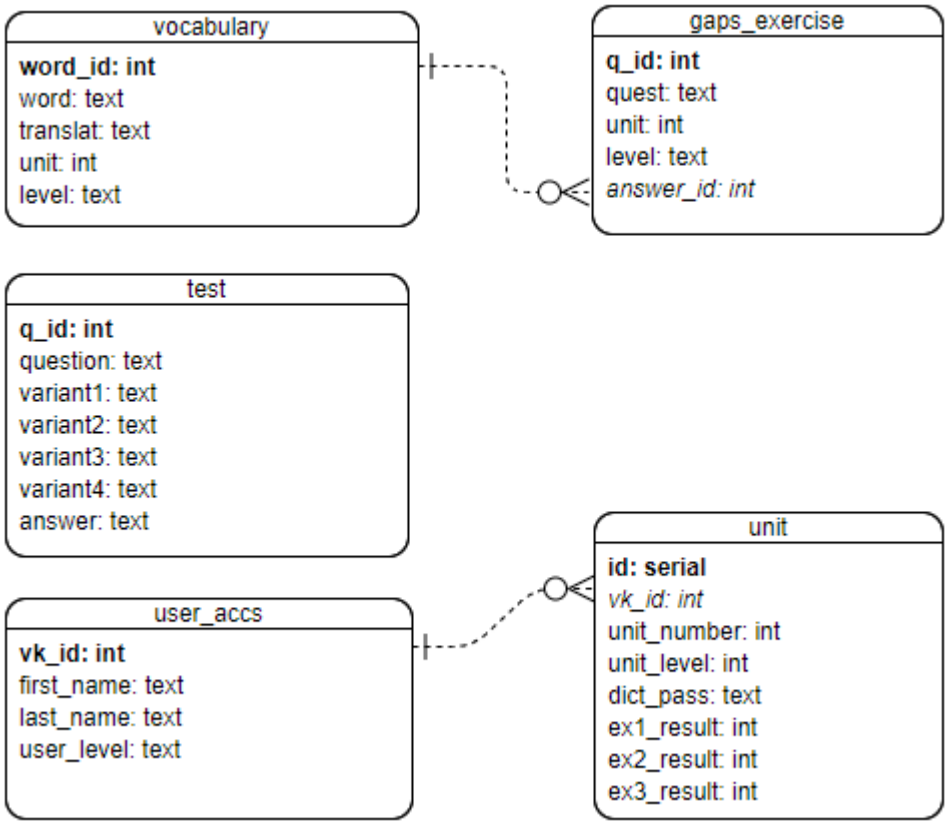

Рис. 11. Схема базы данных сервера

Таблица *gaps_exercise* используется для хранения данных упражнения на выбор пропущенных слов и имеет следующие поля:

– *q_id* – уникальный идентификатор предложения в таблице, представленный в формате целого числа;

– *quest* – предложение из задания, представленное в формате текста;

– *unit* – номер раздела, которому соответствует предложение; представлен в формате целого числа;

– *level* – значение уровня, которому соответствует предложение; представлен в формате текста;

– *answer_id* – уникальный идентификатор слова, соответствующий пропущенному слову в предложении; представлен в формате целого числа.

Таблица *test* используется для хранения данных теста на определение уровня владения английским языком и имеет следующие поля:

– *q_id* – уникальный идентификатор вопроса теста в таблице, представленный в формате целого числа;

– *question* – вопрос теста, представленный в формате текста;

– *variant1* – первый вариант ответа на вопрос теста, представленный в формате текста;

– *variant2* – второй вариант ответа на вопрос теста, представленный в формате текста;

– *variant3* – третий вариант ответа на вопрос теста, представленный в формате текста;

– *variant4* – четвертый вариант ответа на вопрос теста, представленный в формате текста;

– *answer* – правильный ответ на вопрос теста, представленный в формате текста.

Таблица *user_accs* используется для хранения данных о пользователе, инициировавшем диалог с чат-ботом:

– *vk_id* – уникальный идентификатор пользователя «ВКонтакте» в таблице, представленный в формате целого числа;

– *fisrt_name* – имя пользователя, представленное в формате текста;

– *last_name* – фамилия пользователя, представленная в формате текста;

– *user_level* – значение уровня пользователя, представленное в формате текста.

Таблица *unit* используется для хранения статистики пользователей по прохождению заданий и имеет следующие поля:

– *id* – уникальный идентификатор записи в таблице, представлена в формате serial (что соответствует целочисленному типу данных, который позволяет автоматически присваивать следующий порядковый номер);

– *vk_id* – идентификатор пользователя «ВКонтакте», представленный в формате целого числа;

– *unit_number* – номер раздела, которому соответствует упражнение; представлен в формате целого числа;

– *unit_level* – значение уровня, которому соответствует упражнение; представлен в формате текста;

– *dict_pass*– обозначение прохождения пользователем словаря соответствующего уровня; представлен в формате текста;

– *ex1_result* – результат пользователя в ходе прохождения упражнения 1 в чат-боте; представлен в формате целого числа;

– *ex2_result* – результат пользователя в ходе прохождения упражнения 1 в чат-боте; представлен в формате целого числа;

– *ex3_result* – результат пользователя в ходе прохождения упражнения 1 в чат-боте; представлен в формате целого числа.

2.7. Архитектура основного цикла системы чат-бота

На рисунке 12 представлена диаграмма деятельности чат-бота. Система разрабатываемого бота находится в ожидании запроса от сервера социальной сети «Вконтакте». Если пришедший от сервера запрос является новым непрочитанным сообщением, сообщением в диалоге с чат-ботом и сообщением в текстовом формате, то чат-бот начинает сопоставлять команду, распознанную из запроса с сервера социальной сети, с возможными командами, известными боту. В случае если команда соответствует какой-

либо команде из списка команд, то чат-бот начинает формировать сообщение. Сервер чат-бота запрашивает данные из базы данных, содержащих учебные материалы. После этого, полученные данные добавляются в текст ответного сообщения и отправляется пользователю. Если распознанная команда не соответствует ни одной из команд, чат-бот формирует сообщение с инструкцией, содержащей возможные команды чат-бота.

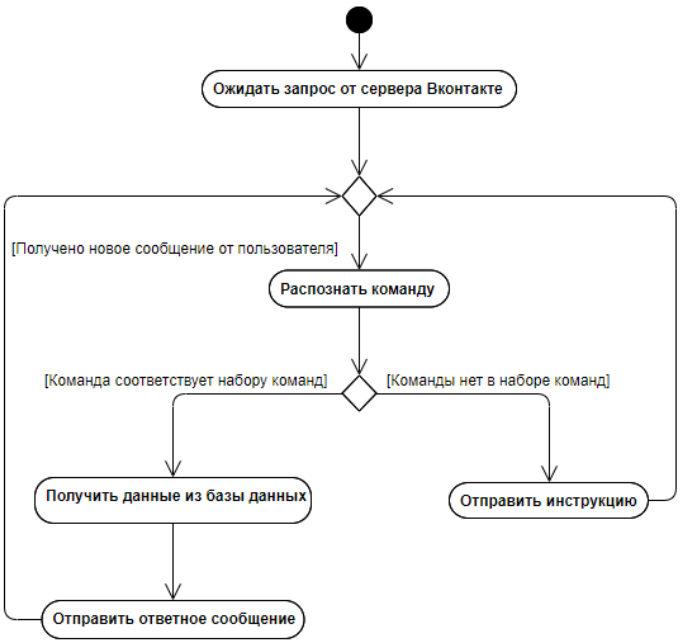

Рис. 12. Диаграмма деятельности чат-бота

2.8. Архитектура предоставления теста на определение уровня английского языка

На рисунке 13 представлена диаграмма деятельности чат-бота в рамках модуля «Тест на определение уровня». Система отправляет пользователю первый вопрос и ожидает ответного сообщения от пользователя. Если пришедшее сообщение содержит команду-вариант ответа на вопрос теста «A», «B», «C» или «D», система определяет, был ли дан правильный ответ на предыдущий вопрос, после чего определяет следующий вопрос и формирует ответное сообщение с ним. Если пришедшее сообщение является командой-вариантом, соответствующей ответу на последний вопрос теста, или содержит команду «Завершить», система обновляет результат

прохождения теста пользователем в базе данных и отправляет пользователю результат теста. Затем модуль «Тест на определение уровня» завершается.

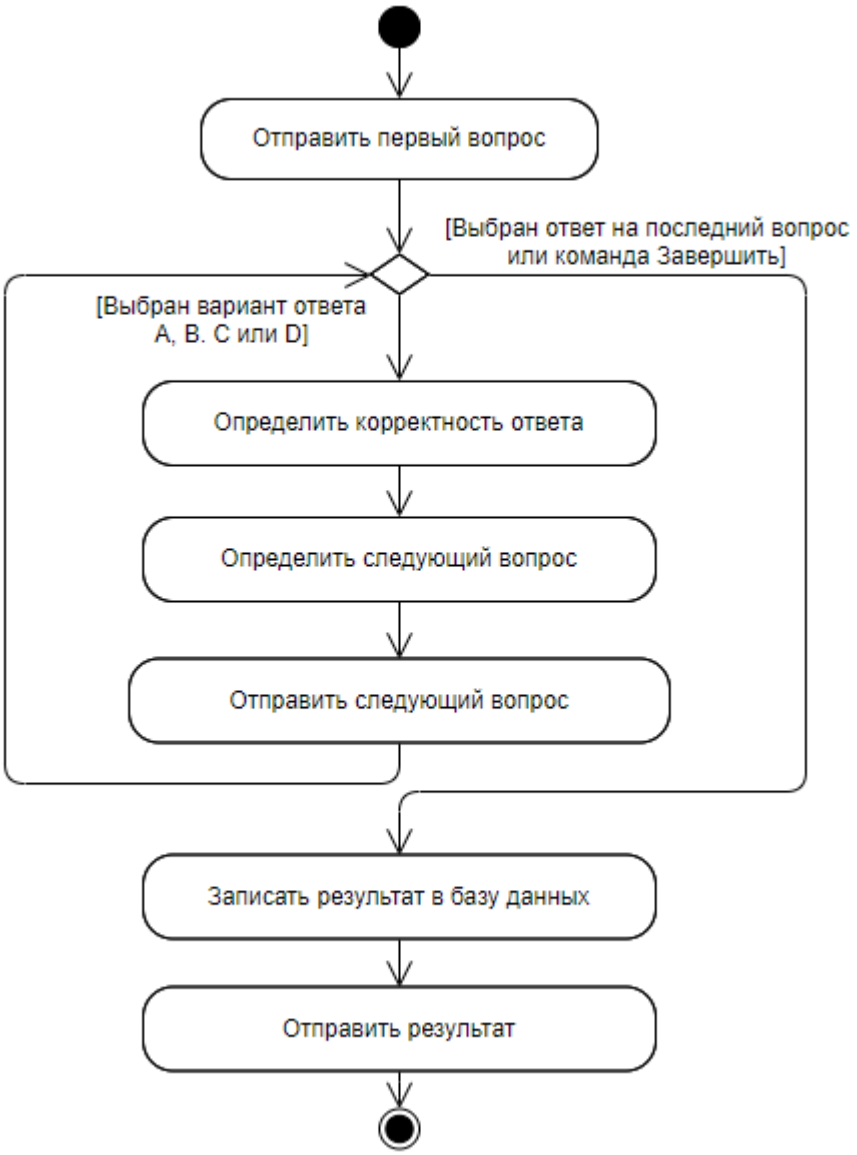

Рис. 13. Диаграмма деятельности чат-бота в рамка модуля «Тест на определение уровня»

2.8. Архитектура предоставления словарных слов

На рисунке 14 представлена диаграмма деятельности чат-бота в рамках модуля «Словарь». Система определяет словарь какого уровня и из какого раздела был выбран. Учитывая полученную информацию, из базы данных извлекаются соответствующие пары слов и переводов, которые формируются в список. Таким образом, система отправляет пользователю

первое слово и его перевод и ожидает ответного сообщения от пользователя. Если пришедшее сообщение содержит команду «Далее», система определяет и отправляет следующую пару. Если пришедшее сообщение содержит команду «Готово», относящуюся к последнему слову в списке система, система обновляет данные о прохождении словаря для текущего пользователя и завершает модуль «Словарь».

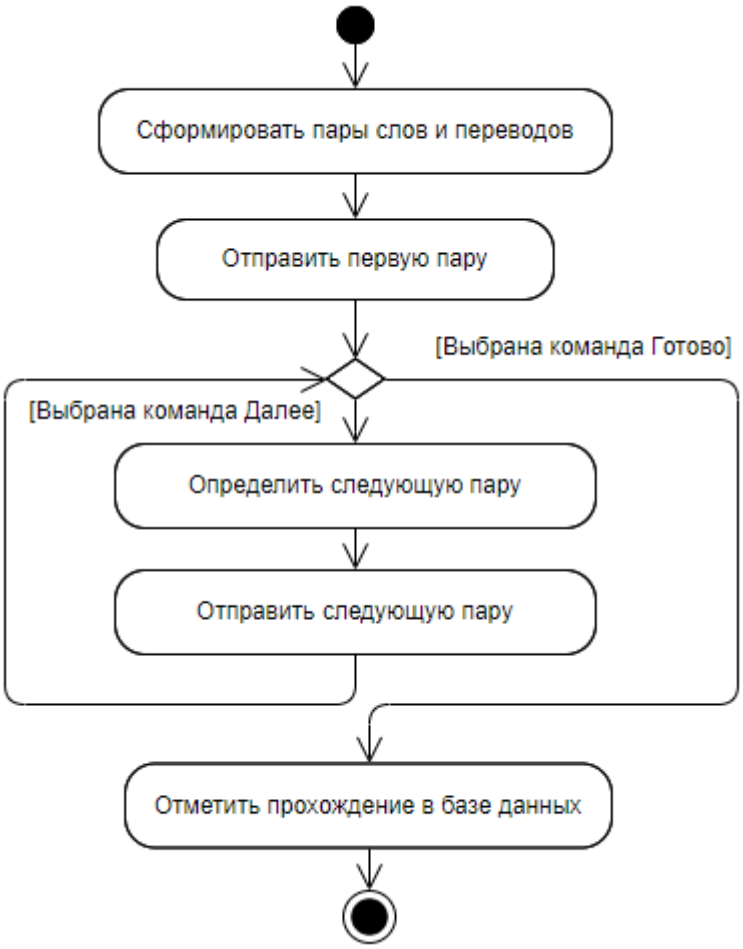

Рис. 14. Диаграмма деятельности чат-бота в рамка модуля «Словарь»

2.9. Архитектура тестового задания и задания на пропуск слов

На рисунке 15 представлена диаграмма деятельности чат-бота в рамках модулей «Тестовое задание» и «Выбор пропущенного слова». Система определяет для какой уровня и из какого раздела было выбрано тестовое задание или задание с пропуском слов. Учитывая полученную информацию, из базы данных извлекаются соответствующие вопросы и ответы к ним. Таким образом, формируется список заданий и ответов, на основе ко-

торых будет составлено первое задание, а потом и следующие. Система отправляет пользователю первый задание и ожидает ответного сообщения от пользователя. Если пришедшее сообщение содержит один из вариантов ответа на здание, система определяет, выбрал ли пользователь правильный ответ на предыдущее задание, а также определяет и формирует следующие задание. В случае если пользователь выбирает последнее задание или команду завершить, система обновляет данные по прохождению соответствующего задания для текущего пользователя в базе данных, а также формируется ответное сообщение с результатом пользователя. После чего система завершает модуль.

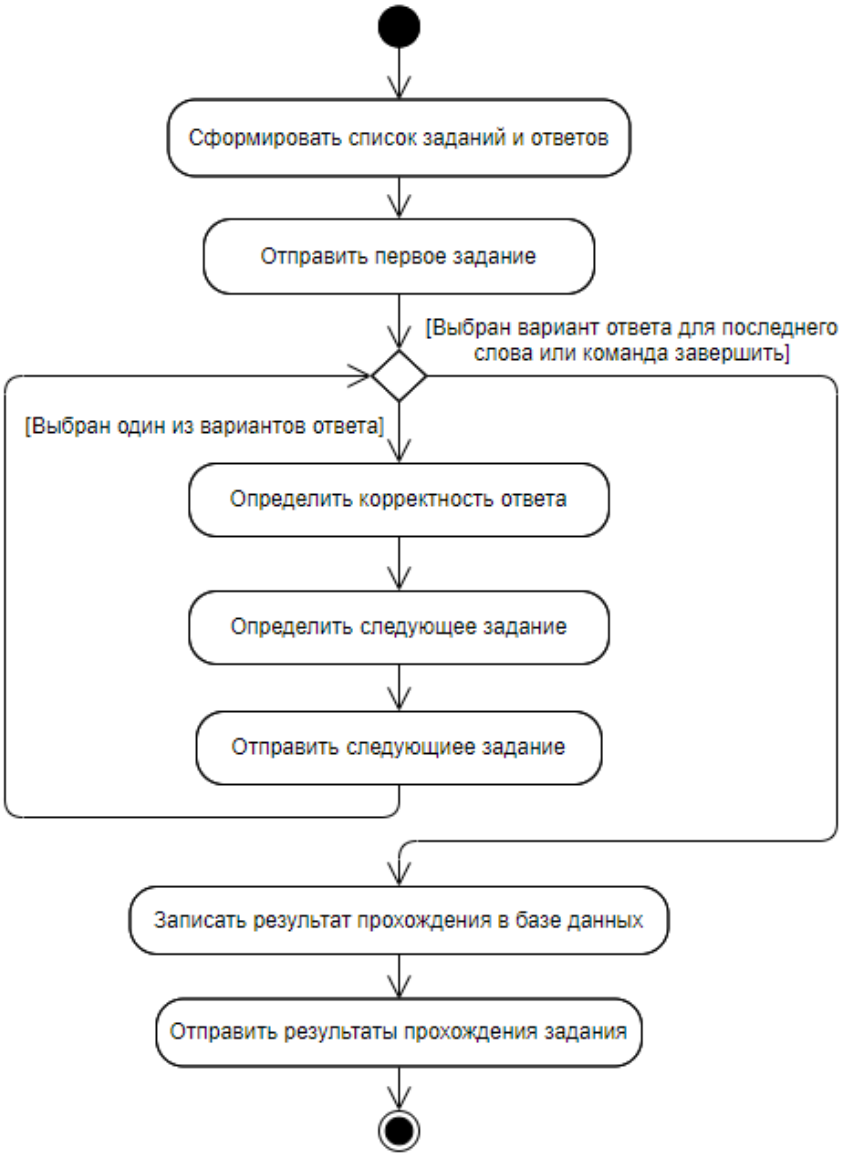

Рис. 15. Диаграмма деятельности чат-бота в рамках модулей «Тестовое задание» и «Выбор пропущенного слова»

2.10. Архитектура предоставления статистики

На рисунке 16 представлена диаграмма деятельности чат-бота в рамках модуля «Статистика». Система определяет для какой уровня и из какого раздела запрашивается статистика. Затем из базы данных извлекается статистка по определенному разделу и уровню, соответствующую пользователю, который запросил статистику. После чего система формирует сообщение с результатами статистики для пользователя и завершает модуль «Статистика».

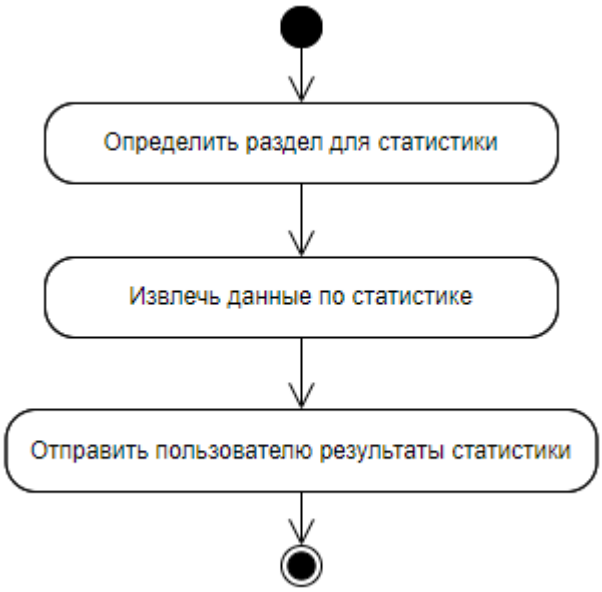

Рис. 16. Диаграмма деятельности чат-бота в рамка модуля «Статистика»

Выводы по второй главе

В процессе анализа требований были сформулированы основные функциональные и нефункциональные требования, предъявляемые к разрабатываемой системе. На основе этих требований были представлены диаграммы вариантов использования и последовательности обработки команд, диаграммы деятельности системы, а также спроектирована база данных.

3. РЕАЛИЗАЦИЯ

3.1. Средства реализации

В соответствии с требованиями к разрабатываемой системе, для реализации системы чат-бота использовался язык Python 3. Написание программного кода и его отладки осуществлялась в среде разработки PyCharm Community Edition 2018.3.5.

Для работы с запросами социальной сети «ВКонтакте» и формирования ответных запросов на сервер социальной сети использовался библиотека vk_api и ее модули VkLongPoll и VkEventType. Данная библиотека предназначена для написания скриптов для социальной сети и корректной работы с VK API. В библиотеке реализованы методы, необходимые для обработки и формирования запросов для работы с серверами социальной сети.

Для работы с файлами формата JSON был использован встроенная библиотека json. Данная библиотека предоставляет необходимые методы для сериализации строкового типа данных языка Python в json-объект и десериализации json-объекта в строковый тип данных.

База данных, содержащая учебные материалы для чат-бота, реализована с помощью СУБД PostgreSQL. Для работы с этой СУБД использовалась библиотека psychopg2.

Помимо этого, вспомогательные функции систем чат-бота, а также неизменяемые списки данных были вынесены в дополнительные модули add_functions.py, keyboards.py и payloads.py.

Порядок подключения вышеуказанных библиотек и модулей представлены на листинге 1.

Листинг 1. Подключение используемых библиотек и модулей

```
import vk_api
from vk_api.longpoll import VkLongPoll, VkEventType
from payloads import *
from add_functions import *
```

3.2. Начальная настройка чат-бота

Для того, чтобы приступить к разработке чат-бота необходима начальная настройка возможностей чат-бота в сообществе социальной сети «ВКонтакте», которая осуществляется в разделе «Сообщения» в управлении сообщества (как показано на рисунке 17).

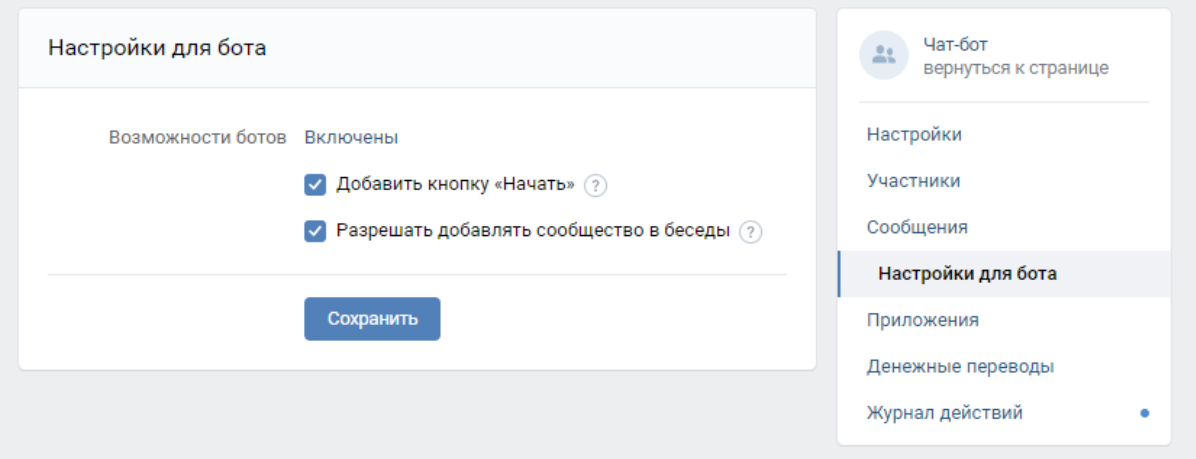

Рис. 17. Окно настройки чат-бота

3.3. Работа с сервисом API ВКонтаке

API ВКонтакте предоставлен разработчиками социальной сети для взаимодействия с базами данных и сервером «ВКонтакте».

Основным классом API, с помощью которого осуществляется взаимодействие с сервисами социальной сети, является vk_api.VkApi. С помощью данного метода происходит начальная аутентификация системы-чат-бота. Так как разработанный чат-бот участвует в диалоге от имени сообщества, аутентификация проходит по параметру token – секретный ключ доступа, состоящий из последовательности цифр, которую предоставляет социальная сеть при начальной настройке чат-бота.

На рисунке 18 представлен раздел настройки сообщества, где необходимо получить ключ доступа.

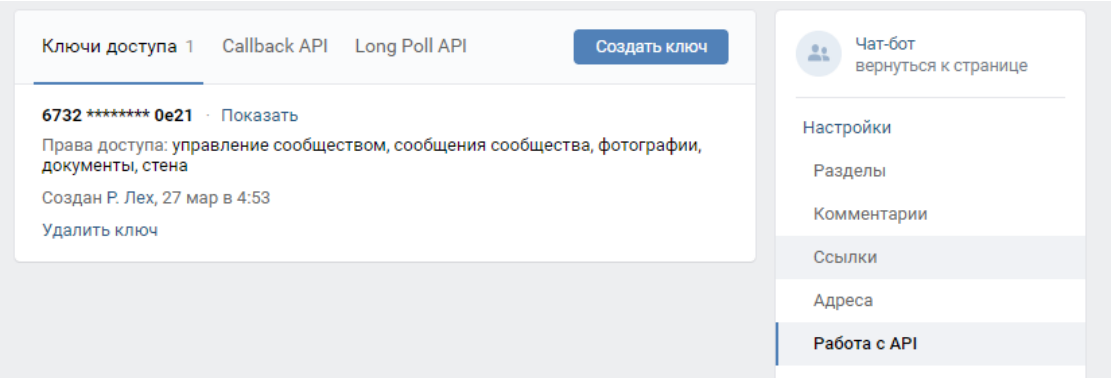

Рис. 18. Получение ключа доступа

После того как класс получил ключ доступа, создается сеанс взаимодействия чат-бота с сервером социальной сети, как представлено на листинге 2.

Листинг 2. Создание сеанса чат-бота

```
vk_session = vk_api.VkApi(token = '6732..0e21')
```

После аутентификации создается объект, обеспечивающий передачу данных между системой чат-бота. Данный объект создается с помощью класса vk_api.longpoll.VkLongPoll, принимающий в качестве параметра объект, отвечающий за сеанс взаимодействия чат-бота. Порядок создания объекта представлен на листинге 3.

Листинг 3. Создание объекта LongPoll

```
longpoll = VkLongPoll(vk_session)
```

Для удобства взаимодействия с методами API использовался метод get_api() объекта класса VkApi. Данный метод позволяет обращаться к методам API как к обычным классам языка программирования Python. Вызов метода представлен на листинге 4.

Листинг 4. Метод get_api()

```
vk = vk_session.get_api()
```

3.4. Вспомогательные функции системы чат-бота

Большая часть вспомогательных функций чат-бота были вынесены в дополнительные модули add_functions.py и keyboards.py. В первом модуле определены функции, определяющие работу чат-бота и его взаимодействие с базой данных. Второй модуль содержит функции, описывающие структуру клавиатуры чат-бота.

Создание клавиатуры чат-бота

Сервер «ВКонтакте» принимает клавиатуру в качестве json-объекта, преобразованного в строковый тип данных. Для этого был создана функция get_kybd (keyb), принимающая в качестве параметра json-объект keyb и возвращающая клавиатуру в строке. С помощью метода json.dumbs – объект сериализуется (переводится в последовательность битов) и с помощью функции str() – преобразуется в строковый тип данных. Порядок преобразования клавиатуры для чат-бота представлен на листинге 5.

Листинг 5. Создание клавиатуры

```
def get_kybd(keyb):
    return str(json.dumps(keyb, ensure_ascii=False))
```

На листинге 6 приведен пример создания клавиатуры из двух кнопок (подобная клавиатура используется в главном меню чат-бота). Функция keyboard2button(n1, n2) принимает в качестве параметров строку n1 и n2, текст которых будет нанесен на соответствующие кнопки клавиатуры. В качестве полей json-объекта указываются временной параметр клавиатуры, то есть будет ли она постоянно присутствовать на экране ли нет («one_time»), и количество кнопок («action»). Кроме того, указываются текст кнопки («type» и «label») и ее цвет («color»). В результате функция возвращает сформированный json-объект.

Аналогичным образом создаются клавиатуры для текущего тестового вопроса и для меню текущего раздела в функциях gen_test_keyboard(q_id), которая принимает на вход идентификатор текущего вопроса в тесте q_id, и gen_unit_keyboard(unit_num, lvl_name), кото-

рая принимает на вход значение текущего раздела unit_num и текущего уровня lvl_name. В обеих функция pld_set является списком полезных нагрузок для необходимых кнопок. Функции представлены на листингах 7 и 8.

Листинг 6. Пример создания клавиатуры с двумя кнопками

```
def keyboard2button(n1, n2):
    return {
    "one_time" : True,
    "buttons": [[
        {
            "action": {
                "type": "text",
                "label": n1
            },
            "color": "primary"
        },
        {
            "action": {
                "type": "text",
                "label": n2
            },
            "color": "primary"
        },
    ]]
    }
```

Листинг 7. Создание клавиатуры для теста

```
def gen_test_keyboard(q_id):
    pld_set = ["{\"button\": \"A" + str(q_id) + "\"}",
               "{\"button\": \"B" + str(q_id) + "\"}",
               "{\"button\": \"C" + str(q_id) + "\"}",
               "{\"button\": \"D" + str(q_id) + "\"}"]
    kb = get_keyboard(keyboard_test_buttons(pld_set))
    return kb
```

Листинг 8. Создание клавиатуры для раздела

```
def gen_unit_keyboard(unit_num, lvl_name):
    pld_set = ["{\"button\": \"dictU" + str(unit_num) + lvl_name + "\"}",
               "{\"button\": \"ex1U" + str(unit_num) + lvl_name + "\"}",
               "{\"button\": \"ex2U" + str(unit_num) + lvl_name + "\"}",
               "{\"button\": \"ex3U" + str(unit_num) + lvl_name + "\"}",
               "{\"button\": \"bU" + str(unit_num) + lvl_name + "\"}"]
    kb = get_keyboard(unit_keyboards(pld_set))
    return kb
```

Получение списка всех слов и их переводов, находящихся в базе данных

Для получения списков всех слов и их переводов из базы данных была реализована функция get_all_words(). Для начала работы с базой дан-

ных необходимо открыть соединение и курсор базы данных с помощью функций connect() и cursor(). Затем с помощью функции execute() формируется запрос базе данных, результат выполнения которого извлекается из курсора с помощью функции fetchall(), возвращающую все записи запроса. После обращения к базе данный соединение и курсор необходимо закрыть с помощью метода close(). Полученные данные с помощью цикла for добавляются в списки слов out_list1 и out_list2 для английских и русских слов соответственно. В результате функция возвращает два списка слов. Реализация функции представлена на листинге 9.

Листинг 9. Получения списка всех слов

```
def get_all_words():
    command_line1 = 'SELECT word FROM vocabular;'
    command_line2 = 'SELECT translat FROM vocabular;'

    conn = psycopg2.connect(dbname='bottest3', user='postgres',
                            password='1234', host='localhost')
    cursor = conn.cursor()
    cursor.execute(command_line1)
    out1 = cursor.fetchall()
    out_list1 = []
    for e in out1:
       out_list1.append(e[0])
    cursor.execute(command_line2)
    out2 = cursor.fetchall()
    out_list2 = []
    for e in out2:
        out_list2.append(e[0])
    cursor.close()
    conn.close()
    return out_list1, out_list2
```

Получение вопроса и ответа на вопрос теста из базы данных

Для получения вопроса и ответа теста на определения уровня английского языка из базы данных была реализована функции get_test_q (q_id) и get_test_answ(a_id), принимающие в качестве параметра идентификатор текущего вопроса теста. Для начала работы с базой данных также необходимо открыть соединение и курсор базы данных. Затем сформированный запрос к базе данных извлекается из курсора с помощью функции fetchone(), возвращающую одну запись запроса. После обращения к базе данный соединение и курсор необходимо закрыть с помощью метода

close(). Функция возвращает результат запроса в строковом типе данных. Реализация функций представлена на листингах 10 и 11.

Листинг 10. Получение вопроса теста

```
def get_test_q(q_id):
    conn = psycopg2.connect(dbname='bottest3', user='postgres',
                                password='1234', host='localhost')
    cursor = conn.cursor()
    cursor.execute('SELECT question, variant1, variant2, variant3, variant4
FROM test WHERE q_id = %s;', (q_id,))
    out = cursor.fetchone()
    out_q = str(q_id)+'. '+out[0]+'\n'+out[1]+'\n'+out[2]+'\n'+
            out[3]+'\n'+out[4]
    cursor.close()
    conn.close()
    return out_q
```

Листинг 11. Получение ответа на вопрос теста

```
def get_test_answ(a_id):
    conn = psycopg2.connect(dbname='bottest3', user='postgres',
                                password='1234', host='localhost')
    cursor = conn.cursor()
    cursor.execute('SELECT answer FROM test WHERE q_id = %s;', (a_id,))
    out = cursor.fetchone()
    out_a = out[0]
    cursor.close()
    conn.close()
    return out_a
```

Получение списка слов и ответов для словаря из базы данных и тестовых заданий

Для получения списка слов была реализована функция get_vocab(unit, lvl, exer_name), принимающая в качестве параметров порядковый номер текущего раздела unit, выбранный пользователем уровень английского языка lvl и выбранный пользователем тип упражнения exer_name. Здесь также необходимо открыть соединение и курсор. Затем в зависимости от типа упражнения, уровня и раздела формируется запрос command_line к соответствующей таблице базы данных. После обращения к базе данный соединение и курсор необходимо закрыть с помощью метода close(). В результате функция возвращает список с извлеченными данными. Реализация функции представлена на листинге 12.

Листинг 12. Фрагмент функции получения списка словарных слов.

```
def get_vocab(unit, lvl, exer_name):
    if exer_name.lower() == 'словарь' or exer_name.lower() == 'упражнение
1':
        command_line = 'SELECT word, translat ' \
                        'FROM vocabular ' \
                        'WHERE unit = %s AND lvl = %s;'
    elif exer_name.lower() == 'упражнение 2':
        command_line = 'SELECT translat, word ' \
                       'FROM vocabular ' \
                       'WHERE unit = %s AND lvl = %s;'
    ...
    return out
```

Получение списка предложений и ответов для упражнения на выбор пропущенных слов

Для получения списка предложений и ответов для указанного задания была реализована функция get_exer(unit_nb, level), принимающая в качестве параметров номер необходимого раздела unit_nb и значение соответствующего уровня level. Здесь также необходимо открыть соединение и курсор. Затем в зависимости от типа упражнения, уровня и раздела формируется запрос command_line к соответствующей таблице базы данных. После обращения к базе данный соединение и курсор необходимо закрыть с помощью метода close(). Функция возвращает результат запроса списком. Реализация функции представлена на листинге 13.

Листинг 13. Фрагмент функции получения задания для упражнения

```
def get_exer(unit_nb, level):
    command_line = 'SELECT q_id, quest, word ' \
                   'FROM vocabular, proba_gaps_exercise as pg ' \
                   'WHERE answer_id=word_id AND pg.unit = %s ' \
                       'AND pg.lvl = %s ORDER BY q_id;'
    ...
    return out
```

Формирование вариантов ответа на упражнения

Функция form_random (c_word, c_pld, w_word, w_pld) позволяет представить варианты ответа на вопрос в упражнении в случайном порядке. Данная функция принимает на вход значение и полезную нагрузку для кнопки с правильным ответом (c_word и c_pld, соответственно), а также значение и полезную нагрузку для кнопки с неправильным (w_word,

w_pld, соответственно). Далее с помощью функции randint() программного модуля random случайно выбирается число, которое определяет порядок заполнения списка значений и полезных нагрузок кнопок. Таким образом, если случайное значение равно 0, то в начале в список добавляются значение и полезная нагрузка кнопки с правильным ответом, а если 1 – то эти значение и полезная нагрузка добавляются в конце. В результате функция возвращает сформированный список значений и полезных нагрузок. В листинге 14 представлен программный код данной функции.

Листинг 14. Формирование случайного набора.

```
def form_rand(c_word, c_pld, w_word, w_pld):
    rnd = random.randint(0,1)
    answer_set =[]
    if rnd == 0:
        answer_set.append(c_word)
        answer_set.append(c_pld)
        answer_set.append(w_word)
        answer_set.append(w_pld)
    elif rnd == 1:
        answer_set.append(w_word)
        answer_set.append(w_pld)
        answer_set.append(c_word)
        answer_set.append(c_pld)
    return answer_set
```

Определение всех пользователей, записанных в базе данных

Для того, чтобы получить список всех пользователей, информация о которых хранится в базе данных, была реализована функция get_db_users(). Здесь также необходимо открыть соединение и курсор. Затем с помощью функции execute() формируется запрос базе данных, результат выполнения которого извлекается из курсора с помощью функции fetchall(). После обращения к базе данный соединение и курсор необходимо закрыть с помощью метода close(). Функция возвращает список с результатом запроса. Реализация данной функции приведена на листинге 15.

Листинг 15. Получения списка всех пользователей

```
def get_db_users():
    ...
    cursor.execute('SELECT vk_id FROM user_accs;')
    ...
    return out
```

Добавление нового пользователя в базу данных

Для того, чтобы добавить информацию о новом пользователе была реализована функция add_new_user_to_db(user_vk_id, name, surname). Данная функция принимает в качестве входных параметров целочисленный идентификатор пользователя в социальной сети vk_id, его имя name и фамилию surname в строковом формате. Порядок открытия и закрытия соединения и курсора базы данных, а также исполнения запроса аналогичен порядку в ранее описанных функциях. Запросы к базе данных user_command_line и unit_command_line представляют собой форматированную строку, в которую впоследствии будут подставляться необходимые значения. Для того, чтобы изменения в базе данных вступили в силу, необходимо вызвать функцию commit(), после чего результаты исполненного запроса будут сохранены в базе данных. В результате данной функции информация о пользователе добавляется в таблицу пользователей и в таблицу со статистикой по разделам. Реализация функции представлена на листинге 16.

Листинг 16. Фрагмент функции добавления информации о пользователе в база данных

```
def add_new_user_to_db(user_vk_id, name, surname):
    user_accs_command_line = 'INSERT INTO user_accs(vk_id, first_name, ' \
                             'last_name, user_level) ' \
                             'VALUES (%s, %s, %s, %s);'
    conn = psycopg2.connect(dbname='bottest3', user='postgres',
                            password='1234', host='localhost')
    cursor = conn.cursor()
    cursor.execute(user_accs_command_line, (user_vk_id, name, surname,
'B1'))
    conn.commit()
    for lvl_nb in range(2):
        for unit_nb in range(6):
            unit_command_line = 'INSERT INTO unit(vk_id, unit_number, ' \
                                'unit_level, dict_pass, ex1_result, ' \
                                'ex2_result, ex3_result) ' \
                                'VALUES (%s, %s, %s, %s, 0, 0, 0);'
            lvl = 'B' + str(lvl_nb + 1)
            cursor.execute(unit_command_line, (user_vk_id, unit_nb+1, lvl,
'No'))
            conn.commit()
    cursor.close()
    conn.close()
```

Получение статистики по пользователю

Для получения статистики по пользователю была реализована функция get_unit_stats(unit_nb, lvl, vk_id), принимающая в качестве параметров номер необходимого раздела unit_nb, значение соответствующего уровня level и идентификатор пользователя в социальной сети. Здесь также необходимо открыть соединение и курсор. Затем в зависимости уровня и раздела формируется запрос command_line к соответствующей таблице базы данных. После обращения к базе данный соединение и курсор необходимо закрыть с помощью метода close(). Функция возвращает результат виде сформированной строки с результатами. Реализация функции представлена на листинге 17.

Листинг 17. Фрагмент функции получения статистики по пользователю

```
def get_unit_stats(unit_nb, lvl, vk_id):
    command_line = 'SELECT * FROM unit WHERE vk_id = %s ' \
                        'and unit_number = %s and unit_level=%s;'
    ...
    out_str = 'Словарь пройден:' + dict + '\n' + 'Упр. 1: ' +
                str(out[3]) + '\n' + 'Упр. 2: ' + str(out[4]) + '\n'
                + ' Упр. 3: ' + str(out[5])
    ...
    return out_str
```

Обновление информации о результатах пользователя

Для обновления результатов пользователя по мере прохождения теста, изучения словарных слов и выполнения тестовых заданий и задний с пропущенными словами были разработаны функции user_acc_update(user_lvl, vk_id), которая принимает на входе значение уровня пользователя user_lvl после прохождения теста и идентификатор пользователя vk_id, и unit_stat_update(unit, lvl, vk_id, exer, result), которая принимает на вход значение уровня lvl и раздела unit, а так же вид упражнения exer и результат упражнения result, выполненного пользователем с идентификатором vk_id. Как и в функции add_new_user_to_db() необходимо открыть соединение и курсор для базы данных. Затем в зависимости от типа упражнения, уровня и раздела сформировать запрос command_line к соответствующей таблице базы данных. После сохранения изменений в ба-

зе данных, произведенных с помощью функции commit(), соединение и курсор необходимо закрыть. Реализация функции добавления результатов теста на определение уровня и функции добавления результата по мере прохождения упражнений в рамках раздела представлены на листингах 18 и 19, соответственно.

Листинг 18. Фрагмент функции обновления уровня

```
def user_acc_update(user_lvl, vk_id):
    ...
    cursor.execute('UPDATE user_accs SET user_level = %s where vk_id =
%s;', (user_lvl, vk_id))
    conn.commit()
    ...
```

Листинг 19. Фрагмент функции обновление результатов пользователя

```
def unit_stat_update(unit, lvl, vk_id, exer, result):
    attribute = ''
    if exer == 'словарь':
        attribute = 'dict_pass'
    elif exer == 'ex1':
        attribute = 'ex1_result'
    elif exer == 'ex2':
        attribute = 'ex2_result'
    elif exer == 'ex3':
        attribute = 'ex3_result'
    command_line = 'UPDATE unit SET ' + attribute + '=%s WHERE vk_id = %s
and unit_number = %s and unit_level=%s;'
    ...
```

Получение полезной нагрузки кнопки

Для определения того, какая именно кнопка была выбрана пользователем, необходимо получить ее полезную нагрузку. Для этого была реализована функция get_payload. С помощью функции method вызывается метод getConversations, содержащий информацию о всех беседах чат-бота. После этого из списка, содержащего информацию о полученном сообщении, извлекается значение полезной нагрузки кнопки из последнего полученного сообщения. Реализация функции приведена на листинге 20.

Листинг 20. Получение полезной нагрузки

```
def get_payload():
    messages = vk_session.method("messages.getConversations", {"offset": 0,
"count": 20, "filter": "unread"})
    payload = messages["items"][0]["last_message"]["payload"]
    return payload
```

Получение имени и фамилии пользователя

Для определения имени и фамилии пользователя, взаимодействующего с чат-ботом в текущий момент, была реализована функция get_user_name(vk_id), принимающая в качестве параметра идентификатор пользователя. С помощью функции method вызывается метод users.get, который возвращает расширенную информацию о пользователях. Далее из списка пользователей выбирается тот, чей идентификатор соответствует vk_id и считываются данные, соответствующие полям fisrt_name и last_name. В результате функция возвращает имя и фамилию пользователя. Реализация функции приведена на листинге 21.

Листинг 21. Получение имени и фамилии пользователя

```
def get_user_name(vk_id):
    user_info = vk_session.method("users.get", {"user_ids": vk_id})
    first_name = user_info[0]["first_name"]
    last_name = user_info[0]["last_name"]
    return first_name, last_name
```

3.5. Основной цикл системы чат-бота

Основной цикл представляет собой цикл for, просматривающий каждое новое событие event в списке, возвращаемом методом longpoll.listen() объекта longpoll. Если тип события event.type является новым сообщением, которое пользователь прислал в диалоге с чат-ботом, чат-бот начинает искать команду в списке возможных команд, иначе выписывает инструкцию для чат-бота. Чтобы определить новое сообщение для чат-бота проверяется два условия: тип события должен соответствовать типу MESSAGE_NEW в списке событий VkEventType, и событие должно иметь поле event.to_me. Текст сообщения извлекается из поля event.text. Если текст сообщения совпадает с одной из команд, система начинает формировать сообщение с помощью метода messages.send, где в качестве параметра передается идентификатор пользователя user_id, случайный идентификатор random_id, текст ответного сообщения message и клавиатура бота keyboard.

Стоит отметить, что новый пользователь должен инициировать диалог с чат-ботом с команды «Начать». При выборе этой команды, пользова-

тель будет определен как новый с помощью функции get_db_users(), затем информация о нем будет добавлена в базу данных с помощью команды add_new_user_to db() и в список all_users_set, который хранит текущие настройки (в виде списка new_user) определённого пользователя. Список new_user включает идентификатор пользователя, значения текущего раздела и уровня, в которых будет работать пользователь, и настройки для модулей, такие как: отметка последнего вопроса, счет пользователя, порядковый номер текущего задания, номер предыдущего задания и список заданий и ответов.

Фрагмент реализации основного цикла и последовательности действий при команде «Начать» представлен на листинге 22.

Листинг 22. Фрагмент кода основного цикла

```
for event in longpoll.listen():
    if event.type == VkEventType.MESSAGE_NEW and event.to_me and
event.text:

        if event.text.lower() == 'начать' or \
               event.text.lower() == 'назад в главное меню' \
                or event.text.lower() == 'start':
            name = ''
            surname = ''
            name, surname = get_user_name(event.user_id)
            db_users = get_db_users()
            user_ids = []
            for user in db_users:
                user_ids.append(user[0])
            if not (event.user_id in user_ids):
                add_new_user_to_db(event.user_id, name, surname)
                all_users_set.append(new_user)

            kb = get_keyboard(keyboard2button("Пройти тест", "Начать обуче-
ние"))
            mssg = 'Приветствую, ' + name + ' ' + surname + ...'
            vk.messages.send(user_id=event.user_id, ran-
dom_id=event.random_id, message=mssg, keyboard=kb)
```

3.6. Модуль «Тест на определение английского языка»

Данный модуль состоит из трех условных операторов ветвления, описывающих работу модуля при начале выполнения теста в течение прохождения теста и при завершении теста через кнопку. Модуль начинается после того, как чат бот распознает команду пользователя «Начать тест». При этом определяется какой пользователь ввел команду и текущие

настройки пользователя (счет пользователя e[4], порядковый номер задания e[5] и отметка последнего вопроса e[3]) в списке all_users_set обнуляются. Затем пользователю отправляется сообщение с первым вопросом (код представлен на листинге 23).

Листинг 23. Начало модуля «Тест на определение уровня английского языка»

```
if event.text.lower() == 'начать тест':
    for e in all_users_set:
        if e[0] == event.user_id:
            e[4] = 0
            e[5] = 1
            e[3] = 0
            mssg = get_test_q(e[5])
            kb = gen_test_keyboard(e[5])
            vk.messages.send(user_id=event.user_id, ran-
dom_id=event.random_id, message=mssg, keyboard=kb)
```

Если пользователь выбирает одну из команд-вариантов ответа «A», «B», «C» или «D», система определяет какой пользователь ввел ответ и обновляет текущие настройки модуля в all_users_set и test_settings. Затем проверяется корректность введенного ответа, то есть система сверяет полученный от пользователя результат event.text с необходимым ответом на соответствующий вопрос, полученный из функции get_test_answ(). В случае если был дан ответ на последний вопрос (порядковый номер равен 40 и отметка последнего вопроса была изменена на 1), пользователь получает результаты теста в ответном сообщении, а его результат в базе данных обновляется с помощью функции user_acc_update(). Иначе, исходя из текущих настроек пользователя, система определяет какой следующий вопрос необходимо послать в сформированном сообщении. Фрагмент реализации представлен на листинге 24.

Если пользователь выбрал команду «Завершить», его возвращают в меню теста, как описано на листинге 25.

Листинг 24. Обработки сообщения при выборе ответа в тесте

```
if event.text.lower() in test_vars:
    pld = get_payload()
    test_settings = []
    for e in all_users_set:
        if e[0] == event.user_id:
            curr_test_id = e[5]
            if curr_test_id < 10:
                e[5] = int(pld[-3])
            elif curr_test_id >=10:
                e[5] = int(pld[-4:-2])
            if event.text == get_test_answ(e[5]):
                e[4] += 1
            test_settings.append(e[3])
            test_settings.append(e[4])
            test_settings.append(e[5])

    if test_settings[0] == 1 and test_settings[2] == 40:
        kb = get_keyboard(keyboard2button("Начать тест",
                                          "Назад в Главное меню"))
        if test_settings[1] >= 0 and test_settings[1] < 9:
            user_acc_update('A1', event.user_id)
            mssg = 'Ваш результат: Beginner(начальный уровень) - A1.\n' \
                   'Этот уровень не соответствует уровням B1 и B2-C1 \n' \
                   'Вы можете попробовать пройти тест снова или ' \
                       'вернуться в главное меню'
            vk.messages.send(user_id=event.user_id, ran-
dom_id=event.random_id, message=mssg, keyboard=kb)
        ...
    elif test_settings[2] <= 39:
        for e in all_users_set:
            if e[0] == event.user_id:
                if test_settings[2] == 39:
                    e[3] = 1
                e[5] += 1
                test_settings = []
                test_settings.append(e[3])
                test_settings.append(e[4])
                test_settings.append(e[5])
        mssg = get_test_q(test_settings[2])
        kb = gen_test_keyboard(test_settings[2])
        vk.messages.send(user_id=event.user_id, random_id=event.random_id,
message=mssg, keyboard=kb)
```

Листинг 25. Завершение модуля

```
if event.text.lower() == 'завершить':
    kb = get_keyboard(keyboard2button("Начать тест", "Назад в Главное ме-
ню"))
    mssg = 'Вы можете снова попробовать пройти тест. \n' \
           'Вам будет предложено 40 вопросов, после того как вы ответите' \
           ' на все из них Вы узнаете свой уровень английского языка на ' \
           'данный момент. \n \n Нажмите НАЧАТЬ, чтобы пройти тестирование,
' \
           'или НАЗАД, чтобы вернуться в меню.'
    vk.messages.send(user_id=event.user_id, random_id=event.random_id, mes-
sage=mssg, keyboard=kb)
```

3.7. Модуль «Словарь»

Данный модуль состоит из трех условных операторов ветвления, описывающих работу модуля при начале, в течение прохождения словаря и при завершении. Модуль начинается после того, как чат-бот распознает команду пользователя «Словарь». При этом определяется какой пользователь ввел команду и для какого уровня и раздела, при этом текущие настройки пользователя (номер раздела e[1], значение уровня e[2] и список словарных слов e[7]) в списке all_users_set обновляются. Затем пользователю отправляется сообщение с первой парой слов (код представлен на листинге 26).

Листинг 26. Начало модуля «Словарь»

```
if event.text.lower() == 'словарь':
    pld = get_payload()
    for e in all_users_set:
        if e[0] == event.user_id:
            unit = pld.find('U')
            level = pld.find('B')
            e[1] = int(pld[unit + 1])
            e[2] = "B" + pld[level + 1]
            e[7] = get_vocab(e[1], e[2], event.text)
            mssg = e[7][0][0] + '\n' + e[7][0][1]
            button_pld = "{\"button\": \"dictU" + str(e[1]) + e[2] + "W" +
str(0) + "\"}"
            kb = get_keyboard(dict_button(button_pld, "Далее"))
            vk.messages.send(user_id=event.user_id, ran-
dom_id=event.random_id, message=mssg, keyboard=kb)
```

Если пользователь выбирает команду «Далее», система определяет какой пользователь ее ввел и для какого словаря, затем текущие настройки модуля в all_users_set и dict_settings обновляются. В случае если пользователь прошел предпоследнее слово из списка слов, то модуль переходит к завершению, иначе, исходя из текущих настроек пользователя, система определяет какую следующую пару слов необходимо послать в сформированном сообщении. Фрагмент реализации представлен на листинге 27.

Если пользователь выбрал команду «Готово», результат прохождения словаря записывается в базу данных с помощью функции unit_stat_update(), и пользователя возвращают в меню соответствующего раздела, как описано на листинге 28.

Листинг 27. Работа модуля по мере прохождения словаря

```
if event.text.lower() == 'далее':
    pld = get_payload()
    dict_settings = []
    for e in all_users_set:
        if e[0] == event.user_id:
            unit = pld.find('U')
            level = pld.find('B')
            word = pld.find('W')
            e[1] = int(pld[unit + 1])
            e[2] = "B" + pld[level + 1]
            e[5] = int(pld[word+1])+1
            dict_settings.append(e[1])
            dict_settings.append(e[2])
            dict_settings.append(e[5])
            dict_settings.append(e[7])

    if dict_settings[2] == len(dict_settings[3])-1:
        button_pld = "{\"button\": \"bU" + str(dict_settings[0]) +
dict_settings[1] + "\"}"
        kb = get_keyboard(dict_button(button_pld, "Готово"))
    else:
        button_pld = "{\"button\": \"dictU" + str(dict_settings[0]) +
dict_settings[1] + "W" + \
                     str(dict_settings[2]) + "\"}"
        kb = get_keyboard(dict_button(button_pld, "Далее"))
    mssg = dict_settings[3][dict_settings[2]][0] + '\n' +
dict_settings[3][dict_settings[2]][1]
    vk.messages.send(user_id=event.user_id, random_id=event.random_id, mes-
sage=mssg, keyboard=kb)
```

Листинг 28. Завершение модуля «Словарь»

```
if event.text.lower() in unit_labels or event.text.lower() == 'готово':
    pld = get_payload()
    for e in all_users_set:
        if e[0] == event.user_id:
            unit = pld.find('U')
            level = pld.find('B')
            e[1] = int(pld[unit + 1])
            e[2] = "B" + pld[level + 1]
            #unit_nb = int(pld[unit + 1])
            #lvl_nm = "B" + pld[level + 1]
            if event.text.lower() == 'готово':
                unit_stat_update(e[1], e[2], event.user_id, 'словарь',
'Yes')
            kb = gen_unit_keyboard(e[1], e[2])
            mssg = 'Вы выбрали юнит ' + str(e[1]) + ' уровня ' + e[2] +'
\n\n' \
           'Выберите изучение словарных слов или упражнение. \n' \
           'Вы также можете вернуться к выбору разделов'
            vk.messages.send(user_id=event.user_id,
random_id=event.random_id, message=mssg, keyboard=kb)
```

3.8. Модули «Тестовое задание» и «Выбор пропущенного слова»

Данный модуль начинается после того, как чат бот распознает команду пользователя «Упражнение 1», «Упражнение 2», или «Упражне-

ние 3. При этом определяется какой пользователь ввел команду и для какого уровня и раздела, при этом текущие настройки пользователя (номер раздела e[1], значение уровня e[2], счет пользователя e[4], порядковый номер задания e[5], отметка последнего вопроса e[3] и список словарных слов e[7]) в списке all_users_set обновляются. Список вопросов и ответов извлекается из базы данных с помощью функции get_vocab() для модуля «Тестовое задание» и функции get_exer() для модуля «Выбор пропущенного слова». Затем пользователю отправляется сообщение с первым заданием, ответы для которого формируются в случайном порядке с помощью функции form_random() (код представлен на листинге 29).

Листинг 29. Начало модуля «Тестовое задание»

```
if event.text.lower() == 'упражнение 1' or event.text.lower() == 'упражне-
ние 2':
    pld = get_payload()
    for e in all_users_set:
        if e[0] == event.user_id:
            unit = pld.find('U')
            level = pld.find('B')
            e[1] = int(pld[unit + 1])
            e[2] = "B" + pld[level + 1]
            e[7] = get_vocab(e[1], e[2], event.text)
            e[5] = 0
            e[4] = 0
            e[3] = 0
            mssg = e[7][0][0]
            corr_word_lb = e[7][e[5]][1]
            corr_word_pld = "{\"button\": \"enru" + str(e[5]) + "cor-
rect\"}"
            wrong_word_lb = e[7][e[5] + 1][1]
            wrong_word_pld = "{\"button\": \"enru" + str(e[5]) + "wrong\"}"

            answ_set = form_rand(corr_word_lb, corr_word_pld,
wrong_word_lb, wrong_word_pld)

            p = "{\"button\": \"bEnruU" + str(e[1]) + e[2] + "\"}"

            kb = get_keyboard(gen_kb(answ_set, p))
            vk.messages.send(user_id=event.user_id, ran-
dom_id=event.random_id, message=mssg, keyboard=kb)
```

Если пользователь выбирает один из вариантов ответа хранимых в списке всех английских и русских слов all_eng_words и all_rus_words, система определяет какой пользователь ее ввел, для какого упражнения, какого раздела и какого уровня, затем обновляются текущие настройки модуля в all_users_set и enru_exer_settings либо в gaps_exer_settings (для мо-

дулей «Тестовое задание» либо «Выбор пропущенного слова», соответственно). В случае если пользователь прошел последнее задание из списка, то модуль переходит к завершению, иначе, исходя из текущих настроек пользователя, система определяет какое следующее задание необходимо послать в сформированном сообщении. Фрагмент реализации представлен на листинге 30.

Листинг 30. Обработки сообщения при выборе ответа для модуля «Тестовое задание»

```
if event.text.lower() in all_eng_words or event.text in all_rus_words:
    pld = get_payload()
    enru_chk = pld.find('enru')
    qst_chk = pld.find('qst')
    if enru_chk > 0:
        enru_exer_settings = []
        for e in all_users_set:
            if e[0] == event.user_id:
                e[5] = int(pld[enru_chk + 4])
                corr_chk = enru_chk + 5
                if pld[corr_chk] == 'c':
                    e[4] +=1
                e[6] = e[5] + 1
                enru_exer_settings.append(e[1]) #0 unit
                enru_exer_settings.append(e[2]) #1 lvl
                enru_exer_settings.append(e[3]) #2 last_f
                enru_exer_settings.append(e[4]) #3 count
                enru_exer_settings.append(e[5]) #4 w_id
                enru_exer_settings.append(e[6]) #5 curr_w_id
                enru_exer_settings.append(e[7]) #6 word_set
        if enru_exer_settings[5] <= len(enru_exer_settings[6])-1:
            enru_exer_settings[4] += 1
            mssg = enru_exer_settings[6][enru_exer_settings[4]][0]
            corr_word_lb = enru_exer_settings[6][enru_exer_settings[4]][1]
            corr_word_pld = "{\"button\": \"enru" + \
                            str(enru_exer_settings[4]) + "correct\"}"
            if enru_exer_settings[5] == len(enru_exer_settings[6])-1:
                enru_exer_settings[2] = 1
                for e in all_users_set:
                    if e[0] == event.user_id:
                        e[3] = 1
                wrong_word_lb = enru_exer_settings[6][0][1]
            else:
                wrong_word_lb =
enru_exer_settings[6][enru_exer_settings[4]+1][1]
            wrong_word_pld = "{\"button\": \"enru" + \
                             str(enru_exer_settings[4]) + "wrong\"}"
            answ_set = form_rand(corr_word_lb, corr_word_pld,
                                 wrong_word_lb, wrong_word_pld)
            p = "{\"button\": \"bEnruU" + str(enru_exer_settings[0]) \
                + enru_exer_settings[1] + "\"}"
            kb = get_keyboard(gen_kb(answ_set, p))
            vk.messages.send(user_id=event.user_id,
                             random_id=event.random_id,message=mssg,
                             keyboard=kb)
```

Если пользователь выбрал команду «Закончить», результат прохождения упражнения записывается в базу данных с помощью функции unit_stat_update(), и пользователя возвращают в меню соответствующего раздела, как описано на листинге 31.

Листинг 31. Завершение модулей «Тестовое задание» и «Выбор пропущенного слова»

```
if event.text.lower() == 'закончить':
    pld = get_payload()
    for e in all_users_set:
        if e[0] == event.user_id:
            enru_chk = pld.find('Enru')
            qst_chk = pld.find('Qst')
            if enru_chk > 0:
                if e[7][0][0] in all_eng_words:
                    unit_stat_update(e[1], e[2], event.user_id, 'ex1',
e[4])
                elif e[7][0][0] in all_rus_words:
                    unit_stat_update(e[1], e[2], event.user_id, 'ex2',
e[4])
                mssg = "Your score is: " + str(e[4]) + "\n"
            elif qst_chk > 0:
                unit_stat_update(e[1], e[2], event.user_id, 'ex3', e[4])
                mssg = "Your score is: " + str(e[4]) + "\n"
            kb = gen_unit_keyboard(e[1], e[2])
            mssg = mssg + 'Вы выбрали юнит ' + str(e[1]) + ' уровня ' +
e[2] + ' \n\n' \
                    'Выберите изучение словарных слов или упражнение. \n' \
                    'Вы также можете вернуться к выбору разделов'
            vk.messages.send(user_id=event.user_id,
random_id=event.random_id, message=mssg, keyboard=kb)
```

3.9. Модуль «Статистика»

Данный модуль начинается после того, как чат-бот распознает команду пользователя, соответствующую тексту кнопки вызова статистики для определенного раздела из списка stat_b1_unit_labels или stat_b2_unit_labels. При этом определяется какой пользователь ввел команду и с помощью функции get_unit_stats() из базы данных извлекается необходимая статистика, которая затем отправляется пользователю (код представлен на листинге 32).

3.10. Обработка неверных команд

Для того, чтобы определить, правильную ли команду ввел пользователь при взаимодействии с чат-ботом, система сверяет полученную коман-

ду со списком возможных команд possible_answers либо списком всех английских и русских слов all_eng_words и all_rus_words. Если полученная команда не совпадает ни с одним элементом из списков, пользователю отправляется инструкция со списком команд для взаимодействия (код представлен на листинге 33).

Листинг 32. Реализация модуля «Статистика»

```
if event.text.lower() in stat_b1_unit_labels or event.text.lower() in
stat_b2_unit_labels:
    index = event.text.find('t')
    unit_nb = event.text[index + 2]
    level = ''
    if event.text in stat_b1_unit_labels:
        level = 'b1'
        kb = get_keyboard(stat_unit_buttons(stat_b1_unit_labels))
    elif event.text in stat_b2_unit_labels:
        level = 'b2'
        kb = get_keyboard(stat_unit_buttons(stat_b2_unit_labels))
    user_stats = get_unit_stats(unit_nb, level, event.user_id)
    mssg = 'Ваша статистика по юниту ' + str(unit_nb) + ' уровня ' + level
+ '\n' + user_stats
    vk.messages.send(user_id=event.user_id, random_id=event.random_id, mes-
sage=mssg, keyboard=kb)
```

Листинг 33. Обработка неверных команд

```
if not(event.text.lower() in possible_answers or event.text.lower()
       in all_rus_words or event.text.lower() in all_eng_words):
    mssg = 'Приветствую, это справка по командам управления чат-бота.\n' \
           'Если Вы попали сюда, значит скорее всего ввели неправильную ' \
           'команду, \nВы можешь вернутся в Главное меню, набрав ' \
           'команду "Начать" или "Start" или нажав на кнопку \n Вы также' \
           ' можете ввести такие команды как: \n ' \
           '- "Начать обучение" или "Start study" \n' \
           '- "Начать тест"\n' \
           '- "B1" либо "B2" \n' \
           '- "Инструкция" или "Help"'
    kb = get_keyboard(keyboard1button("Назад в главное меню"))
    vk.messages.send(user_id=event.user_id, random_id=event.random_id,
                     message=mssg, keyboard=kb)
```

Выводы по третьей главе

На основе разработанной архитектуры была реализована система чат-бота. Была выполнена начальная настройка чат-бота, а также реализованы вспомогательные функции, основной цикл системы и модули системы.

4. ТЕСТИРОВАНИЕ

Для тестирования системы применялось функциональное тестирование, т.е. тестирование программного обеспечения в целях проверки реализуемости функциональных требований. В таблице 9 приведен протокол тестирования некоторых аспектов работы системы.

Табл. 9. Функциональное тестирование чат-бота

№	Аспект работы	Действия	Результат	Тест прой ден?
1	Вывод теста	1. Выбрать команду «Начать тест».	Выводится первый вопрос с вариантами ответа и клавиатура с вариантами ответа	Да.
2	Вывод следующего вопроса в тесте	1. Выбрать команду «A», «B», «C» или «D».	Выводится следующий вопрос с вариантами ответа и клавиатура с вариантами ответа	Да.
3	Вывод разделов уровня	1. Выбрать команду «B1» или «B2-C1».	Выводится клавиатура с шестью вариантами разделов	Да.
4	Вывод упражнений раздела	1. Выбрать команду «Unit 1», «Unit 2», «Unit 3», «Unit 4», «Unit 5» или «Unit 6».	Выводится клавиатура с вариантами упражнений и словаря	Да.
5	Вывод словарных слов	1. Выбрать команду «Словарь».	Выводится первое словарное слово и его перевод, и клавиатура для навигации	Да.
6	Вывод следующего словарного слова	1. Выбрать команду «Далее».	Выводится следующее словарное слово и его перевод, и клавиатура для навигации	Да
7	Вывод упражнения	1. Выбрать команду «Упражнение 1», «Упражнение 2» или «Упражнение 3».	Выводится первое слово для тестового задания и варианты его перевода	Да.
8	Вывод следующего вопроса в упражнении	1. Выбрать один из предоставленных вариантов на предыдущий вопрос	Выводится следующий вопрос из упражнения	Да

Окончание табл. 9

№	Аспект работы	Действия	Результат	Тест прой ден?
9	Вывод результатов теста	1. Выбрать команду «Закончить».	Выводится результат теста и клавиатура для навигации	Да.
10	Вывод результатов тестового упражнения или упражнения на пропуск слов	1. Выбрать команду «Завершить».	Выводится результат тестового упражнения и клавиатура с вариантами упражнений и словаря.	Да.
11	Вывод инструкции	1. Ввести команду «Инструкция».	Выводится список возможных команд управления чат-ботом.	Да.
12	Вывод статистики по пользователю	1. Выбрать кнопку «unit 1(b1)», «unit 2(b1)», «unit 3(b1)», «unit 4(b1)», «unit 5(b1)», «unit 6(b1)», «unit 1(b2)», «unit 2(b2)», «unit 3(b2)», «unit 4(b2)», «unit 5(b2)», «unit 6(b1)»	Выводится статистика по соответствующему разделу	Да
13	Ввод неверной команды	1. Ввести команду с орфографическими ошибками, либо команду, не определённую в чат-боте	Выводится список возможных команд управления чат-ботом	Да

Выводы по четвертой главе

В рамках тестирования было проведено функциональное тестирование системы чат-бота. Тестирование было пройдено успешно.

ЗАКЛЮЧЕНИЕ

В данной работе была разработана система чат-бота для изучения академического английского языка. Чат-бот был реализован для социальной сети «ВКонтакте». Код системы составил свыше 1000 строк на языке Python.

На основе требований были выделены варианты использования системы, спроектирована архитектура системы и базы данных и определены диаграмма последовательности и диаграммы деятельности необходимых модулей.

На основе спроектированных диаграмм были рассмотрены методы реализации системы чат-бота.

Также было успешно проведено функциональное тестирование разработанной системы.

ОСНОВНЫЕ РЕЗУЛЬТАТЫ.

1. Выполнен анализ предметной области и произведен обзор существующих решений.

2. Спроектирован чат-бот.

3. Реализован чат-бот.

4. Проведено тестирование разработанного чат-бота.

ЛИТЕРАТУРА

1. Ju Long, Michael Juntao Yuan, Hsun-Ming Lee. How to Program a Chatbot – An Introductoty Project and Student Perceptions. // Issues in Informing Science and Information Technology, Vol. 16, 2019. – P.1-31.

2. M. Gottlieb, G. Ernst-Slavit. Academic Language in Diverse Classrooms: Definitions and Contexts. Corwin, 2014. – P.4-5.

3. Матвеева Н.Ю., Золотарюк А.В. Технологии создания и применения чат-ботов // Научные записки молодых исследователей, №1. 2018. – P.28-30.

4. График запросов по чат-ботам в Google Trends. [Электронный ресурс]. URL: https://trends.google.co.in/trends/explore?date=today%205-y&q=Chatbot (дата обращения: 19.03.2020).

5. Architecture for public service chatbots / ISA^2 Programme. [Электронный ресурс]. URL: https://joinup.ec.europa.eu/sites/default/files/news/2019-09/ISA2_Architecture%20for%20public%20service%20chatbots.pdf (дата обращения: 19.03.2020).

6. 8 Strategies for Teaching Academic Language. [Электронный ресурс]. URL: https://www.edutopia.org/blog/8-strategies-teaching-academic-language-todd-finley (дата обращения: 19.03.2020).

7. Ellis Pratt. Artificial Intelligence and Chatbots in Technical Communication – A primer. // The Intelligent Information Blog. URL: https://intelligent-information.blog/en/artificial-intelligence-and-chatbots-in-technical-communication-a-primer (дата обращения: 19.03.2020).

8. S.A. Abdul-Kader, Dr. J. Woods. Survey on Chatbot Design Techniques in Speech Conversation Systems. // International Journal of Advanced Computer Science and Applications, Vol. 6, No. 7, 2015. – P.72-80.

9. K. Nimavat, T. Champaneira. Chatbots: An overview Types, Architecture, Tools and Future Possibilities.// International Journal for Scientific Research & Development, Vol. 5, No. 07, 2017. – P.1019-1026.

10. Duolngo. [Электронный ресурс]. URL: https://www.duolingo.com (дата обращения: 19.03.2020).

11. Dragon-English. [Электронный ресурс]. URL: https://dragon-english.ru (дата обращения: 19.03.2020).

12. Платформа Facebook Messenger. URL: https://developers.facebook.com/docs/messenger-platform/introduction (дата обращения: 21.03.2020).

13. Viber API Documentation. Get Started. URL: https://developers.viber.com/docs/general/get-started/ (дата обращения: 21.03.2020).

14. Viber API Documentation. URL: https://viber.github.io/docs/ (дата обращения: 21.03.2020).

15. API для чат-ботов. URL: https://vk.com/dev/bots_docs (дата обращения: 21.03.2020).

16. API для чат-ботов, часть 2. URL: https://vk.com/dev/bots_docs_2 (дата обращения: 21.03.2020).

17. Боты для сообществ. URL: https://vk.com/dev/bots (дата обращения: 21.03.2020).

18. Клавиатуры для ботов, часть 3. URL: https://vk.com/dev/bots_docs_3 (дата обращения: 21.03.2020).