МИНИСТЕРСТВО ПРОСВЕЩЕНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ
ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ им. А. И. ГЕРЦЕНА»


# Выпускная квалификационная работа

на тему

«Разработка чат-бота для справочного сайта по онлайн-играм»

Обучающейся 4 курса
очной формы обучения
направление подготовки:
09.03.01 Информатика и вычислительная техника
направленность (профиль):
Технологии разработки программного обеспечения
Пляскиной Ульяны Сергеевны

Руководитель выпускной квалификационной работы:
к. ф.-м. н., доцент
Жуков Николай Николаевич

Санкт-Петербург
2021

СОДЕРЖАНИЕ

# ВВЕДЕНИЕ

Актуальность темы выпускной квалификационной работы обусловлена тем, что в настоящее время доступность разнообразных гаджетов достигла необычайных масштабов, тем самым подтолкнув к развитию игровую индустрию. В связи с этим остро встала необходимость выбора наиболее подходящего игрового проекта лично для каждого человека с учётом его интеллектуального развития, социального статуса, а отсюда индивидуальных потребностей.

Цель работы – разработать чат-бот для справочного сайта по онлайн-играм помогающий пользователям в принятии решения при выборе игрового проекта исходя из личных предпочтений.

Объект исследования: искусственный интеллект.

Предмет исследования: разработка чат-бота.

Для реализации поставленной цели были сформулированы следующие задачи:

- провести анализ готовых решений и методов их реализации;
- разработать структуру программного продукта;
- разработать клиентскую и серверную часть чат-бота;
- размещение чат-бота на сервере;
- интеграция чат-бота в справочный сайт по онлайн-играм.

Практическая значимость выпускной квалификационной работы состоит в том, что её результаты в виде чат-бота могут использоваться для помощи пользователям в принятии решения при выборе наиболее подходящего игрового проекта с учётом индивидуальных потребностей.

Объём и структура выпускной квалификационной работы. Работа состоит из введения, двух глав основного текста, заключения, библиографического списка, включающего 12 источников и N приложений. В работе представлено 26 рисунков.

В первой главе, которая является теоретической, проведён анализ различных источников по исследуемой теме, а также доступных решений и инструментов для реализации.

Во второй главе, исследовательской, разработан чат-бот, проведены тесты, бот размещён на сервере и интегрирован в сайт.

В заключении изложены основные результаты работы, сформулированы выводы, имеющие практическое значение.

# ГЛАВА 1. АНАЛИЗ ЗАДАЧИ И ПОИСК СУЩЕСТВУЮЩИХ ПУТЕЙ ЕЁ РЕШЕНИЯ

## 1.1 Обзор существующих решений поставленной задачи

В настоящем параграфе проведён анализ научной литературы и интернет-ресурсов, в которых рассматриваются вопросы разработки чат-ботов. Разработка этих объектов является одной из задач, которые решаются в рамках научного направления «искусственного интеллекта». Чат-боты помогают пользователю в принятии решений.

Для понимания сути и функционала чат-ботов необходимо проанализировать различные варианты описания этих объектов специалистами в области искусственного интеллекта (ИИ). Анализ литературы [10, 2], интернет-ресурсов [5, 8] позволил выявить два основополагающих направления его изучения. Первое – со стороны пользователя. В данном случае чат-бот – это некий интерфейс, с помощью которого происходит взаимодействие между человеком и ботом. Создаётся впечатление конструктивного и осмысленного диалога с видимостью того, что бот понимает человеческий язык и отвечает пользователю. Второе – со стороны разработчика. В этом случае представляет интерес для изучения набор чётко прописанных команд, функций, методов результатом работы которых является создание некоего искусственного интеллекта, способного взаимодействовать с пользователем и предоставлять ему заранее прописанную информацию. В результате можно сделать вывод, что чат-бот – это некий искусственный интеллект. Для того, чтобы приступить к его разработке необходимо углубиться и понять, как специалистами в области ИИ трактуется понятие искусственный интеллект и какие методы являются основными в этом научном направлении.

Первое упоминание термина искусственный интеллект приходится на 1956 год на Дартмутской конференции. Молодой учёный Джон Маккарти трактовал

смысл термина «искусственный интеллект» как технологию и науку о создании интеллектуальных компьютерных программ. По итогу конференции участники сошлись во мнении о том, что ИИ может представлять из себя любой аспект разумной деятельности человека, который чётко прописан таким образом, что машина может его сымитировать [1].

С течением времени данный термин не раз трактовался в несколько ином смысле. Так, например, в британской энциклопедии «Britannica» дана следующая формулировка – «способность цифровых компьютеров решать задачи, которые обычно ассоциируются с высоко интеллектуальными возможностями человека». В свою очередь один из исследователей в области ИИ Эдвард Фейгенбаум считал, что ИИ представляет из себя компьютерную систему, обладающую возможностями, связанными с человеческим разумом, будь то понимание языка или способность к обучению [3, 11]. Несмотря на различные трактовки, во всех этих случаях присутствует единое мнение о том, что искусственный интеллект – это компьютерная система, способная имитировать поведение, традиционно свойственное человеку.

Существуют разные подходы к исследованию и реализации искусственного интеллекта. Наибольшую практическую значимость и востребованность получили такие направления ИИ, как: обработка текста и речи на естественном языке, машинное обучение и виртуальные ассистенты. Рассмотрим каждый из них подробнее.

Обработка естественного языка представляет из себя анализ с последующим синтезом. В контексте искусственного интеллекта анализ представляет из себя понимание языка, в то же время синтез представлен в виде генерации грамотного текста или синтезирования речи. Главной целью данного направления является изучения методов анализа и синтеза естественного языка. В настоящее время самыми распространёнными сферами, в которых используется данное направление ИИ, являются голосовые помощники и машинный перевод [7].

Следующим рассматриваемым подходом является машинное обучение. Часто можно столкнуться с мнением о том, что искусственный интеллект и машинное обучение — это неотъемлемые части друг друга, но данное мнение в корне не верно. Машинное обучение — это всегда искусственный интеллект, но искусственный интеллект — это не всегда машинное обучение. Суть машинного обучения заключается в том, что присутствует процесс самостоятельного получения знаний интеллектуальной системой в процессе её работы [4]. Но далеко не каждая компьютерная система способна к самообучению. Таким образом можно сделать вывод о том, что машинное обучение является лишь составной частью искусственного интеллекта и не является определяющим фактором. Несмотря на это не стоит отрицать того факта, что данное направление было одним из основных с самого начала развития искусственного интеллекта. Это направление является один из самых перспективных. В будущем возможно появление роботов, способных самостоятельно обучаться, а программистам не придётся прописывать огромный объём кода, поскольку машина сама обучится всему, что ей необходимо на основе анализа данных. Но в настоящий момент дела обстоят не так грандиозно и главной сферой, в которой используется машинное обучение, является реклама, а именно технология показа релевантной рекламы, основанная на анализе данных пользователей и отсеивании информации, которая по мнению машины является неинтересной для конкретного человека [6].

В заключении рассмотрим виртуальных ассистентов, которые для данной работы имеют наибольшую степень значимости, потому что именно им посвящена работа. Основными примерами таких ассистентов являются всевозможные чат-боты и виртуальные помощники, главной целью которых является в первую очередь автоматизированное обслуживание клиентов. В перспективе виртуальные ассистенты могут быть полностью построены на основе машинного обучения, но надёжность таких ассистентов вызывает сомнение. На данный момент технологии машинного обучения не так хорошо изучены, чтобы всецело довериться машине, в любом случае необходим контроль над результатом [10]. В настоящее время чаще

всего виртуальные ассистенты встречаются двух видов. Первый – программа, полностью контролируемая человеком, то есть все команды заранее прописаны, а способность к самообучению отсутствует. Второй – гибридные виртуальные ассистенты. Они представляют из себя сочетание чётко прописанной программы и способность к обучению, но конечный результат всё равно контролируется человеком [5]. Примерами ассистентов первого вида являются всевозможные чат-боты и поиск по документу, а второго – голосовые виртуальные помощники. В рамках данной выпускной квалификационной работы (ВКР) разработан виртуальный ассистент первого вида, а именно чат-бот.

Реализация чат-бота данного типа возможна с помощью средства поддержки принятия решения, а именно дерева принятия решений. Оно представляет из себя иерархическую древовидную структуру, состоящую из правил вида «Если…, то…» [9]. Само дерево состоит из узлов и листьев. В узлах находится решающее правило, а в листьях непосредственно решение. Для того, чтобы попасть на конкретный лист необходимо удовлетворить всем правилам конкретной ветви. Каждый лист имеет единственно верный путь, что обеспечивает единственно верное решение. Основными сферами применения дерева решений являются машинное обучение и анализ данных.

В процессе изучения вопроса, которому посвящена ВКР, автором был проведён обширный анализ интернет-ресурсов на предмет поиска чат-ботов с функционалом поддержки принятия решения по выбору интересующей пользователя компьютерной игры. Были найдены отдельные разработки чат-ботов, которые используются в сфере торговли, обслуживания населения, сфере образования и т.д. Вместе с тем найти чат-бот с функционалом, аналогичным тому, что разрабатывает автор данной работы среди проанализированных источников не удалось. Дополнительно следует отметить, что автору удалось найти опросы, созданные пользователями на различных площадках и форумах, посвящённых онлайн-играм, на основе которых их участники пытались дать рекомендации по выбору интересующих людей игр. Анализ материала опросов и рекомендаций

показал, что предоставляемая пользователю информация весьма низкого качества с точки зрения предложенных формулировок, ограниченности предлагаемых рекомендаций и т.д. Однако даже подобные опросы пользуются спросом у любителей компьютерных игр и как следствие автор сделал вывод о целесообразности разработки специального чат-бота, основным функционалом которого является поддержка принятия решения пользователем по выбору компьютерных игр, соответствующих его интересам, личным и профессиональным потребностям.

В следующем параграфе будет представлен материал, посвященный классификации существующих чат-ботов (по результатам анализа, проведенного автором ВКР), а также классификации и описанию инструментов актуальных для их разработки.

## 1.2 Классификация чат-ботов и методы их реализации

### 1.2.1 Классификация чат-ботов

В первую очередь чат-боты разделяют на две основные группы: стандартные и самообучающиеся.

К стандартным относятся те чат-боты, которые имеют заранее прописанные команды, в рамках которых происходит взаимодействие с пользователем. При вводе иного запроса взаимодействие становится невозможным и чат-бот бездействует вплоть до момента, когда к нему обратятся с понятной для него командой.

К самообучающимся относят тех чат-ботов, которые в процессе взаимодействия с пользователями обучаются на основании их запросов и расширяют собственный функционал. Эта способность заранее прописывается при разработке бота, что позволяет в дальнейшем чат-боту отвечать на самые нестандартные запросы. Такие боты представляют из себя искусственный

интеллект в стандартном его понимании и реализуются с помощью машинного обучения.

Помимо разделения на две основные группы также классифицируют чат-боты по формату их взаимодействия с пользователями. Примерами таких чат-ботов являются:

1. Текстовые. Данный вид считается самым гибким и максимально приближенным к беседе с реальным человеком. Чат-бот обрабатывает запрос, полученный от пользователя, и выдаёт наиболее релевантный ответ из заранее прописанных. Как правило такие боты способны к самообучению, но это не является обязательным критерием.
2. Встраиваемые. Обычно представляют из себя бота-подсказку в виде всплывающего окна в различных приложениях. Используются в основном при поиске определённого места, заказе товаров и т.д.
3. Кнопочные. Являются самым примитивным видом чат-ботов. Взаимодействие с пользователями происходит посредством заранее чётко прописанных команд, оформленных в виде кнопок. Чаще всего используются в различных мессенджерах и не имеют способности к самообучению.

### 1.2.2 Анализ инструментов для разработки чат-ботов

Не смотря на существование различных видов чат-ботов, алгоритм разработки у них примерно одинаковый: описание структуры чат-бота, выбор языка программирования, выбор среды разработки, разработка и размещение чат-бота на сервере для автономной работы. Рассмотрим каждый этап подробнее.

В первую очередь необходимо описать структуру будущего чат-бота. На данном этапе определяется назначение чат-бота, продумывается необходимый функционал, выбирается формат взаимодействия с пользователями. Именно на основании желаемой структуры чат-бота и необходимого функционала для

решения поставленной задачи в дальнейшем происходит выбор языка и среды разработки.

Чат-боты могут быть написаны на любом языке программирования. Главным условием является возможность использования web API. Одним из таких языков является – Python. Этот язык удобен тем, что для него уже написаны библиотеки, которые ориентированы на разработку чат-ботов. Благодаря этому разработка становится весьма удобной и нет необходимости в придумывании инновационных решений.

Далее необходимо выбрать среду для разработки. Наиболее распространённой и удобной средой разработки для Python является PyCharm. Главными достоинствами данной среды являются то, что она разработана специально для языка Python, кроссплатформенная и имеет бесплатную версию, которой достаточно для разработки чат-бота.

После выбора языка и среды разработки наступает этап непосредственно самой разработки чат-бота. Используя доступные библиотеки и модули реализовывается желаемый конечный продукт. Результатом разработки является чат-бот с полноценным функционалом, работающий на локальной машине.

Завершающим этапом является размещение чат-бота на сервере для обеспечения автономной работы без использования локальной машины. Главным фактором выбора хостинга является поддержка языка, на котором разработан бот. Одним из таких является облачная платформа Heroku. Данная платформа умеет работать с языком Python, а главным достоинством выступает бесплатный доступ. Размещение на сервере не составляет особого труда, а чат-бот будет работать круглые сутки, разве что при отсутствии активности в течении 30 минут, он будет уходить в спящий режим до следующего использования.

# ВЫВОДЫ ПО ГЛАВЕ 1

В первой главе был проведён анализ литературы и интернет-ресурсов направленных на разработку чат-ботов.

В первом параграфе главы автором ВКР был проведён анализ научного направления «искусственный интеллект». Были представлены определения, методы исследования и реализации данного направления, а также связь между ИИ и чат-ботами. Также был проведён анализ готовых решений по заданной теме данной ВКР.

Во втором параграфе была разобрана классификация чат-ботов по функционалу и формату взаимодействия с пользователями. Был проведён анализ инструментов для разработки чат-ботов с конкретными примерами, а также представлен алгоритм разработки чат-бота.

В второй главе данной ВКР будет представлена поэтапная разработка чат-бота и интеграция готового продукта в автоматизируемую систему – справочный сайт по онлайн-играм.

# ГЛАВА 2. ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ЧАТ-БОТА

## 2.1 Проектирование дерева решений

В первую очередь перед непосредственно самой разработкой необходимо продумать структуру чат-бота. Поскольку главной задачей разработанного чат-бота является поддержка принятия решений пользователя, то наиболее очевидным способом описания структуры является дерево решений.

Для разработки данного типа структуры необходимо классифицировать и обобщить основные потребности пользователей при выборе той или иной игры в жанре MMORPG.

Существует ряд основных отличительных особенностей присущих данному жанру игр, на которые обычно опираются пользователи при своём выборе, а именно:

1. Стилистика игры. Визуальная составляющая игры является основополагающей характеристикой при выборе будущего проекта. Существует устоявшееся в кругах разработчиков и игроков разделение игр данного жанра – западный стиль и восточный стиль. Западному стилю присущи более приглушённые цвета и средневековая стилистика. В противовес западному, восточная стилистика отличается невероятной красочностью, а сама игра пропитана восточной культурой. Именно данное разделение и станет первостепенным в принятии решения.
2. Вид камеры. Следующим важным фактором при выборе игры является положение камеры. Существует 3 основных вида камеры в играх жанра MMORPG: от первого лица, от третьего лица и изометрическая камера. Данный выбор будет следующим пунктом при принятии решения.
3. Система управления. Современные игры делятся на два типа управления таргет (от англ. target – цель) и нон-таргет (от англ. non-target – без цели) система. Таргет система представляет из себя систему управления с

направленной целью. Нон-таргет система характеризуется отсутствием чёткой направленности способностей на персонажа. Это разделение также является важным при выборе игры, поэтому присутствует в дереве решений.

4. Направленность игровой активности. Существует два вида игровой активности в играх жанра MMORPG, а именной ПВЕ (от англ. Player versus environment – игрок против окружения) и ПВП (от англ. Player versus player – игрок против игрока). В первом случае игроки сражаются и взаимодействуют с окружающим миром (подземелья, неигровые персонажи и т.д.), во втором случае игроки сражаются с другими игроками в открытом мире или выделенных зонах. Эта важная особенность также влияет на принятие решения при выборе проекта.

5. ПВЕ активность. В зависимости от того какую игровую активность выбрал пользователь, существуют также конкретные виды ПВЕ активностей, как:

- данжи (от англ. dungeon – подземелье) – обычно представлено ограниченной областью, в которой необходимо сражаться с врагом в составе маленькой группы игроков или в одиночку;
- рейды (от англ. raid) – сражение большой группы игроков с врагом в ограниченной области или открытом мире;
- гринд мобов (от англ. grind – молоть, mob или mobile object – подвижный объект) – представляет из себя убийство толп врагов.

Каждый из видов этих активностей является решающим фактором при выборе игры.

6. ПВП активность. Среди ПВП активностей также есть 3 основных вида:

- 1 vs 1 (от англ. versus – против) – сражение игрок против игрока;
- N vs N (от англ. number – количество) – сражение группы игроков против группы игроков;
- Битва гильдий – сражение между игровыми сообществами игроков в рамках одной игры.

Также как и с ПВЕ активностями, выбор одной из ПВП активностей является важным фактором в принятии решения.

Данная классификация возможных предпочтений при выборе игрового проекта в жанре MMORPG является основанием для разработки дерева принятия решений и выглядит следующим образом (см. Рисунок 1).

Поскольку игр от первого лица с таргет системой нет, данная ветка пропускает этап выбора системы управления и переходит к следующему выбору. Поскольку развитие веток после выбора вида камеры «От 3-го лица» и «Изометрия» аналогично, то на рисунке для ветки «Изометрия» схематично указано дальнейшее продолжение с помощью знака «…». Результатом работы чат-бота по заданной структуре является выдача пользователю списка рекомендованных игр и ссылки на информацию по данным играм на справочном сайте по онлайн-играм.

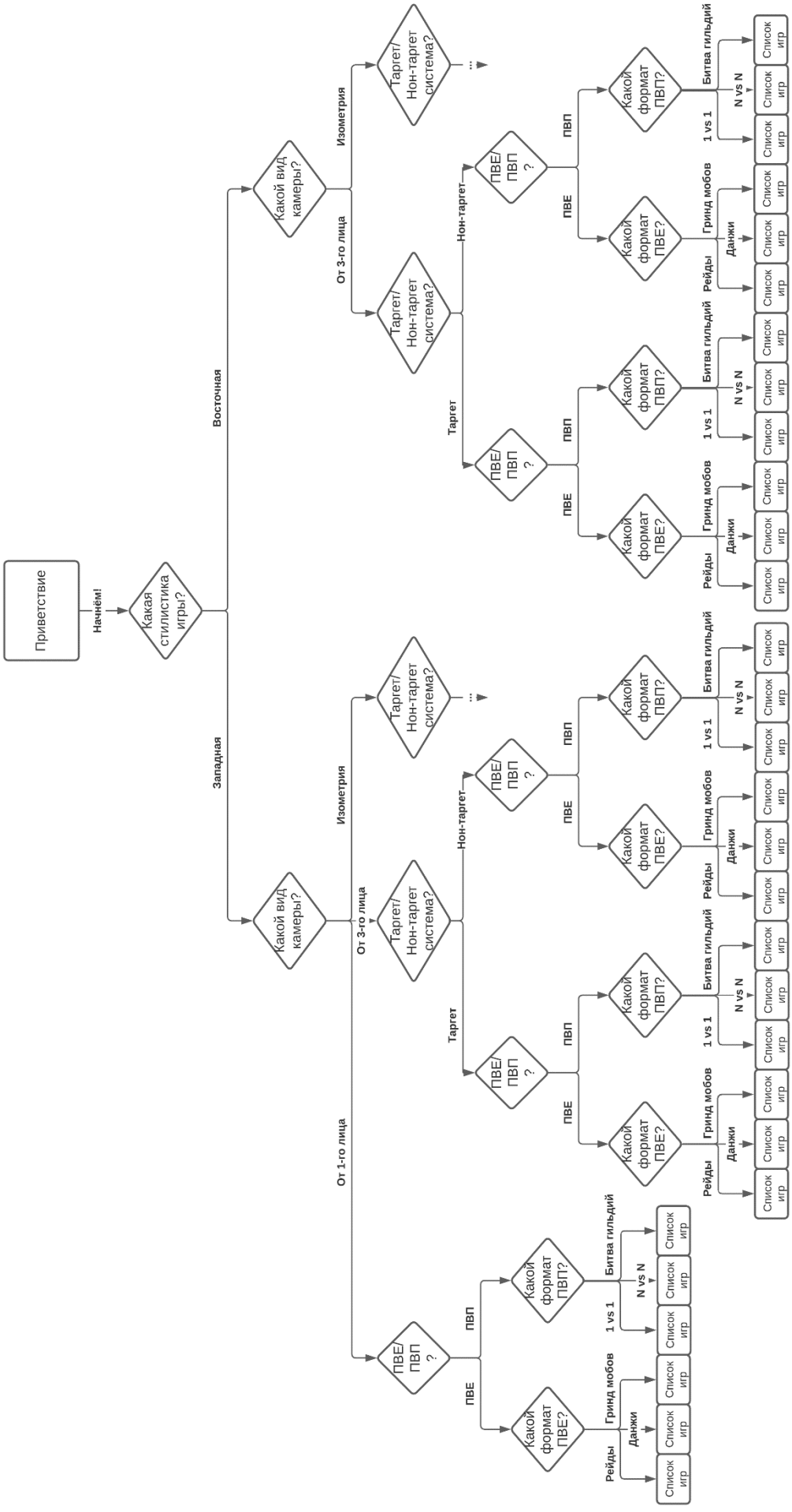


Рисунок 1 – Структура чат-бота

## 2.2 Создание бота

После того, как структура продумана, перед непосредственно написанием кода необходимо подготовить основу для будущего бота. Чат-бот, разработанный в рамках данной ВКР ориентирован на мессенджер Telegram. Данная платформа предоставляет возможность быстрого создания новых ботов с помощью специального официального чат-бота с названием «BotFather». Благодаря данному боту можно с лёгкостью создать собственного чат-бота с желаемым названием и идентификатором, получить ссылку для бота, а также уникальный токен, с помощью которого производится взаимодействие с данным ботом при разработке и дальнейшем его сопровождении. Помимо этого, к ознакомлению предоставляется ссылка на документацию по работе с API.

Для того, чтобы создать нового бота необходимо написать команду «/start» в диалоге с BotFather. После этого данный чат-бот начнёт свою работу и первым делом представит полный список команд, с помощью которых можно с ним взаимодействовать (см. Рисунок 2).

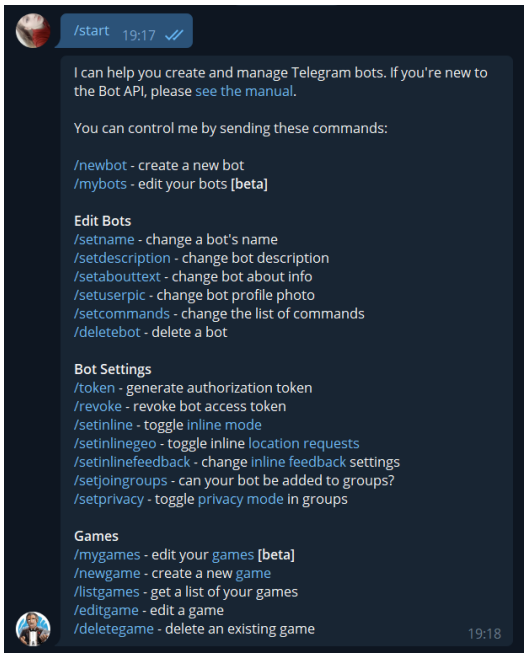


Рисунок 2 – Список команд BotFather

Следуя представленным функциональным возможностям, далее вводится команда «/newbot», которая означает создание или регистрацию нового бота. После ввода данной команды, BotFather сообщает о успешном создании нового бот и предлагает ввести желаемое название. Для разработанного в рамках данной ВКР чат-бота выбрано имя «ReviewMMORPG_bot» (см. Рисунок 3).

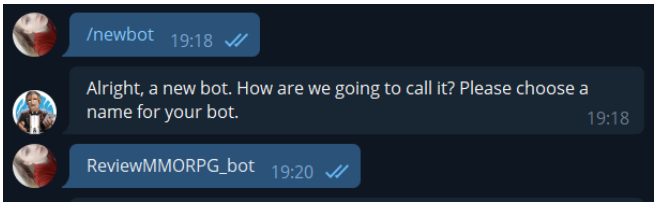


Рисунок 3 – Создание нового бота

После успешного подтверждения названия бота, необходимо также ввести username для бота, то есть уникальный идентификатор, по которому пользователи интернета и мессенджера Telegram смогут найти данный чат-бот и отличить его от других ботов, каналов или пользователей. Данного набора команд достаточно для BotFather, а потому он выводит сообщение об успешном создании нового бота, предоставляет ссылку на него и предлагает ознакомиться с дополнительным функционалом с помощью команды «help». Помимо этого, также предоставляется уникальный токен, который используется при разработке чат-бота и связывает написанный код со созданным ботом в Telegram (см. Рисунок 4). В целях безопасности токен, представленный на скриншоте, был скрыт.

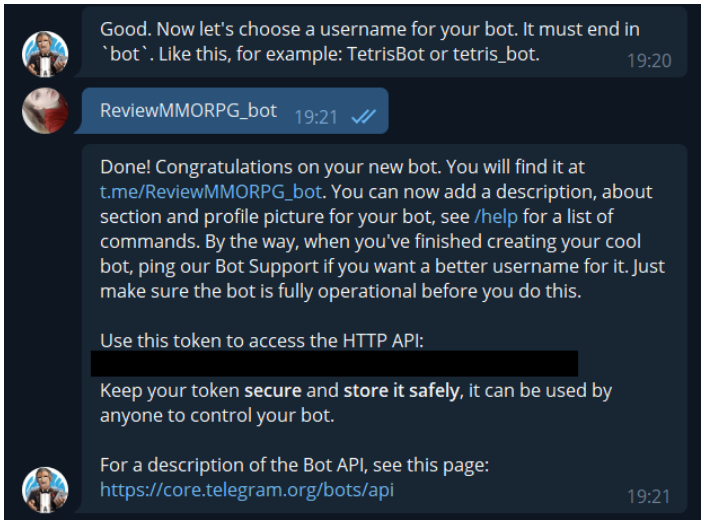


Рисунок 4 – Получение уникального токена бота

Дальнейшая настройка бота с помощью функционала BotFather является опциональной, но именно она придаёт чат-боту уникальный вид и создаёт более дружелюбное по отношению к пользователям представление о функциональных возможностях. Например, с помощью таких команд, как «/setuserpic» (см. Рисунок 5) и «/setabouttext» (см. Рисунок 6) можно настроить иконку бота и его описание соответственно. При добавлении иконки необходимо использовать формат фото или картинок, а при добавлении описания стоит ограничение в 120 символов.


Рисунок 5 – Добавление логотипа для бота

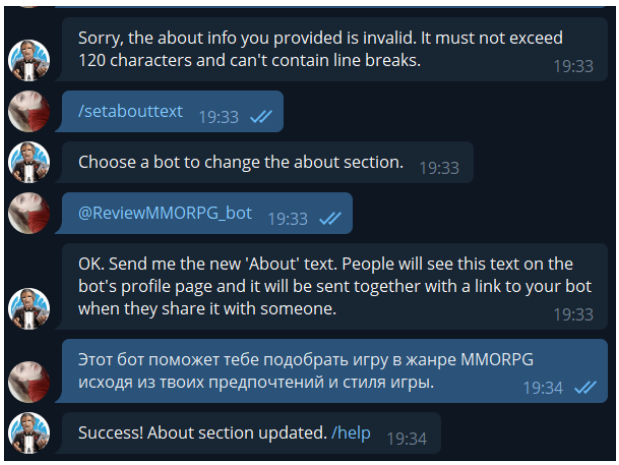


Рисунок 6 – Добавление описания для бота

На этом создание бота на базе мессенджера Telegram можно завершить, так как все необходимые настройки произведены. Далее необходимо приступить непосредственно к разработке и описании функционала чат-бота.

## 2.3 Разработка чат-бота

В первую очередь при разработке чат-бота следует определиться с помощью какой библиотеки будет реализовываться необходимый функционал. Поскольку разработка ведётся на языке программирования Python, то лучше всего воспользоваться специальной библиотекой, которая ориентирована на разработку чат-ботов для Telegram, а именно – pyTelegramBotAPI. Данная библиотека включает в себя все модули и возможности, которые позволят реализовать задуманный функционал чат-бота.

Далее важным этапом при создании бота является определение каким образом лучше представить интерфейс бота. Для того, чтобы пользователю не пришлось долго разбираться с командами, необходимыми для взаимодействия с ботом, существует более удобный и понятный способ подачи информации – кнопки.

Среди функционала, который предоставляет библиотека pyTelegramBotAPI, присутствует возможность создания inline кнопок. Они представляют из себя кнопки с выбором вариантов решений, привязанных к тексту, который отправляет бот. Объём текста ограничен 64 байтами. Данные кнопки относятся к типу InlineKeyboardMarkup и могут быть представлены в следующем виде:

- Callback-кнопки

  Когда пользователь нажимает кнопку обратного вызова, сообщение в чат не отправляется. Вместо этого бот просто получает соответствующий запрос. Получив запрос, бот может отображать результаты в уведомлении или предупреждении в верхней части экрана чата.

- Switch-кнопки

Нажатие переключателя на встроенной кнопке предлагает пользователю выбрать чат, открывает его и вставляет username бота в поле ввода. Также можно передать запрос, который будет вставлен с именем пользователя, который данный запрос отправляет. Это позволяет пользователям быстро получать оперативные результаты, которыми они могут поделиться.

- URL-кнопки

Кнопки этого типа имеют небольшой значок стрелки, чтобы помочь пользователю понять, что нажатие на кнопку URL-адреса откроет внешнюю ссылку. Перед открытием ссылки в браузере отобразится подтверждающее уведомление.

Поскольку главной функциональной особенностью разработанного чат-бота является реакция бота на выбор пользователя, то наиболее подходящим видом кнопок, для реализации данного функционала, являются – callback-кнопки. Они позволяют продемонстрировать пользователю, что выбранный им вариант принят во внимание и на основе этого выведется конечный результат. В результате интерфейс чат-бота будет представлен в виде inline-меню. Сообщения будут располагаться прямо над соответствующими им кнопками с вариантами решений, а всё, что потребуется от пользователя, — это выбрать и нажать одну из представленных кнопок.

Для реализации данного типа кнопок в первую очередь необходимо продумать каким образом и с помощью чего чат-бот будет принимать и обрабатывать входящие команды. В библиотеке pyTelegramBotAPI данный функционал представлен в виде обработчика сообщения – message_handler [12]. Он может как реагировать на определённые сообщения, так и на стикеры, смайлы и т.д. Существует также отдельный вид обработчиков – callback_query_handler. Данный вид используется для обработки callback запросов, которые могут быть реализованы в том числе с помощью callback-кнопок.

При разработке чат-бота в рамках данной ВКР были использованы следующие хэндлеры:

@bot.message_handler(commands=['start']) – данный вид используется для приветственного сообщения, которое отправляется при первом взаимодействии пользователя с чат-ботом (см. Рисунок 7).

```
@bot.message_handler(commands=['start'])
def start_message(message):
    bot.send_message(message.chat.id, 'Привет! Я чат-бот сайта Review MMORPG Games и я помогу '
                                      'подобрать тебе игру в жанре мморпг, которая подойдёт именно'
                                      ' для твоего стиля игры', reply_markup=keyboard_start)
```

Рисунок 7 – Код приветственного сообщения чат-бота

@bot.callback_query_handler(func=lambda call: True) – данный вид используется для взаимодействия с inline-меню, реализованном посредством callback-кнопок (см. Рисунок 8).

```
@bot.callback_query_handler(func=lambda call: True)
def callback_worker(call):
    if call.data == "start_test": #Начало теста
        keyboard = types.InlineKeyboardMarkup()
        key_eu = types.InlineKeyboardButton(text='Западная', callback_data='eu')
        keyboard.add(key_eu)
        key_asia= types.InlineKeyboardButton(text='Восточная', callback_data='asia')
        keyboard.add(key_asia)
        bot.send_message(call.message.chat.id, 'Какая стилистика игры тебе больше нравится?', reply_markup=keyboard)
```

Рисунок 8 – Код взаимодействия inline-меню с ботом

Далее необходимо разработать общую структуру для сообщений и сопровождающих их callback-кнопок. Основные моменты, которые можно выделить при создании данной структуры, это – создание клавиатуры, добавление необходимых кнопок и добавление сопровождающего сообщения. За создание клавиатуры отвечает команда:

keyboard = types.InlineKeyboardMarkup()

Здесь задаётся переменная, которая ассоциируется с клавиатурой, – keyboard, далее прописывается тип задаваемой клавиатуры – InlineKeyboardMarkup(), в данном случае inline. Далее необходимо создать кнопку и добавить её в клавиатуру. За создание отвечает команда:

key = types.InlineKeyboardButton(text=' ', callback_data=' ')

В данном случае key – переменная, в которую записывается создаваемая кнопка; types.InlineKeyboardButton – прописывается тип кнопки; text – сообщение, которое будет на кнопке; callback_data – идентификатор, соответствующий данной кнопке, через который будет происходить взаимодействие. После этого кнопка добавляется в клавиатуру. Происходит это с помощью команды:

keyboard.add(key)

В завершении необходимо добавить сообщение, которое будет сопровождаться с данным набором кнопок. Для этого необходимо ввести следующую команду:

bot.send_message(call.message.chat.id, ' ', reply_markup=keyboard)

bot.send_message – стандартная отправка сообщения в чат-бот, ' ' – здесь необходимо ввести сообщение, reply_markup=keyboard – обращение к созданной клавиатуре. В результате общая структура выглядит следующим образом (см. Рисунок 9).

```
keyboard = types.InlineKeyboardMarkup()
key_eu = types.InlineKeyboardButton(text='Западная', callback_data='eu')
keyboard.add(key_eu)
key_asia= types.InlineKeyboardButton(text='Восточная', callback_data='asia')
keyboard.add(key_asia)
bot.send_message(call.message.chat.id, 'Какая стилистика игры тебе больше нравится?', reply_markup=keyboard)
```

Рисунок 9 – Общая структура inline-меню

После того, как разработана структура, можно разработать непосредственно самого чат-бота. В первую очередь необходимо реализовать первое взаимодействие пользователя с ботом и приветственное сообщение (см. Рисунок 10).

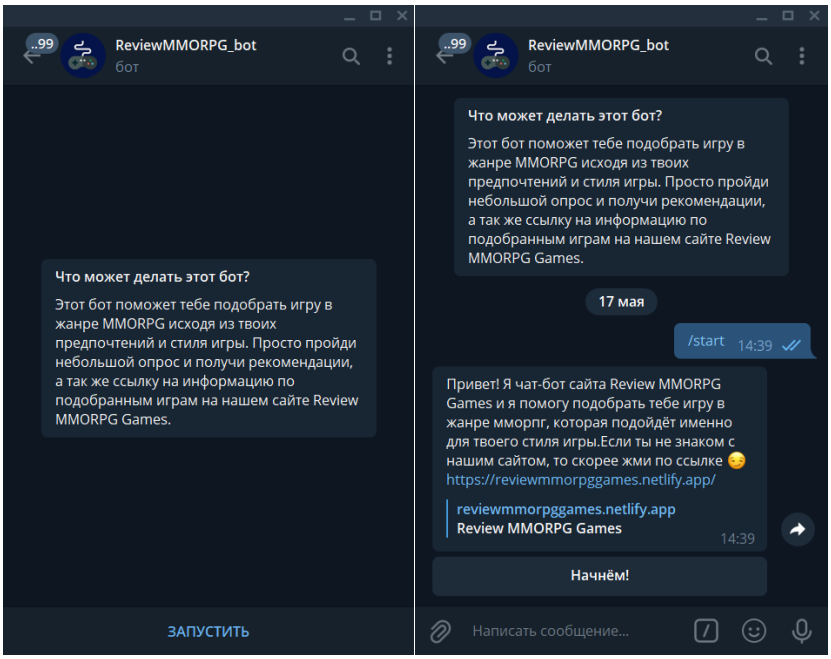


Рисунок 10 – Первое взаимодействие пользователя с чат-ботом

Реализуется данный функционал с помощью следующего кода:

```
import os
import telebot
from telebot import types
token = os.environ['TELEGRAM_TOKEN']

bot = telebot.TeleBot(token)

keyboard_start = types.InlineKeyboardMarkup()
key_start = types.InlineKeyboardButton(text='Начнём!',
callback_data='start_test')
keyboard_start.add(key_start)

@bot.message_handler(commands=['start'])
def start_message(message):
    bot.send_message(message.chat.id,
'Привет! Я чат-бот сайта Review MMORPG Games, и я помогу '
'подобрать тебе игру в жанре мморпг, которая подойдёт именно'
' для твоего стиля игры.'
'Если ты не знаком с нашим сайтом, то скорее жми по ссылке 😉\n'
'https://reviewmmorpggames.netlify.app/',
reply_markup=keyboard_start)

bot.polling()
```

Для перехода к основной функциональной особенности данного чат-бота – помощь пользователю в выборе подходящего для него игрового проекта в жанре MMORPG достаточно просто нажать кнопку «Начнём!» В первую очередь

пользователю будет представлен выбор стилистики желаемой игры между западной и восточной (см. Рисунок 11).

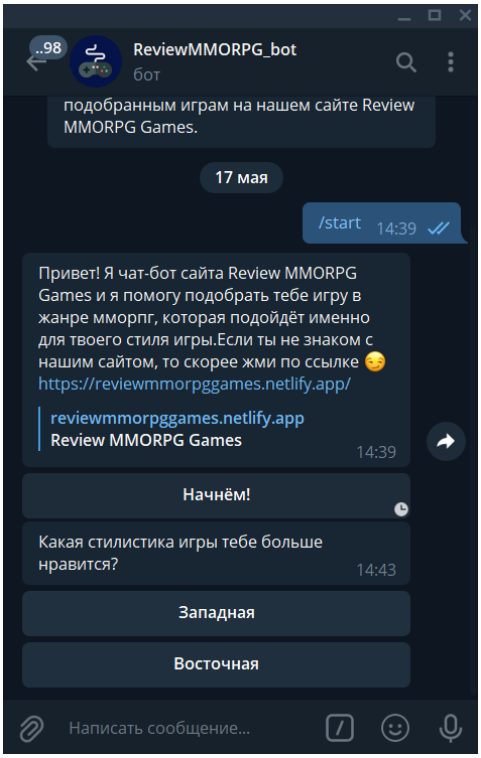


Рисунок 11 – Выбор стилистики игры

Данная клавиатура и сообщение вызывается благодаря следующему коду:

```
@bot.callback_query_handler(func=lambda call: True)
def callback_worker(call):
    if call.data == "start_test": #Начало теста
        keyboard = types.InlineKeyboardMarkup()
        key_eu = types.InlineKeyboardButton(text='Западная',
         callback_data='eu')
        keyboard.add(key_eu)
        key_asia= types.InlineKeyboardButton(text='Восточная',
         callback_data='asia')
        keyboard.add(key_asia)
        bot.send_message(call.message.chat.id, 'Какая стилистика
         игры тебе больше нравится?', reply_markup=keyboard)
```

Предположим, что пользователь предпочитает западную стилистику, тогда следующее, что чат-бот предложит ему будет выбор предпочитаемого вида камеры. В боте представлены следующие варианты: «От 1-го лица», «От 3-го лица» и «Изометрия» (см. Рисунок 12).

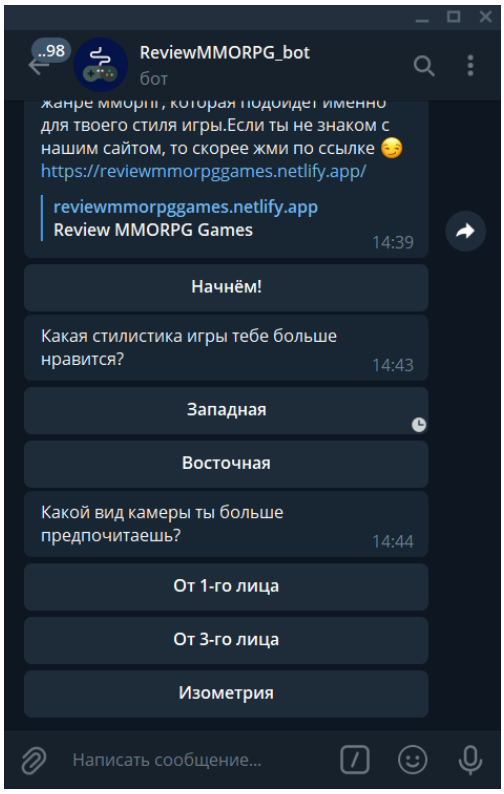


Рисунок 12 – Выбор вид камеры

Для того, чтобы вызвать данную клавиатуру используется следующий набор команд:

```
#Выбор стилистики игры
if call.data == "eu": #Выбран "Западный" стиль
   keyboard = types.InlineKeyboardMarkup()
   key_typeCamera_1 = types.InlineKeyboardButton(text='От 1-го
лица', callback_data='eu_typeCamera_1')
   keyboard.add(key_typeCamera_1)
   key_typeCamera_2 = types.InlineKeyboardButton(text='От 3-го
лица', callback_data='eu_typeCamera_2')
   keyboard.add(key_typeCamera_2)
   key_typeCamera_3 = types.InlineKeyboardButton(text='Изометрия',
callback_data='eu_typeCamera_3')
   keyboard.add(key_typeCamera_3)
   bot.send_message(call.message.chat.id, 'Какой вид камеры ты
больше предпочитаешь?', reply_markup=keyboard)
```

Если же при первом выборе пользователь предпочёл восточный стиль игры, то код дополняется следующими командами:

```
elif call.data == "asia": #Выбран "Восточный" стиль
   keyboard = types.InlineKeyboardMarkup()
   key_typeCamera_2 = types.InlineKeyboardButton(text='От 3-го
лица', callback_data='asia_typeCamera_2')
   keyboard.add(key_typeCamera_2)
```

```
   key_typeCamera_3 = types.InlineKeyboardButton(text='Изометрия',
callback_data='asia_typeCamera_3')
   keyboard.add(key_typeCamera_3)
   bot.send_message(call.message.chat.id, 'Какой вид камеры ты
больше предпочитаешь?', reply_markup=keyboard)
```

Вызов следующих клавиатур происходит посредством аналогичного кода. Для примера выберем вид камеры «От 3-го лица», тогда следующим выбором будет – система управления в игре. На данный момент самыми распространёнными системами управления являются таргет и нон-таргет система (см. Рисунок 13).

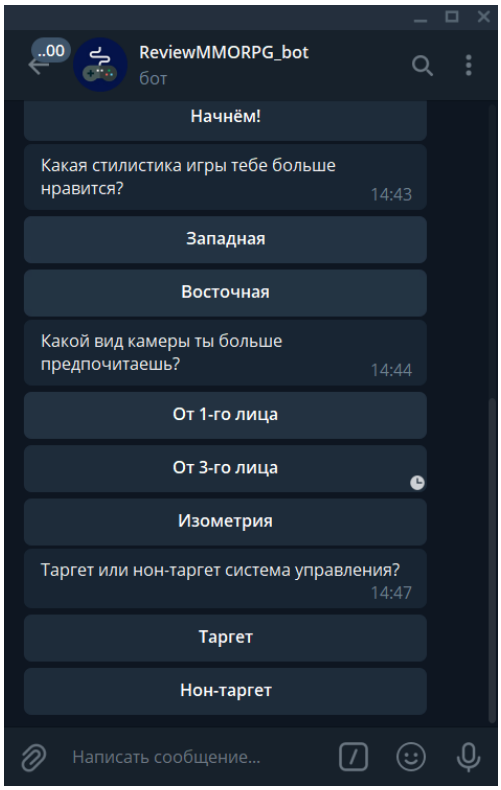


Рисунок 13 – Выбор системы управления

Если же выбирается вид камеры «От 1-го лица» данный выбор пропускается и переходит к следующему, поскольку игр от первого лица с таргетной системой управления не существует. Следующим выбором является – выбор основной игровой активности, а именно «ПВЕ» или «ПВП» (см. Рисунок 14).

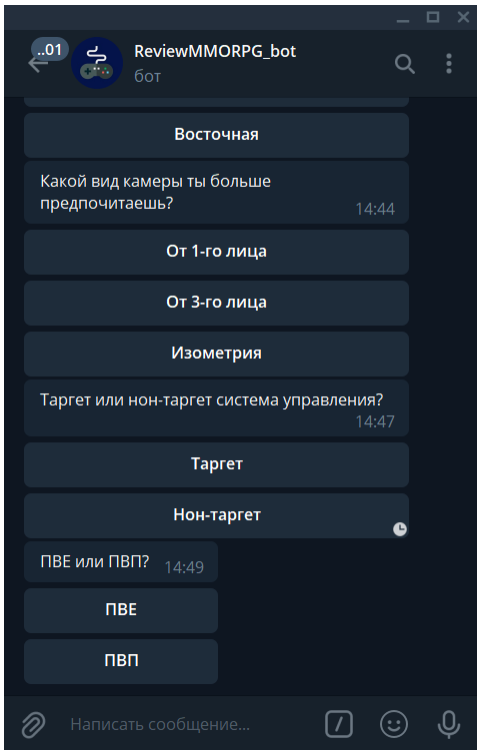


Рисунок 14 – Выбор направленности игровой активности

Далее в зависимости от выбора пользователя предоставляется список доступных ПВЕ или ПВП активностей (см. Рисунок 15).

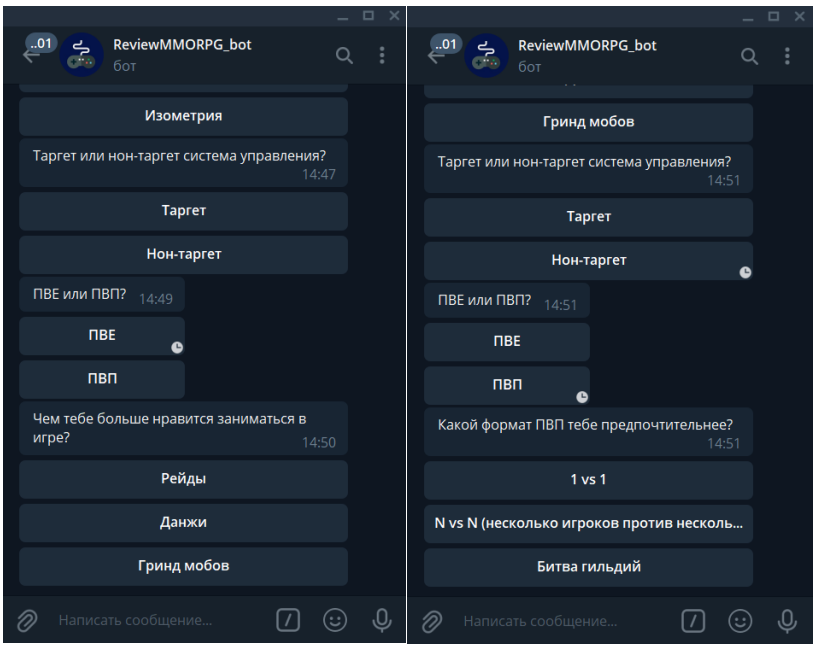


Рисунок 15 – Выбор вида игровой активности

Это финальный выбор и далее пользователю представляется список рекомендованных проектов, сопровождающийся ссылками на материал по данным играм на справочном сайте. Вне зависимости от того, будет ли пользователь

просматривать информацию на сайте, чат-бот уже выполнил свою главную функцию – помог принять решение в выборе проекта. Данная выборка охватывает самые важные аспекты игр данного жанра, а потому успешность подбора игры, которая с большей долей вероятности понравится пользователю, весьма велика. Выглядит конечный результат следующим образом (см. Рисунок 16).

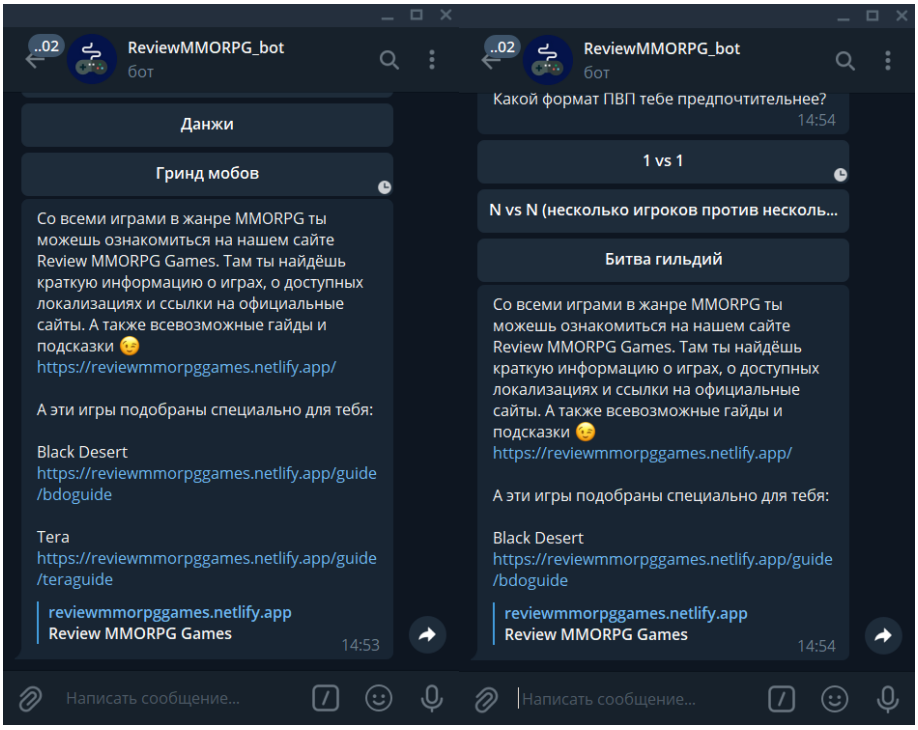


Рисунок 16 – Результат работы чат-бота

Каждый список рекомендованных игр выводится благодаря следующему коду:

```
if call.data == "etC2ntpve_grind": #От 3-го лица - Нон-Таргет - ПВЕ - гринд - список
  bot.send_message(call.message.chat.id,
'Со всеми играми в жанре MMORPG ты можешь ознакомиться '
'на нашем сайте Review MMORPG Games. Там ты найдёшь краткую информацию '
'о играх, о доступных локализациях и ссылки на официальные сайты. '
'А также всевозможные гайды и подсказки 😉'
'https://reviewmmorpggames.netlify.app/ \n''\n'
'А эти игры подобраны специально для тебя: \n''\n'
'Black Desert\n'
'https://reviewmmorpggames.netlify.app/guide/bdoguide\n''\n'
'Tera\n' 'https://reviewmmorpggames.netlify.app/guide/teraguide\n')
```

В результате работы данного чат-бота даже если он был найден на просторах интернета, бот познакомит и направит пользователя на справочный сайт, выдаст список игр и сопроводит их ссылками на справочный материал представленный непосредственно на самом сайте.

## 2.4 Тестирование

Для проверки конкретной работы созданного чат-бота были проведены тесты на трёх различных устройствах.

В первую очередь тестирование проводилось на смартфоне, поскольку мессенджер Telegram в первую очередь ориентирован именно на данный вид устройств. Основные характеристики устройства:

- операционная система – Android 5.1;
- версия Telegram – 7.7.2.

Второй тест проводился на стационарном компьютере. Доступ к чат-боту осуществлялся с помощью десктопной версии Telegram. Основные характеристики устройства:

- операционная система – Windows 7 Максимальная;
- версия Telegram – 2.7.4.

Третье тестирование проводилось на ультрабуке. Доступ к чат-боту также осуществлялся с помощью десктопной версии Telegram. Основные характеристики устройства:

- операционная система – Windows 10 Домашняя;
- версия Telegram – 2.7.4.

На всех трёх устройства тестирование прошло успешно. Чат-бот работал корректно. Все реализованные функции работали без ошибок. Предоставляемая

информация выводилась так, как задумано, а ссылки на внешний сайт были представлены в виде гипертекста. Исходя из тестов можно быть уверенным в корректной работе созданного чат-бота на устройствах, поддерживающих версию для мобильных устройств от 7.7.2 и выше, а для десктопной версии от 2.7.4 и выше.

## 2.5 Размещение чат-бота на Heroku

Для обеспечения бесперебойной работы чат-бота необходимо разместить его на сервере. Самой простой в использовании является облачная платформа Heroku, отличительной особенностью которой также является наличие бесплатного тарифа.

Перед тем как начать размещать бота на Heroku необходимо установить расширение для командной строки Windows – «Heroku CLI». Данное расширение позволит удобно работать с сервисом непосредственно со своего компьютера без особых трудностей.

После установки необходимо подготовить ряд файлов, которые необходимы для корректного развёртывания чат-бота на сервере. В первую очередь необходимо создать файл «requirements.txt». В нём прописываются дополнительно установленные модули и их версии, в данном случае библиотека «pyTelegramBotAPI». Далее создаётся файл «Procfile». Он не имеет расширения и содержит инструкции для Heroku необходимые для запуска приложения. Также, если необходимо, создаётся файл «runtime.txt» в котором указывается конкретная версия языка, в данном случае python, используемая при создании чат-бота.

После подготовки необходимо загрузить все файлы в открытый репозиторий на GitHub. Для того, чтобы третьи лица не получили доступ к управлению ботом, уникальный токен чат-бота заменяется конфигурационной переменной и будет введён позже, на этапе размещения бота на сервере. Это очень важный момент поскольку очень важно обеспечить безопасную и стабильную работу бота.

Далее необходимо перейти в командную строку Windows и зайти в директорию приложения на локальном компьютере. Процесс размещения чат-бота на сервере начинается с вхождения в свой аккаунт на Heroku. Делается это с помощью команды «heroku login» (см. Рисунок 17).

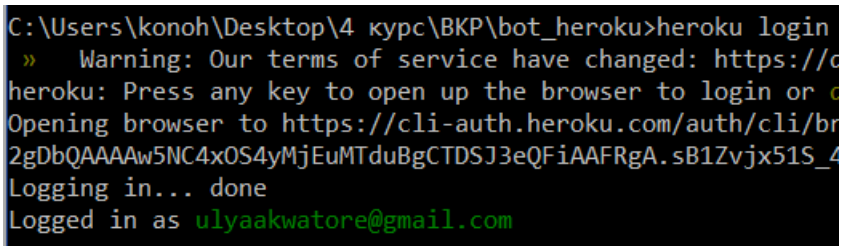


Рисунок 17 – Авторизация на Heroku

Далее необходимо создать приложение на heroku – «heroku create –region eu reviewmmorpg-bot». Это приложение – основа для бота (см. Рисунок 18).

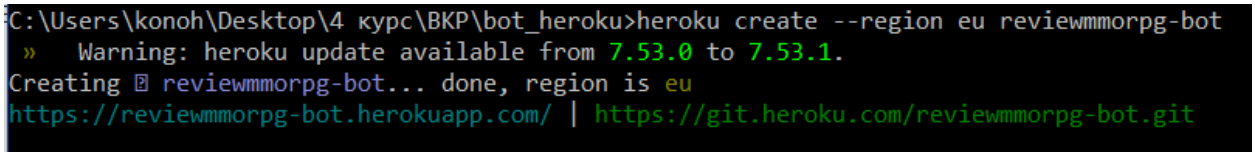


Рисунок 18 – Создание приложения на Heroku

Следующий этап – установка пакета сборки. Поскольку необходимо установить пакет python, вводится команда «heroku buildpacks:set heroku/python» (см. Рисунок 19).

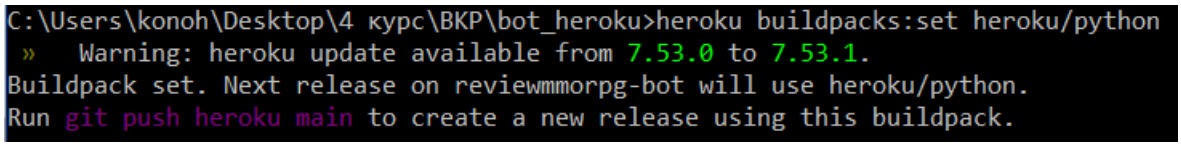


Рисунок 19 – Установка билдпакета

Сам heroku подсказывает следующий этап – необходимо разместить все созданные файлы на сервер. Выполняется это с помощью команды «git push heroku master» (см. Рисунок 20).

```
C:\Users\konoh\Desktop\4 курс\ВКР\bot_heroku>git push heroku master
Enumerating objects: 23, done.
Counting objects: 100% (23/23), done.
Delta compression using up to 4 threads
Compressing objects: 100% (18/18), done.
Writing objects: 100% (23/23), 8.06 KiB | 1.01 MiB/s, done.
Total 23 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Compressing source files... done.
remote: Building source:
remote:
remote: -----> Building on the Heroku-20 stack
remote: -----> Using buildpack: heroku/python
remote: -----> Python app detected
remote: -----> Compressing...
remote:        Done: 52M
remote: -----> Launching...
remote:        Released v3
remote:        https://reviewmmorpg-bot.herokuapp.com/ deployed to Heroku
remote:
remote: Verifying deploy... done.
To https://git.heroku.com/reviewmmorpg-bot.git
 * [new branch]      master -> master
```

Рисунок 20 – Размещение чат-бота на Heroku

На этом развёртывание закончено, но перед запуском чат-бота необходимо ввести ранее скрытый уникальный токен. Для этого в командную строку вводится команда «heroku config:set TELEGRAM_TOKEN=<токен>» (см. Рисунок 21).

```
C:\Users\konoh\Desktop\4 курс\ВКР\bot_heroku>heroku config:set TELEGRAM_TOKEN=
 »   Warning: heroku update available from 7.53.0 to 7.53.1.
Setting TELEGRAM_TOKEN and restarting ⬚ reviewmmorpg-bot... done, v3
TELEGRAM_TOKEN:
```

Рисунок 21 – Ввод уникального токена чат-бота

Запуск чат-бот осуществляется с помощью команды «heroku ps:scale bot=1» (см. Рисунок 22).

```
C:\Users\konoh\Desktop\4 курс\ВКР\bot_heroku>heroku ps:scale bot=1
 »   Warning: heroku update available from 7.53.0 to 7.53.1.
Scaling dynos... done, now running bot at 1:Free
```

Рисунок 22 – Запуск чат-бота

Для того, чтобы проверить корректно ли работает созданный чат-бот можно посмотреть логи приложения, для этого вводится команда «heroku logs --tail» (см. Рисунок 23).

```
2021-05-16T23:06:50.000000+00:00 app[api]: Build started by user ulyaakwatore@gmail.com
2021-05-16T23:07:14.073993+00:00 app[api]: Deploy d0e48a55 by user ulyaakwatore@gmail.com
2021-05-16T23:07:14.073993+00:00 app[api]: Release v7 created by user ulyaakwatore@gmail.com
2021-05-16T23:07:16.896531+00:00 heroku[bot.1]: State changed from crashed to starting
2021-05-16T23:07:19.796045+00:00 heroku[bot.1]: Starting process with command `python3 main.py`
2021-05-16T23:07:20.511508+00:00 heroku[bot.1]: State changed from starting to up
2021-05-16T23:07:23.000000+00:00 app[api]: Build succeeded
```

Рисунок 23 – Просмотр логов на Heroku

При необходимости можно остановить работу бота: «heroku ps:stop bot» (см. Рисунок 24).

```
Stopping bot dynos on ⬢ reviewmmorpg-bot... done

C:\Users\konoh\Desktop\4 курс\ВКР\bot_heroku>heroku ps:stop bot
```

Рисунок 24 – Остановка работы чат-бота

### 2.6 Интеграция чат-бота в сайт

Поскольку данный чат-бот разрабатывался для конкретного справочного сайта по онлайн играм, то необходимо перенаправлять аудиторию сайта на него. Для этого на главной странице сайта был размещён интерактивный баннер с рекламой чат-бота. При нажатии на баннер происходит переход по ссылке на чат-бота в Telegram. Для удобства пользователей ссылка открывается в новом окне. Также при наведении курсора на баннер, пользователь сразу видит мгновенный отклик в виде появляющейся надписи «Скорее нажимай!». Это сделано для того, чтобы предрасположить пользователя и сделать реакцию на его действия более «живой». Реализовано это следующим образом (см. Рисунок 25).


Рисунок 25 – Рекламный баннер чат-бота

Для интеграции в сайт использовался следующий код:

```
<a class="py-2" href=https://t.me/ReviewMMORPG bot target="_blank">
     <img src="img/img_5.png" alt="chat-bot" width ="570px"
     onmouseout="this.src='img/img_5.png'"
     onmouseover="this.src='img/img_6.png'">
</a>
```

На главной странице сайта конечный результат представлен на Рисунке 26.

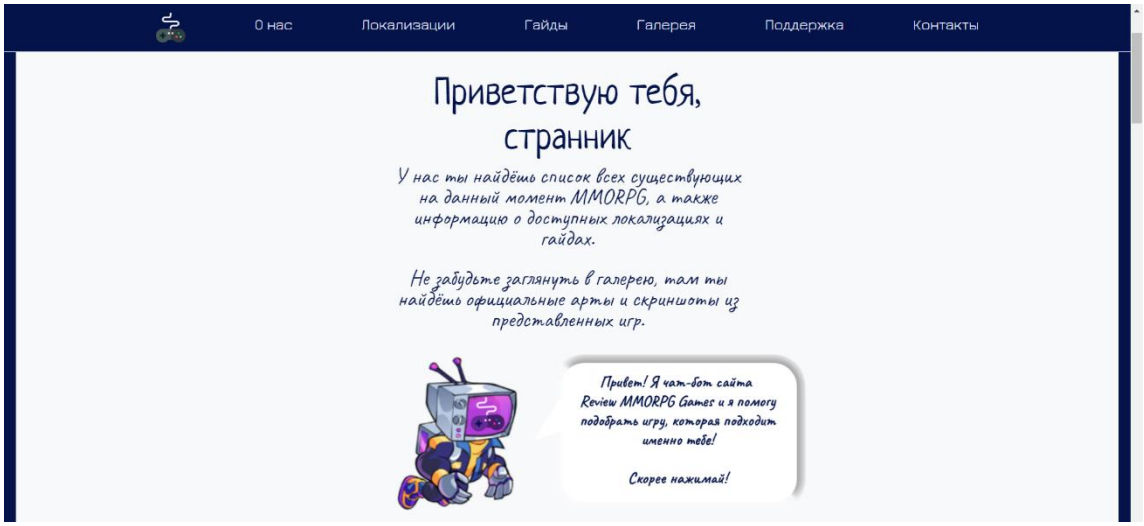


Рисунок 26 – Главная страница справочного сайта по онлайн-играм

# ВЫВОДЫ ПО ГЛАВЕ 2

Во второй главе был разработан чат-бот для справочного сайта по онлайн играм.

В первом параграфе главы автором ВКР была разработана структура чат-бота. За основу было взято дерево принятия решений. Были проанализированы важные аспекты предметной области и на основании этой информации построено дерево.

Во втором параграфе был создан бот на платформе Telegram. Также был получен необходимый для разработки уникальный токен и проведена первоначальная настройка бота, предоставляемая внутренним функционалом платформы.

В третьем параграфе был разработан чат-бот с использованием возможностей библиотеки pyTelegramBotAPI. Взаимодействие между пользователем и ботом реализовано с помощью inline-клавиатуры. Также разработана общая структура для выдержки кода в едином стиле.

В четвёртом параграфе было проведено тестирование работоспособности чат-бота на трёх устройствах с разными характеристиками. Тесты успешно подтвердили корректную работу бота.

В пятом параграфе была проведена работа по размещению чат-бота на сервере для обеспечения бесперебойной работы. Предварительно были созданы необходимые файлы и загружены в открытый репозиторий на GitHub. Далее произведено размещение чат-бота на облачной платформе Heroku. Для обеспечения безопасности уникальный токен бота был предварительно скрыт и введён непосредственно при развёртывании.

В шестом параграфе была произведена интеграция чат-бота в сайт. Для этого был создан рекламный баннер-ссылка и размещён на главной странице справочного сайта по онлайн играм.

В результате проделанной работы был разработан чат-бот для справочного сайта по онлайн-играм, размещённый на сервере.

# ЗАКЛЮЧЕНИЕ

В рамках данной выпускной квалификационной работы было проведено исследование в области искусственного интеллекта и методов его реализации в виде чат-бота.

В качестве реализации структуры разрабатываемого чат-бота было выбрано дерево принятия решений.

В ходе написания выпускной квалификационной работы был разработан чат-бот для справочного сайта по онлайн играм на базе мессенджера Telegram, который помогает пользователям принимать решение при выборе игрового проекта основываясь на индивидуальных предпочтениях и стиле игры.

Для обеспечения бесперебойной работы чат-бот был размещён на сервере, в качестве которого была выбрана облачная платформа Heroku.

В завершении проделанной работы была произведена интеграция чат-бот в справочный сайт по онлайн-играм в виде интерактивного рекламного баннера.

В ходе работы над приложением получен опыт создания собственного проекта с прохождением всего жизненного цикла продукта. Получен опыт работы в роли аналитика, проектировщика и разработчика. Помимо этого, приобретён опыт работы с облачной платформой Heroku.

Готовый продукт был размещён в публичном репозитории на веб-сервисе GitHub по адресу https://github.com/Akwatore/Telegram_bot_heroku.

Благодаря однотипной структуре кода чат-бота в будущем возможно производить актуализацию данных, в том числе рекомендованных списков игр, без особых усилий.

# СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ


1. Toshiba. 8 историй, повлиявших на развитие искусственного интеллекта/ Toshiba // Хабр : Сообщество IT-специалистов: [сайт], 2018. – URL: https://habr.com/ru/company/toshibarus/blog/424007/ (дата обращения: 14.03.2021).
2. Джанарсанам С. Разработка чат-ботов и разговорных интерфейсов. - М.: ДМК Пресс, 2019.
3. Искусственный интеллект // Tadviser : Государство. Бизнес. IT : [сайт], 2019. – URL: https://www.tadviser.ru/index.php/Продукт:Искусственный_интеллект_(ИИ,_Artificial_intelligence,_AI) (дата обращения: 14.03.2021).
4. Искусственный интеллект // Википедия : Свободная энциклопедия : [сайт], 2021. – URL: https://ru.wikipedia.org/wiki/Искусственный_интеллект (дата обращения: 14.03.2021).
5. Искусственный интеллект, чат-боты и виртуальные агенты: угроза человечеству и контакт-центрам? // Call Center Guru : Интернет-портал. Сообщество профессионалов : [сайт], 2017. – URL: https://callcenterguru.ru/articles/iskusstvennyy-intellekt-chat-boty-i-virtual-nyye-agenty-ugroza-chelovechestvu-i-kontakt-tsentram (дата обращения: 14.03.2021).
6. Машинное обучение // Calltouch : Сервис сквозной аналитики, коллтрекинга и управления рекламой : [сайт]. – URL: https://www.calltouch.ru/glossary/mashinnoe-obuchenie/ (дата обращения: 14.03.2021).
7. Обработка естественного языка // neerc.ifmo.ru: Сайт вики-конспектов кафедры компьютерных технологий Университета ИТМО: [сайт], 2021. – URL: http://neerc.ifmo.ru/wiki/index.php?title=Обработка_естественного_языка (дата обращения: 14.03.2021).

8. Тематический обзор. Мессенджеры и конструкторы чат-ботов // ResearchGate : [сайт], 2019. – URL: https://www.researchgate.net/publication/330523883_Messengers_and_Chat-bot_Design_Tools (дата обращения: 25.03.21).
9. Шахиди А. Деревья решений: общие принципы/ Шахиди А.// Loginoma : Аналитическая платформа : [сайт], 2019. – URL: https://loginom.ru/blog/decision-tree-p1 (дата обращения: 14.03.2021).
10. Яковлев К. Перспективы интеллектуальных агентов/ Яковлев К.// ПостНаука : Проект о современной фундаментальной науке и ученых, которые ее создают : [сайт], 2018. – URL: https://postnauka.ru/faq/83426 (дата обращения: 14.03.2021).
11. Edward Feigenbaum. The Age of Intelligent Machines: Knowledge Processing--From File Servers to Knowledge Servers/ Edward Feigenbaum // KurzweilAI.net : [сайт], 1990. – URL: https://kurzweilai-brain.gothdyke.mom/articles/art0098.html (дата обращения: 14.03.2021).
12. pyTelegramBotAPI. — Текст : электронный // GitHub : [сайт]. — URL: https://github.com/eternnoir/pyTelegramBotAPI#message-handlers (дата обращения: 25.04.2021).