|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ \_\_\_***Робототехника и комплексная автоматизация***\_\_\_\_\_\_

КАФЕДРА \_***Системы автоматизированного проектирования (РК-6)***\_

**ОТЧЕТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ**

Студент\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*фамилия, имя, отчество*

Группа\_\_***РК6-82Б***\_\_\_

Тип практики \_\_\_\_\_\_\_\_\_\_\_***Преддипломная***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Название предприятия**:\_\_\_\_ООО “АКСТИМ”\_\_\_**

Студент **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Редколис Р.Р.**

*подпись, дата фамилия, и.о.*

Руководитель практики

от кафедры **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_ Витюков Ф. А.**

*подпись, дата фамилия, и.о.*

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

***2024 г.***

**«Московский государственный технический университет имени Н.Э. Баумана**

**(национальный исследовательский университет)»**

**(МГТУ им. Н.Э. Баумана)**

УТВЕРЖДАЮ

Заведующий кафедрой *РК6*

\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_ *А.П. Карпенко* \_

« \_\_\_ » \_\_\_\_\_\_\_\_\_\_\_\_ 2024 г.

**ЗАДАНИЕ**

**на прохождение производственной практики**

**\_\_\_\_\_\_*Преддипломная*\_\_\_\_\_\_**

Тип практики

Студент

Редколис Руслан Русланович \_\_***4***\_\_ курса группы \_***РК6-82Б***\_

Фамилия Имя Отчество № курса индекс группы

в период с \_***13 мая 2024***\_\_\_ г. по ***26 мая 2024*** г.

*Предприятие:* \_\_\_ ***ООО ”АКСТИМ”***

*Подразделение:* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(отдел/сектор/цех)

*Руководитель практики от предприятия (наставник):*

***Красильникова Евгения Алексеевич,***

***Руководитель направления администрирования персонала ООО ”АКСТИМ”***

(Фамилия Имя Отчество полностью, должность)

*Руководитель практики от кафедры:*

**Витюков Фёдор Андреевич, Старший преподаватель**

(Фамилия Имя Отчество полностью, должность)

*Задание:*

**1.** *Реализовать API запросы разными способами*

**2. Провести анализ полученных результатов**

Дата выдачи задания ***14 мая 2024***г.

Руководитель практики от предприятия  **\_\_\_\_\_\_\_/*Красильникова Е.М.*\_\_/**

Руководитель практики от кафедры  **\_\_\_\_\_\_\_/Витюков Ф.А.\_\_\_\_\_\_/**

Студент **\_\_\_\_\_\_\_/Редколис Р.Р\_\_\_\_\_\_\_\_/**

**СОДЕРЖАНИЕ**

[**ВВЕДЕНИЕ** 4](#_Toc131284932)

[**1.** **ПОСТАНОВКА ЗАДАЧИ** 4](#_Toc131284933)

[**2.** **КРАТКИЙ ОТЧЕТ О ВЫПОЛНЕННЫХ РАБОТАХ** 4](#_Toc131284934)

[**3.** **АНАЛИЗ РЕЗУЛЬТАТОВ** 4](#_Toc131284935)

[**ЗАКЛЮЧЕНИЕ** 5](#_Toc131284936)

[**СПИСОК ЛИТЕРАТУРЫ** 5](#_Toc131284937)

**ВВЕДЕНИЕ**

Современная разработка интерактивных приложений и игр требует интеграции различных внешних сервисов и API для расширения функциональности и улучшения пользовательского опыта. Unreal Engine, как одна из ведущих платформ для создания игр и интерактивного контента, предоставляет мощные инструменты для реализации таких интеграций. В данной работе рассматривается гибридный подход к вызову API-запросов в Unreal Engine, используя как язык программирования C++, так и визуальное скриптование с помощью Blueprints. Этот подход позволяет эффективно обрабатывать сложные сетевые взаимодействия и данные, обеспечивая при этом гибкость разработки и удобство использования. Целью данной работы является демонстрация возможностей и преимуществ комбинированного использования C++ и Blueprints для интеграции внешних сервисов в проекты, созданные на Unreal Engine.

2. **ПОСТАНОВКА ЗАДАЧИ**

В рамках практической работы необходимо реализовать вызов API-запроса как на языке C++, так и с использованием Blueprints в Unreal Engine. Это позволит продемонстрировать возможности гибридного подхода к разработке, где сложные сетевые взаимодействия и обработка данных выполняются на уровне низкоуровневого программирования с использованием C++, а также через визуальное программирование с помощью Blueprints. Такой подход должен обеспечить максимальную гибкость и эффективность при интеграции внешних сервисов и API в игровое или интерактивное приложение, созданное на Unreal Engine.

1. **КРАТКИЙ ОТЧЕТ О ВЫПОЛНЕННЫХ РАБОТАХ**

API-сервисы предоставляют возможность взаимодействия с облачными ресурсами и сервисами через программные интерфейсы. Эти сервисы позволяют разработчикам создавать, развивать и масштабировать свои приложения без необходимости управления физической инфраструктурой. Они предоставляют широкий спектр функций, включая хранение данных, вычислительные мощности, искусственный интеллект, аналитику, машинное обучение и многое другое. Облачные API-сервисы позволяют разработчикам создавать инновационные приложения и сервисы, используя гибкость и масштабируемость облачных ресурсов, что способствует повышению производительности, снижению затрат и ускорению развертывания новых продуктов на рынке.

Разработка API запросов играет ключевую роль в усовершенствовании функциональности виртуального помощника, позволяя ему взаимодействовать с пользователем более естественным и эффективным способом. В данной части работы исследуются современные технологии и применения к ним API, такие как ChatGPT и elevenlabs, которые предоставляют широкий спектр инструментов для создания более реалистичного и интерактивного взаимодействия с виртуальным помощником. Использование этих API открывает новые возможности для расширения функциональности виртуального помощника и повышения его эффективности. В рамках этой части работы уже реализованы функции распознавания речи с последующим переводом в текст (speech-to-text) будет использованы в качестве запроса, результат запроса будет воспроизводится виртуальным голосом (text-to-speech), это значительно улучшает взаимодействие пользователя с виртуальным помощником. Таким образом, разработка API запросов играет важную роль в развитии и совершенствовании функциональности виртуального помощника, открывая перед ним новые перспективы для более эффективного и удобного использования.

В рамках дипломной работы было реализовано взаимодействие с внешними API в Unreal Engine с использованием запросов на языке C++. Результаты этих запросов были успешно интегрированы и воспроизведены в проекте, что позволило расширить функционал виртуального помощника.

3.1 Разработка API запроса в ChatGPT

API-запросы к ChatGPT пишутся в виде HTTP-запросов с использованием методов POST и GET. В запросе указывается URL endpoint API, а также параметры, необходимые для выполнения определенной функции, такие как текст для генерации ответа или параметры модели.

POST: Этот метод используется для отправки данных на сервер для обработки. В отличие от GET, данные в запросе передаются в теле запроса, что делает его более безопасным для передачи больших объемов данных или конфиденциальной информации. POST-запросы часто используются для создания или обновления данных на сервере, простой API запрос представлен на рисунке .

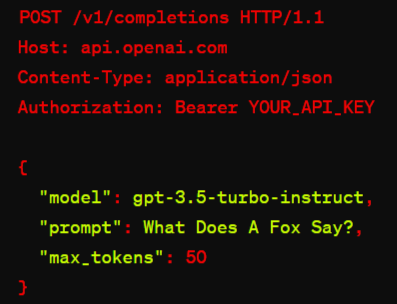


Рисунок 28 пример POST-запроса

Этот пример представляет собой POST-запрос к API OpenAI для получения завершения текста с использованием конкретной модели " gpt-3.5-turbo-instruct". В запросе указывается хост (api.openai.com), тип содержимого (application/json) и авторизационный токен с помощью которого открывается доступ к модели. В теле запроса передаются параметры, такие как выбранная модель, начальная фраза (prompt) и максимальное количество токенов, которые должны быть сгенерированы в ответе.

POST-запрос выполняется в методе Activate(). В этом методе сначала заполняется персональный ключ сервиса, который был указан в ноде blueprint.

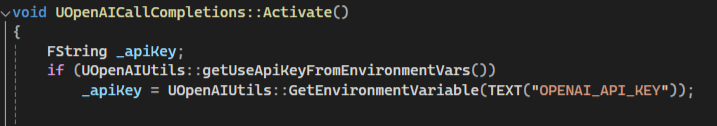


Рисунок 29 установка API ключа

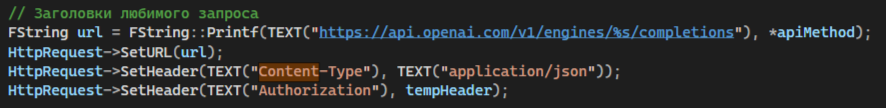
Затем формируется объект HTTP-запроса и устанавливаются заголовки, такие как Content-Type и Authorization, включая API-ключ в заголовке Authorization. 

Рисунок 30 установка заголовков запроса

Затем создается JSON-объект, который содержит параметры запроса, такие как текстовый запрос (prompt), максимальное количество токенов (max\_tokens), температура (temperature) и другие.

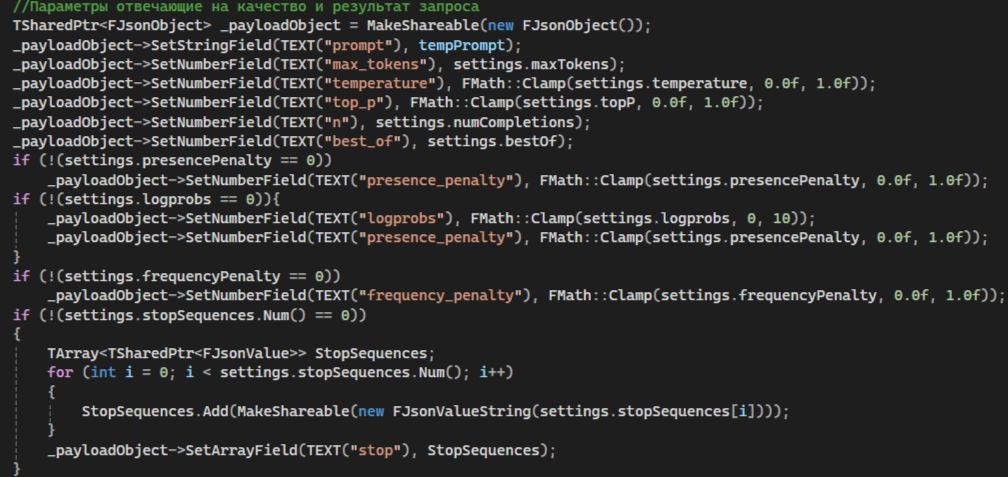


Рисунок 31 параметры модели

Этот JSON-объект сериализуется в строку и устанавливается как содержимое (Content) HTTP-запроса. Затем запрос отправляется на сервер API OpenAI с помощью метода ProcessRequest().



Рисунок 32 NNNNNN

GET: Этот метод используется для запроса данных от сервера. В запросе параметры передаются в URL в виде строки запроса. Этот метод обычно используется для получения данных и не должен изменять состояние сервера.

Мы получаем ответ с сервера, используя объект FHttpResponsePtr Response, который представляет HTTP-ответ от сервера. Для получения текстового содержимого ответа, которое содержит данные в формате JSON

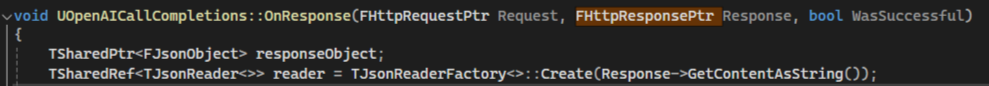


Рисунок 33 NNNN

Далее вызывается метод GetContentAsString() на объекте Response. В результате получаем JSON файл. Данный JSON содержит не только ответ на запрос но иную информацию, десериализация JSON файла представлена на рисунке NNN.

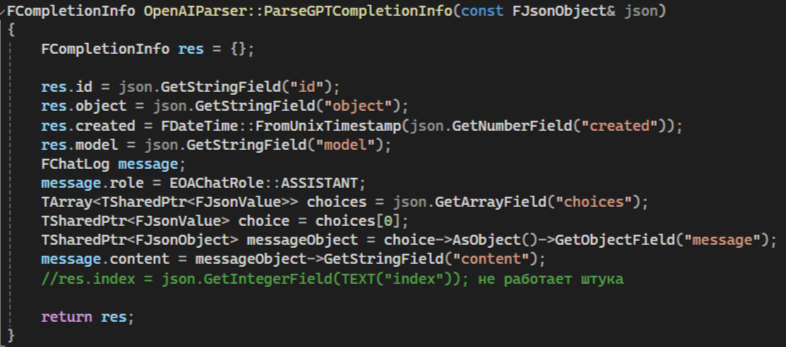


Рисунок 34 десериализация JSON файла

Из responseObject извлекается массив, который находится в поле Choices. Из данного поля извлекаются массивы с содержащие требуемый ответ на запрос, которые добавляется в переменную массива объекта FCompletions.

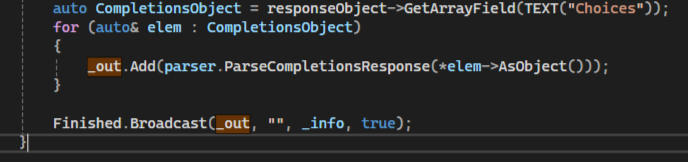


Рисунок 35 Запись ответа в массив

На рисунке NNN представлено извлечение текста из переменной массива FString.

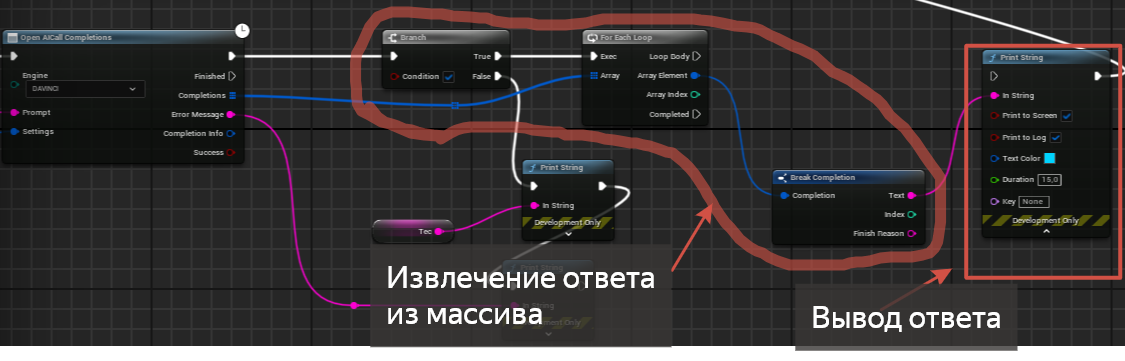


Рисунок 36 Вывод response сервера

В дальнейшем, полученный ответ будет использован в качестве запроса для получения аудиофайла, который будет воспроизводиться голосовым помощником. Этот подход позволит обеспечить более интерактивное и удобное взаимодействие пользователя с системой, предоставляя ему возможность получать необходимую информацию в аудиоформате.

* 1. Разработка API запроса в ElevenLabs

В рамках дальнейшего развития проекта планируется использовать полученные текстовые ответы для создания запросов на получение аудиофайлов через сервис ElevenLabs, которые будут воспроизводиться голосовым помощником, что существенно повысит удобство и эффективность взаимодействия пользователя с системой. Несмотря на то, что у нас имеется возможность использовать функцию преобразования текста в речь (TTS) от MetaHuman SDK, данный сервис не поддерживает необходимый язык и производит более роботизированное звучание. В отличие от этого, сервис ElevenLabs предоставляет высококачественные, естественно звучащие аудиофайлы, что делает его более подходящим для реализации нашего проекта. В этом разделе будет представлена логика, реализованная с использованием blueprint, описывающая процесс отправки текста на сервис ElevenLabs, получения аудиофайла и его воспроизведения, что позволит значительно улучшить пользовательский опыт.

В предшествующем разделе была разработана функциональность для выполнения API-запросов на языке C++. Однако, следует отметить, что Varest предлагает аналогичный функционал. Этот плагин также обеспечивает обработку ответов от серверов, включая чтение JSON-данных и других форматов ответов.

VaRest - это плагин, который предоставляет разработчикам удобные средства для работы с веб-запросами и взаимодействия с внешними веб-сервисами прямо из проектов, созданных на Unreal Engine. Этот плагин обеспечивает возможность отправки HTTP-запросов (GET, POST, PUT, DELETE и других) и обработки ответов от серверов, что позволяет интегрировать ваши игры и приложения с различными сетевыми сервисами и API.

Для выполнения преобразования текста в речь с использованием API сервиса ElevenLabs используется HTTP POST-запрос, включающий различные параметры настройки процесса синтеза речи. Структура запроса представлена на рисунке NNN



Рисунок 37 Post запрос в сервис

Исходя из заголовка можно заметить, что тело запроса содержит данные в формате JSON. Тело запроса представляет собой JSON-объект следующего вида:

"text": Текст, который необходимо преобразовать в речь.

"model\_id": Идентификатор модели синтеза речи, которая будет использоваться.

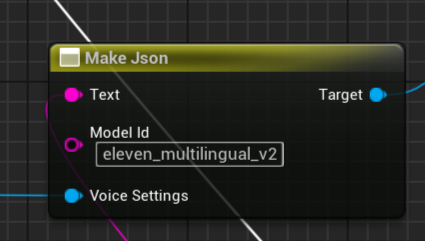


Рисунок 38 версия модели и текста

"voice\_settings": Объект с настройками голоса (см Рисунок NNN) указывается 2 параметра:

"stability": Параметр стабильности голоса.

"similarity\_boost": Параметр усиления сходства.

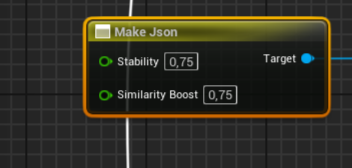


Рисунок 39 параметры голоса модели

Остальные параметры устанавливаются в самом сервисе мануально, но по желанию их также можно передать в POST-запрос.

"style": Параметр стиля речи.

"use\_speaker\_boost": Логическое значение, указывающее на необходимость использования усиления громкости.

"pronunciation\_dictionary\_locators": Массив объектов, указывающих на словари произношения:

"pronunciation\_dictionary\_id": Идентификатор словаря произношения.

"version\_id": Идентификатор версии словаря.

"seed": Значение случайного числа для генерации уникального синтеза речи.

"previous\_text": Предыдущий текст, используемый для контекстуального синтеза речи.

"next\_text": Следующий текст, используемый для контекстуального синтеза речи.

"previous\_request\_ids": Массив идентификаторов предыдущих запросов для сохранения контекста:

"string": Идентификатор предыдущего запроса.

"next\_request\_ids": Массив идентификаторов следующих запросов для сохранения контекста:

"string": Идентификатор следующего запроса.

Итоговый POS

Этот запрос позволяет гибко настроить параметры синтеза речи, включая настройки голоса, словари произношения, а также контекстуальные параметры, такие как предыдущий и следующий текст. Возможность использования настроек голоса и контекстуальных параметров позволяет достичь высокой точности и естественности синтезируемой речи, что является ключевым фактором для создания качественных голосовых аудиозаписи для виртуального помощника. На рисунке NNN показано, как данный API POST запрос можно реализовать с использованием Blueprints в Unreal Engine.

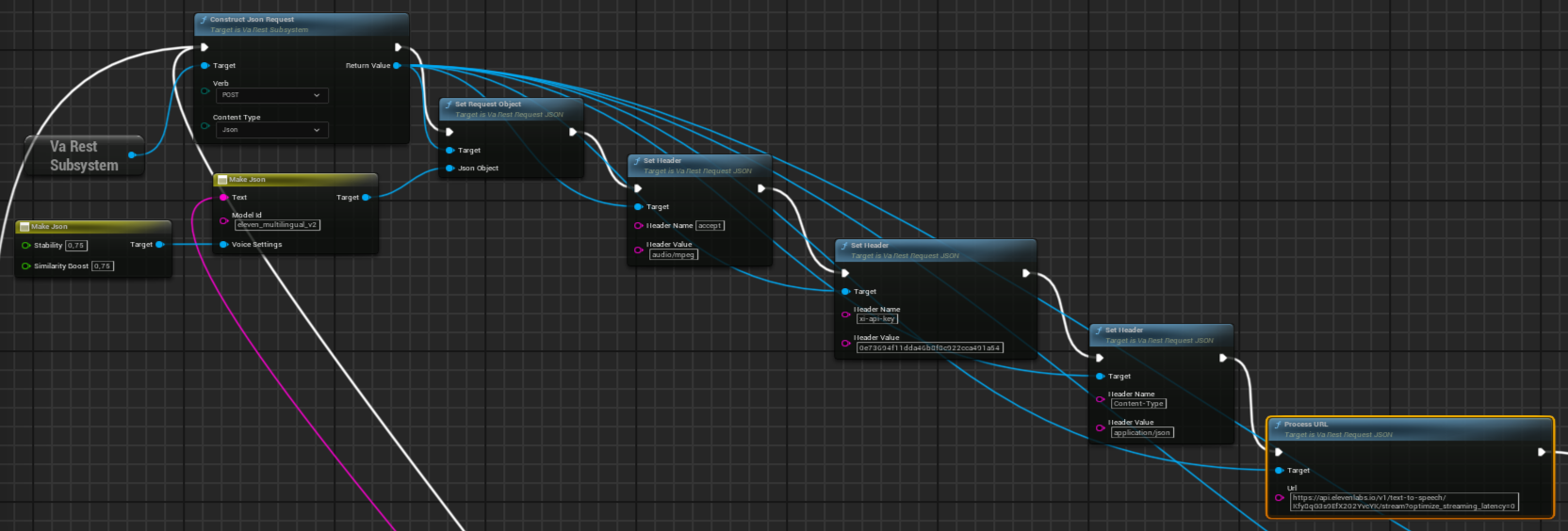


Рисунок 40 Логика POST-запроса в сервис

В дальнейшем на представленной диаграмме Blueprints (Рисунок NNN) выполняется процесс получения аудиозаписи в ответ на API-запрос, проверки содержимого ответа и последующего импорта аудиоданных с использованием плагина Runtime Audio Importer. После получения содержимого ответа в виде строки, проверяется наличие определенной подстроки. В зависимости от результата, создается объект Runtime Audio Importer, который используется для импорта аудиоданных из буфера, передаваемых в формате MP3.

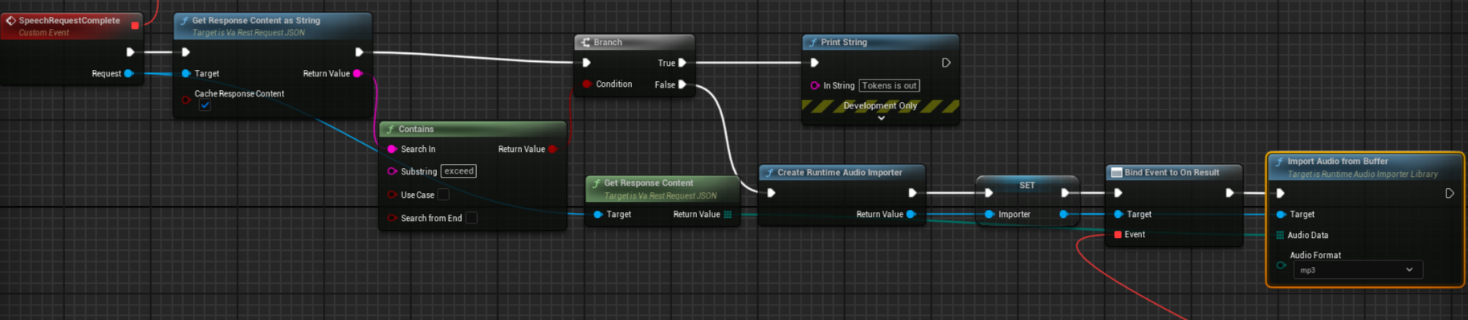


Рисунок 41 Логика обработки Response

**ЗАКЛЮЧЕНИЕ**

В результате выполнения поставленных задач был успешно реализован вызов API-запросов с использованием как C++, так и Blueprints в Unreal Engine. Для демонстрации были использованы следующие подходы:

ChatGPT API:

C++: Написан код на C++ для отправки запроса к API ChatGPT и получения текстового ответа. Используя HTTP-библиотеки, был реализован процесс создания и отправки POST-запроса, обработки JSON-ответа и извлечения текстовой информации. Это позволяет интегрировать возможности генерации текста в игровое или интерактивное приложение, улучшая взаимодействие с пользователем.

Blueprints: Созданы Blueprints для отправки запроса к ChatGPT API и обработки ответа. Визуальные узлы Blueprints позволяют легко настроить параметры запроса, отправить его и получить ответ, который затем может быть использован в различных игровых сценариях, таких как диалоговые системы или интерактивные подсказки.

Аудиозапись и Runtime Audio Importer:

Blueprints: Реализован процесс получения и обработки аудиозаписи с использованием Blueprints и плагина Runtime Audio Importer. После получения аудиофайла по API-запросу, он импортируется в Unreal Engine. В процессе импорта используется плагин Runtime Audio Importer, который позволяет динамически загружать аудиофайлы во время выполнения приложения. Эти аудиофайлы затем используются для озвучки голосового помощника, обеспечивая более интерактивное и реалистичное взаимодействие с пользователем.

Таким образом, оба подхода – C++ и Blueprints – продемонстрировали свою эффективность и гибкость при интеграции внешних сервисов и обработке данных в Unreal Engine, что значительно расширяет возможности создания интерактивных и динамичных приложений.

**СПИСОК ЛИТЕРАТУРЫ**

1. MetaHuman – Unreal Engine // MetaHuman URL:  
   https://www.unrealengine/com/en-US/metahuman/ Дата обращения:  
   18.03.2024.
2. GitHub – Runtime Speech Recognizer // Wiki. URL: https://github.com/gtreshchev/RuntimeSpeechRecognizer/wiki/ Дата обращения: 21.03.2024.
3. Youtube – MetaHuman SDK // MetaHuman SDK URL:  
   https://www.youtube.com/@metahumansdk Дата обращения: 19.03.2024.
4. Level of Detail (computer graphics) // Wikipedia, the free encyclopedia URL: https://en.wikipedia.org/wiki/Level\_of\_detail\_(computer\_graphics). Дата обращения: 22.03.2024.
5. Creating and Using LODs // Unreal Engine Documentation URL: https://docs.unrealengine.com/5.3/en-US/WorkingWithContent/Types/StaticMeshes /HowTo/LODs/. Дата обращения: 22.03.2024.
6. Божко А.Н., Жук Д.М., Маничев В.Б. Компьютерная графика. [Электронный ресурс] // Учебное пособие для вузов. − М.: Изд-во МГТУ им. Н. Э. Баумана, 2007. - 389 с., - ISBN 978-5-7038-3015-4, Режим доступа: http://ebooks.bmstu.ru/catalog/55/book1141.html. Дата обращения: 10.02.2024.