Создание робота копирайтера с использованием ChatGPT

Содержание

1. Базовые понятия

В данном разделе определяется ряд базовых понятий, которые будут использоваться в дальнейшем (рисунок 1.1):

- *Рабочая область* – центральное окно в программе PIX Studio после создания проекта
- *Окно активностей* – левое окно в программе PIX Studio после создания проекта
- *Активности* – базовые функции, из которых конструируется робот
- *Группа активностей* – активности, объединённые единой идей своего функционала. В дальнейшем для ясности активности будут именоваться следующим образом «Группа активностей/Название активности»
- *Использовать/применить активность* – перетащить активность из окна активностей в рабочую область. Для выполнения данного действия требуется нажать ЛКМ на активность и, зажав ЛКМ, перенести курсор в рабочую область, а затем отпустить ЛКМ

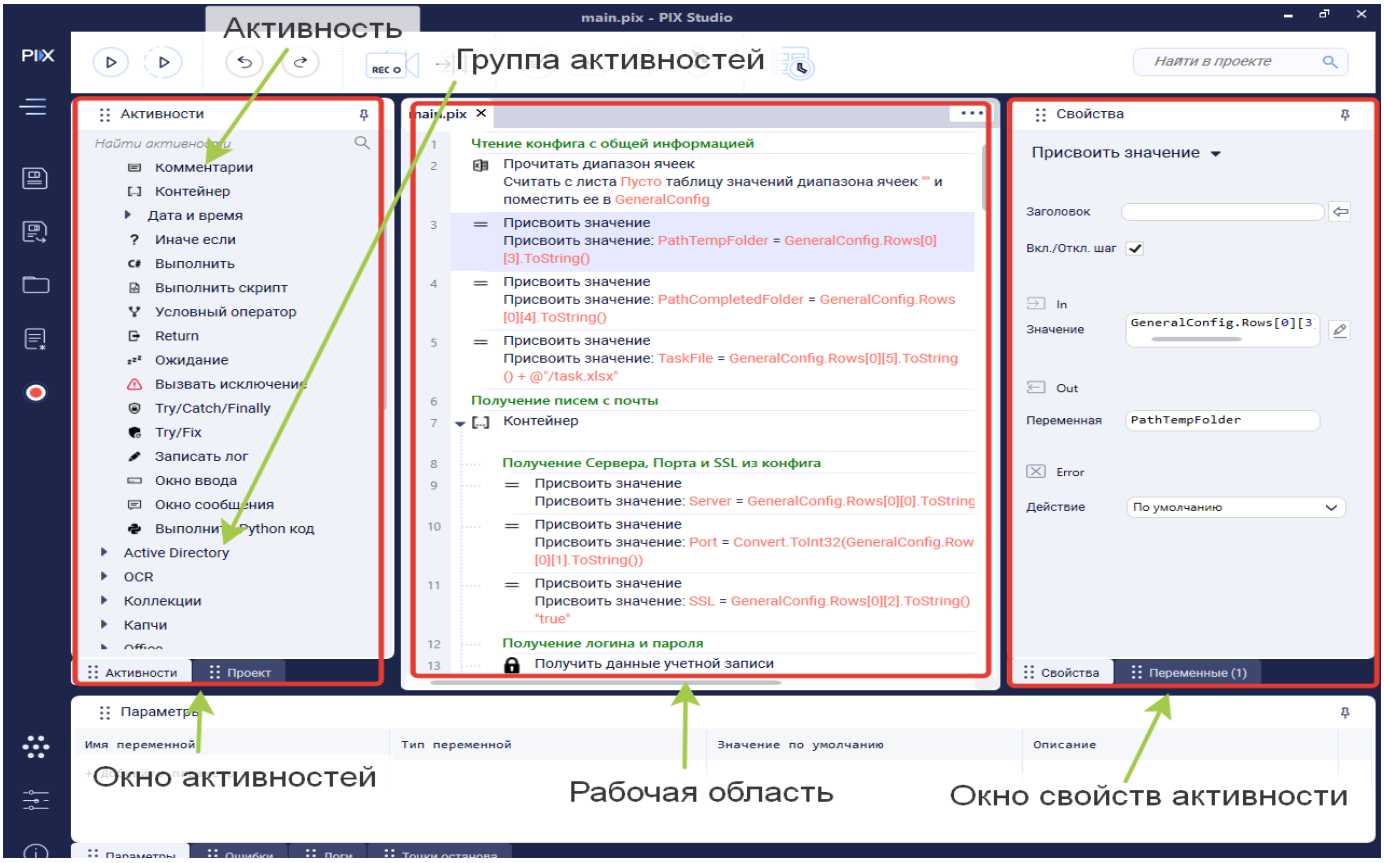


Рисунок 1.1 Окно редактора PIX Studio

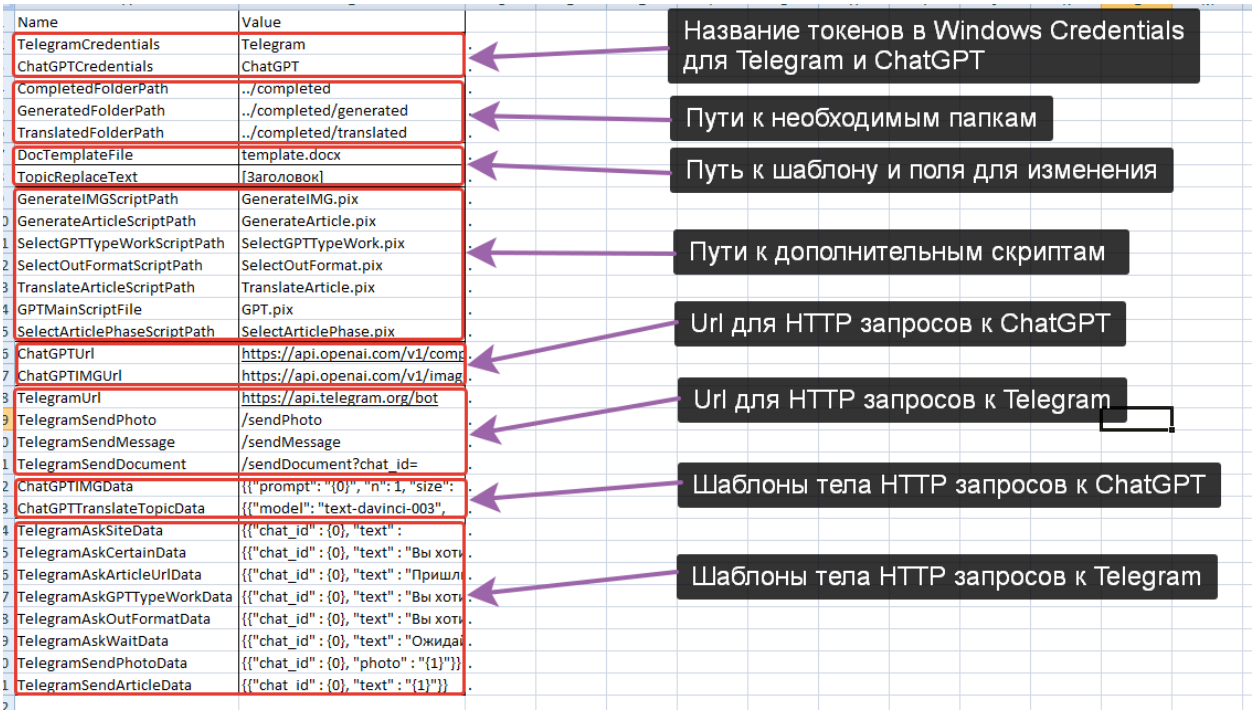
2. Постановка задачи

Робот является телеграмм ботом, которой просит выбрать нужную статью с предложенных сайтов, а затем отправляет статью в ChatGPT с указанием на перевод или перефразирование. Затем поступает указание на генерацию картинки, после чего робот отправляет полученное изображение и текст в чат к пользователю.

3. Предварительные действия

В рамках данной задачи создаётся два робота. Первый является телеграмм ботом, второй обрабатывает полученные данные и отправляет их по почте. Однако в первую очередь, создаётся дополнительный excel документ, который в дальнейшем будет именоваться, как «*конфиг*». В данные документ заносится информация, которая необходима для работы робота, но может быть изменена сторонними действиями. Для данной задачи конфиг может содержать следующую информацию (Рисунок 3.1):

- Относительные пути к необходимым папкам и файлам (путь к папке с необработанными файлами, путь к папке с результатом обработки и т.д.)
- Url и тела HTTP запросов
- Данные для получения токенов из Windows Credentials



3.1. Пример конфига для робота.

А также на отдельной странице заполняются данные для обработки того или иного сайта(Рисунок 3.2).

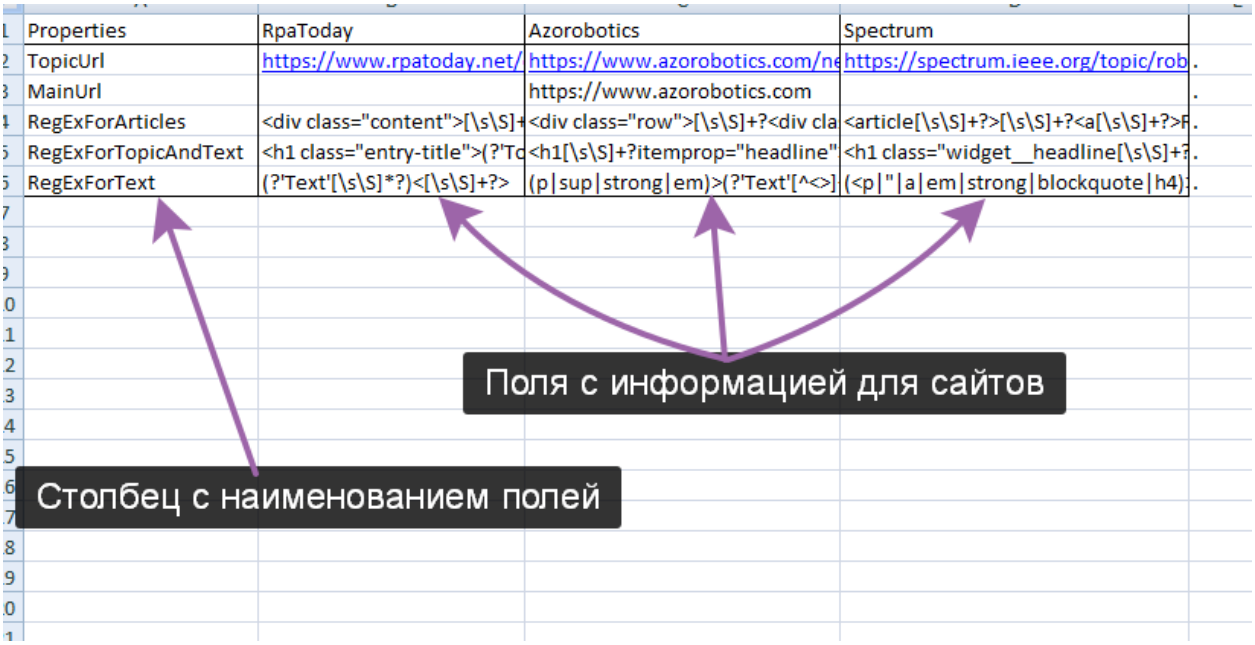


Рисунок 3.2. Пример данных для записи результата

Помимо конфига необходимо убедится в наличие шаблона docx документа для записи полученного текста.

Для написания регулярных выражений можно использовать сайт «regex101.com» или какой-нибудь подобный.

Так же в Windows Credentials нужно токен подключения к боту, токен подключения к ChatGPT. И создать профиль самого бота используя бот @BotFather (https://t.me/BotFather).

4. Работа с PIX Studio

Теперь необходимо запустить PIX Studio и создать два новых проект. Для этого сначала необходимо нажать на панели слева на иконку блокнота со звёздочкой в левом нижнем углу. В появившемся окне выбирается пункт проект (Рисунок 4.1). PIX Studio попросит ввести название проекта и его расположение. Также для одного из проектов необходимо создать 6 дополнительных скриптов. Делается это аналогичным образом, но в окне, появляющемся после нажатия на иконку блокнота выбирается пункт скрипт.

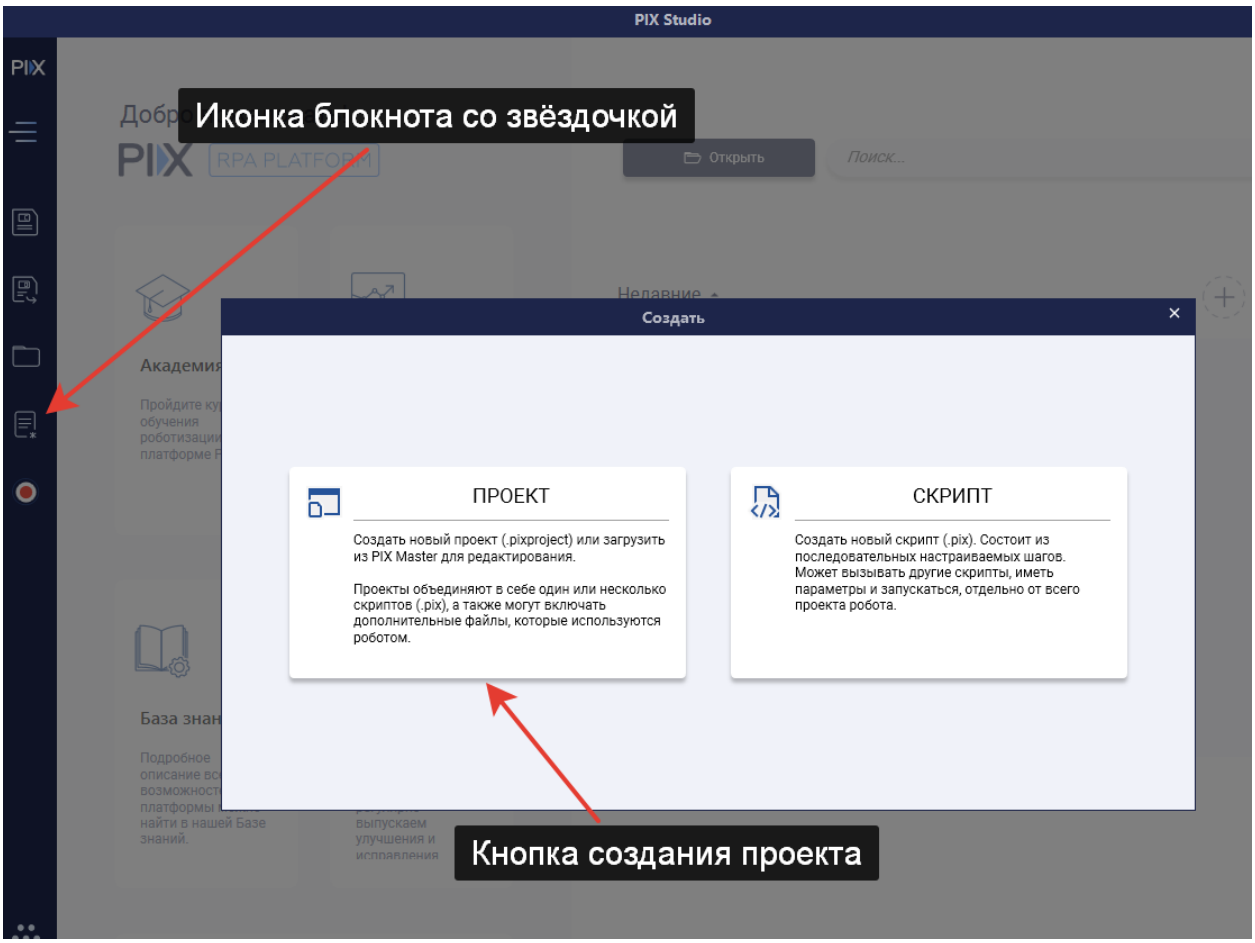


Рисунок 4.1. Создание проекта

5. Проект Telegram бота

В данном проекте присутствует 8 скриптов:

- *TelegramBot.pix* – Отвечает за запуск бота и начальную работу
- *SelectArticlePhase.pix* – Выбор сколько статей обрабатывать
- *SelectGPTTypeWork.pix* – Выбор перевести статью или сгенерировать новую.
- *SelectOutFormat.pix* – Выбор как вывести результат.
- *GPT.pix* –Получение текста и заголовка статьи.
- *GenerateArticle.pix* – Генерация новой статьи.
- *TranslateArticle.*pix – Перевод статьи.
- *GenerateIMG.*pix – Генерация изображения к статье.

5.1. Скрипт TelegramBot.pix

В первую очередь с помощью активности *«Базовые/Присвоить значения»* и C# функции *new* инициализируются важные переменные. (Рисунок 5.1.1)

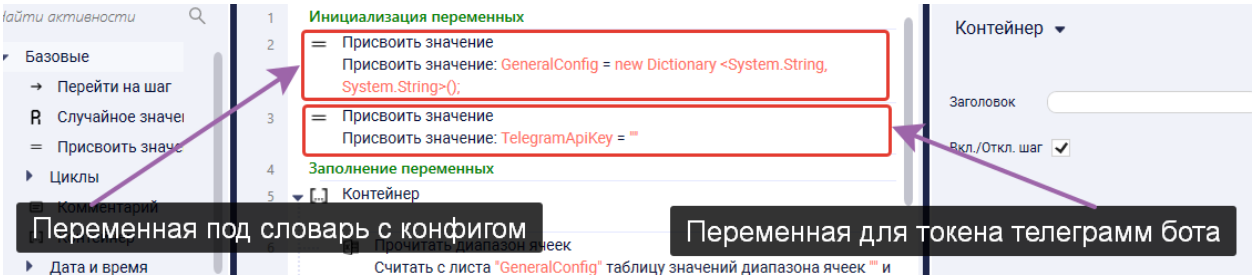


Рисунок 5.1.1. Инициализация переменных

Затем с помощью активности *«Office/Excel/Прочитать диапазон ячеек»* считывается содержимое конфига и заносится в таблицу. В свойствах активности заполняется: *Лист* – название листа с конфигом, *Диапазон* – указывается «""» для прочтения всего диапазона, *С заголовками* – ставится галочка в случае использования строки заголовков в конфиге, *Путь к файлу* – путь до файла с конфигом, *Таблица* – переменная с результирующей таблицей. Затем с помощью активности *«Базовые/Присвоить значение»* и C# функции *Таблица.AsEnumerable().ToDictionary<DataRow, Формат_ключа, Формат_значения>(row => row.Field<Формат_ключа>(Столбец_с_ключами), row => row.Field<Формат_значения>(Столбец_со_значениями))* таблица конфига переводится в словарь.

Следующим шагом с помощью активности *«Windows Credentials/Получить данные учетной записи»* считывается значение токена для бота Telegram. В свойствах заполняется: *Имя ресурса* – наименование под которым сохранено значение, *Логин* – переменная, в которую сохранится строка с токеном, если значение сохранялось не в защищённом виде, *Пароль* – переменная с защищённой строкой, если значение сохранялось в защищённом виде. (Рисунок 5.1.2)

Рисунок 5.1.2. Чтение конфига

Следующим шагом с помощью активности *«Мессенджеры/Telegram/ Создать соединение с Telegram»* создаётся подключение к телеграмм боту. В свойствах указывается: *Токен* – токен подключения к телеграмм боту, *Соединение* – переменная с соединением к телеграмму. После с помощью активности *«Мессенджеры/Telegram/Менеджер Telegram»* запускается бесконечный цикл считывания входящих сообщений. В свойствах заполняется: *Соединение* – переменная с соединением к телеграмму. (Рисунок 5.1.3)

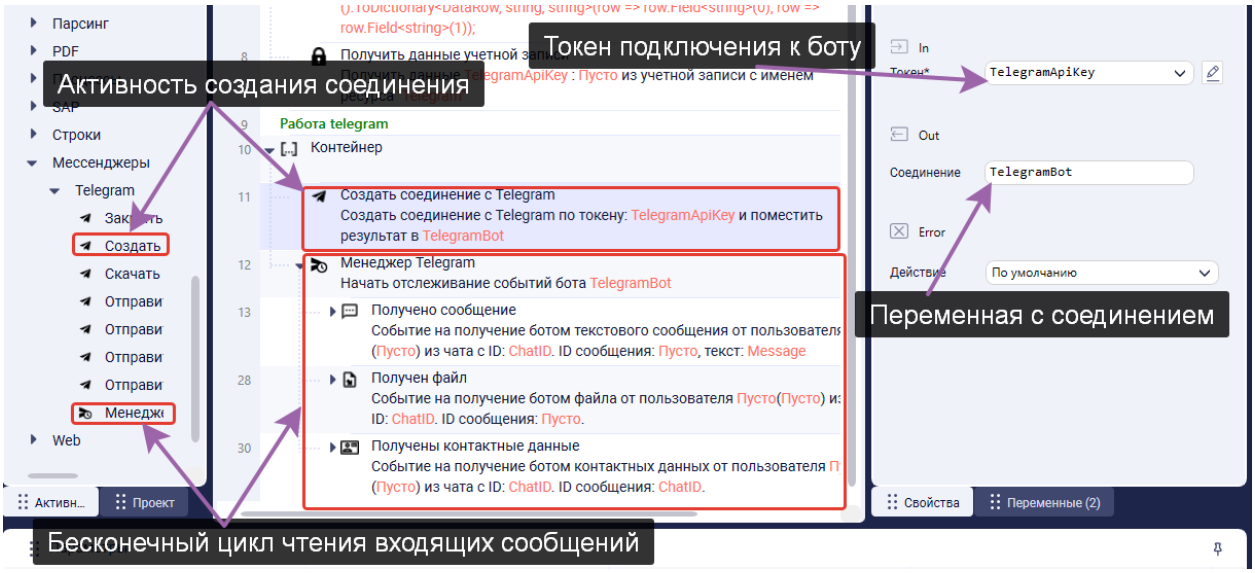


Рисунок 5.1.3. Подключение к телеграмм

В случае, если пришёл контакт другого пользователя или фото, то с помощью активности *«HTTP/Отправить HTTP запрос»* в чат отправляется сообщения с кнопками выбора сайта. В свойствах активности указывается: *Url* – Url отправки сообщения, *Метод* – метод HTTP запроса, в случае отправки равен Post, *Данные* – тело запроса, сообщение, команда вызова кнопок выбора ответа и ID чата, которое инициализируется в соответствующем разделе активности *«Мессенджеры/Telegram/Менеджер Telegram»*. (Рисунок 5.1.4)

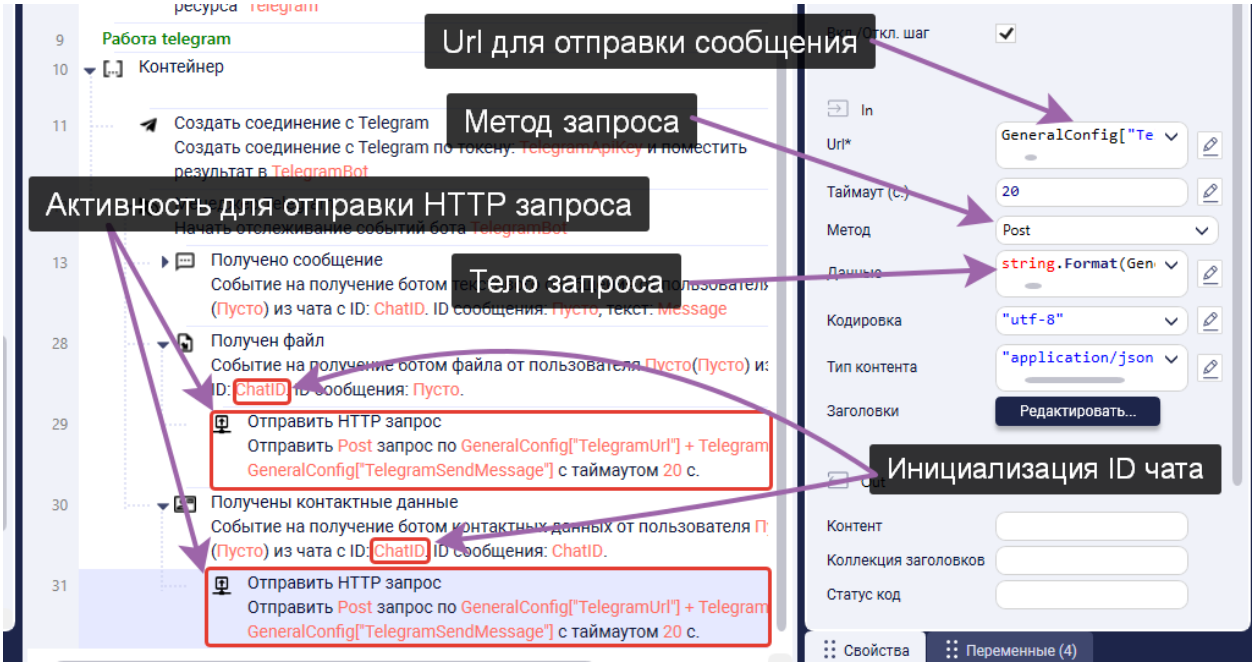


Рисунок 5.1.4. Отправка шаблона входящего сообщения

В случае, если входящее сообщение – это текст, то в соответствующем разделе активности *«Мессенджеры/Telegram/Менеджер Telegram»* он сохраняется в переменную, затем данная переменная с помощью активности *«Базовые/Условный оператор»* и C# функции *Строка.Equals(Строка_для_сравнения)* текст сообщения сравнивается на соответствие одной из команд. Если текст является командой выключения бота, то с помощью активности *«Мессенджеры/Telegram/Закрыть соединение Telegram»* обрывается соединение с телеграммом. Если текст совпадает с одним из названий сайта, то в зависимости от присутствия нужного ключа в словаре конфига с помощью активности

«Базовые/Выполнить» и C# функций *switch(Переменная){case Значение1: Команда1; case Значение2: Команда2; ...}*, *Словарь[Ключ] = Новое_значение* и *Словарь.Add(Ключ, Значение)* выбор пользователя фиксируется в словаре с конфигом. Для проверки наличия ключа используется C# функция *Словарь.ContainsKey(Ключ)*. После внесения нужного значение активностью *«Базовые/Выполнить скрипт»* вызывается скрипт выбора следующих данных. Если полученное сообщение не совпадает ни с одной командой, отправляется сообщение с выбором сайта. (Рисунок 5.1.5)

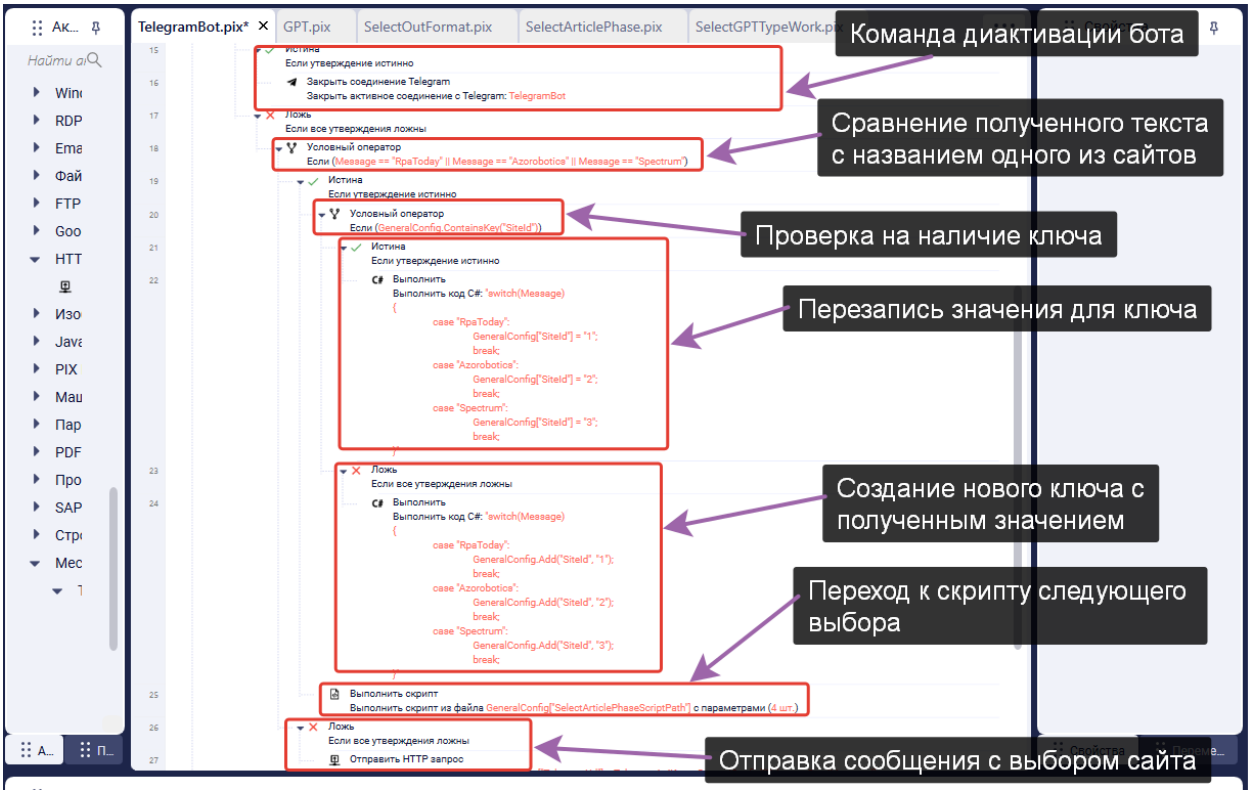


Рисунок 5.1.5. Проверка полученного текста.

В активности *«Базовые/Выполнить скрипт»* при вызове заполняется окно *Параметры запуска скрипта*. В случае вызова скриптов *SelectArticlePhase.pix, SelectGPTTypeWork.pix, SelectOutFormat.pix* Заполняются следующие параметры (Рисунок 5.1.6)

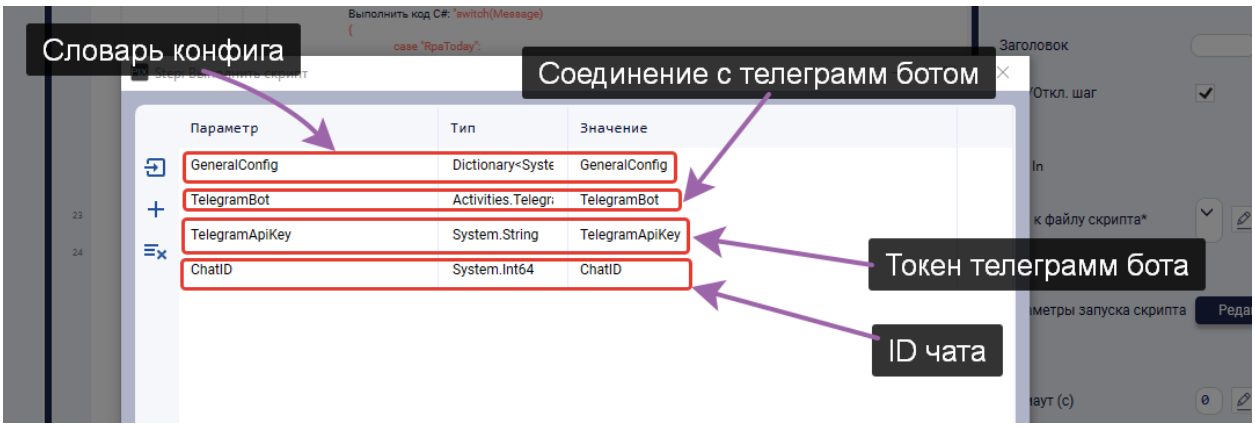


Рисунок 5.1.6. Вызов следующего скрипта

5.2. Скрипты SelectArticlePhase.pix, SelectGPTTypeWork.pix, SelectOutFormat.pix

Действие перечисленных выше скриптов сводится к отправке активностью «HTTP/Отправить HTTP запрос» выбора количества статьей для перевода, режима работы ChatGPT – перевод или генерация нового, выбор формата вывод полученных данных. А затем запуск бесконечного цикла приёма сообщений активностью *«Мессенджеры/Telegram/Менеджер Telegram»* аналогично рисункам 5.1.3, 5.1.4, 5.1.5.

Скрипт *SelectOutFormat.pix* вызывает скрипт GPT.pix аналогично рисунку 5.1.6.

5.3. Скрипт GPT.pix

В начале скрипта аналогично рисункам 5.1.1. и 5.1.2 создаются переменные для хранения токена для ChatGPT а также словаря для хранения конфига сайта.

Также активностями «Коллекции/Словарь/Создать словарь» и «Коллекции/Список/Создать список» создаются список и словарь для хранения списка ссылок на статьи и словаря с темами и текстами статей.

Следующим шагом с помощью активности *«Файлы/Путь существует?»* проверяется наличие необходимой папки. В свойствах указывается: *Путь* – путь к необходимой папке, хранится в словаре конфига, *Результат* – булева переменная, в которую будет записан результат проверки.

Затем с помощью активностей *«Базовые/Условный оператор»* и *«Файлы/Создать папку»* создаётся папка в случае её отсутствия.

Данные действия проводятся для всех необходимых папок. (Рисунок 5.3.1)

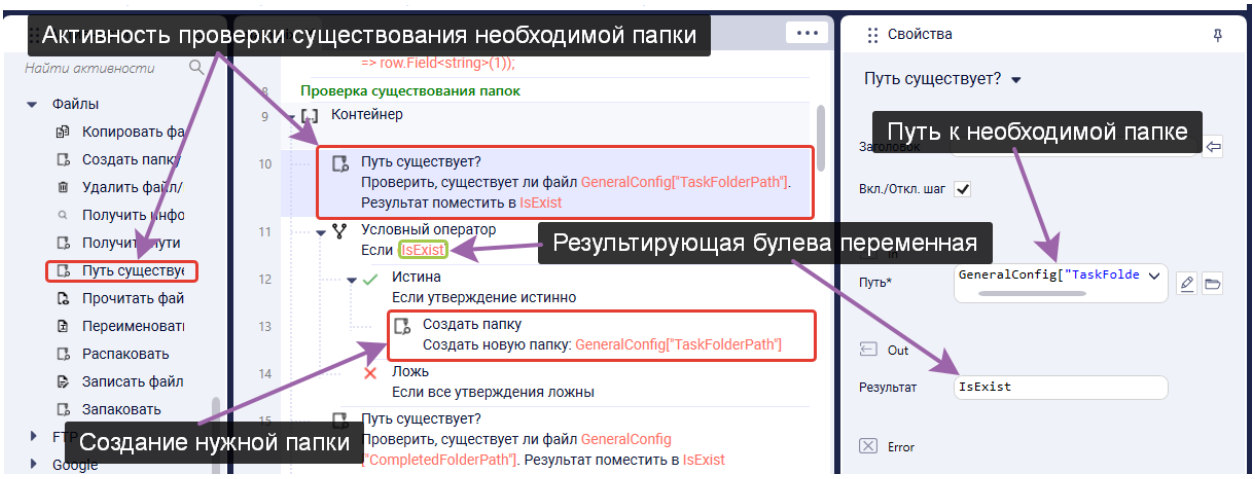


Рисунок 5.3.1. Создание необходимых папок

Затем с помощью активности *«Базовые/Условный оператор»* проверяется выбор пользователя – считать все статьи или одну. Если пользователь выбрал одну конкретную статью, то отправленная ссылка добавляется в список активностью *«Коллекции/Список/Поместить объект в список»*. Ели же выбраны все, то активностью *«HTTP/Отправить HTTP запрос»* запрашивается текст всей главной страницы сайта.

Затем с помощью активности *«Базовые/Присвоить значение»* и C# функции *System.Text.RegularExpression.Regex.Matches(Текст, Регулярное_выражение, Опции_регулярного_выражения)* из текста всей страницы достаются ссылки на все статьи.

Далее активностью *«Базовые/Циклы/Цикл для каждого»* создаются цикл по всем результатам регулярного выражения и активностью *«Коллекции/Список/Поместить объект в список»* с помощью C# функции *Результат_регулярного_выражения.Groups[Название_группы].Value* заносятся в список статей. (Рисунок 5.3.2)

Рисунок 5.3.2. Получение ссылок на статьи

Далее активностью *«Базовые/Цикл/Цикл для каждого»* создаётся цикл прохода по всем статьям.

В начале цикла активностью *«HTTP/Отправить HTTP запрос»* считывается весь текста страницы со статьёй. Затем к полученному тексту активностью *«Базовые/Присвоить значение»* и функцией *System.Text.RegularExpression.Regex.Match (Текст, Регулярное_выражение, Опции_регулярного_выражения)* применяется регулярное выражения для получения текста статьи и заголовка статьи.

Активностью *«Базовые/Присвоить значение»* и C# функцией *Результат_регулярного_выражения.Groups[Название_группы].Value* полученный заголовок и текст сохраняются в отдельные переменные.

Следующим шагом текст повторно обрабатывается регулярным выражения для убирания лишних символов. И склеивается с помощью активностей *«Базовые/Циклы/Цикл для каждого»* и *«Базовые/Присвоить значение»*.

После чего полученные результаты активностью *«Коллекции/Словарь/Задать значение для ключа»* сохраняются раннее созданный словарь, где значение – это текст статьи, а ключ – заголовок статьи. (Рисунок 5.3.3)

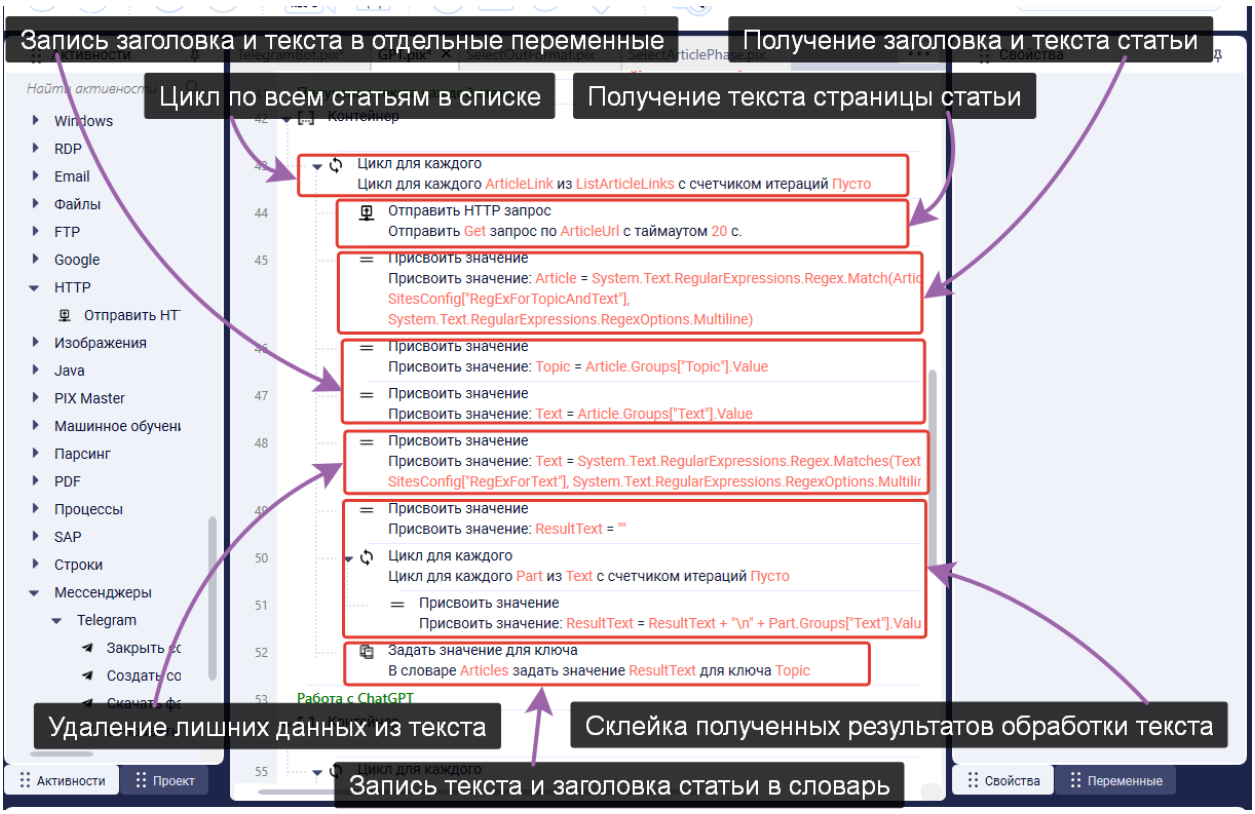


Рисунок 5.3.3. Получение заголовка и текста статьи

Далее активностью *«Базовые/Циклы/Цикл для каждого»* запускается цикл по всем заголовкам и текстам статьей.

В первую очередь активностью *«Базовые/Присвоить значение»* вычисляется и сохранение количества токенов в запросе на перевод названия статьи в запросе к ChatGPT.

Далее отправляется запрос к ChatGPT активностью *«HTTP/Отправить HTTP запрос»*. В свойствах активности указывается: *Url* – URL обращения к ChatGPT, *Метод* – метод HTTP запроса, в данном случае Post, *Данные* – тело запроса, в данному случае Json строка в свойствах которой указывается, *max_tokens*=4000 – Количество_в_запросе, *promt*=”Translate in Rusian” + Заголовок статьи, *Заголовоки* – добаляется заголовок с ключом «*Authorization*» и значение «*Bearer Токен_подключения_к_ChatGPT*», *Контент* – переменная, куда будет записан полученный текст, *Статус код* – переменная куда будет запись код статуса запроса.

Далее активностью «*Базовые/Условный оператор*» код статуса запроса сравнивается с кодом успешного выполнения. Если результат проверки

положительный, то полученный текст с помощью активности *«Парсинг/JSON»* преобразуется в JSON объект. А далее с помощью активности *«Базовые/Присвоить значение»* и C# функций *JSON_Объект.GetProperty(Название_свойства)* и *Строка.Replace* из JSON объекта достаётся текст с переведённым заголовком и удалёнными несколькими символами.

Затем в зависимости от выбранного режима работы запускается следующий этап активностями *«Базовые/Условный оператор»* и *«Базовые/Выполнить скрипт»*. (Рисунок 5.3.4)

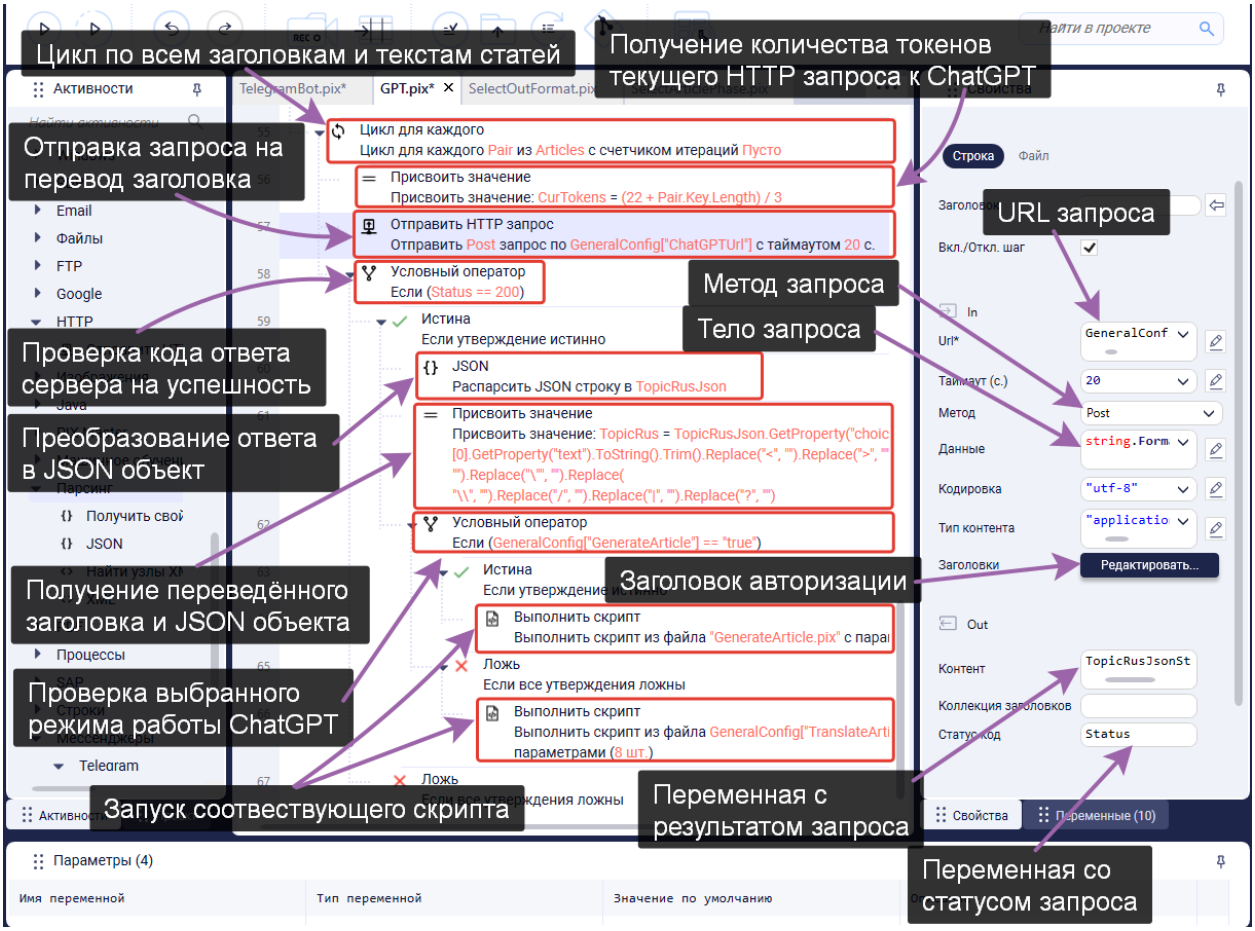


Рисунок 5.3.4. Перевод заголовка.

5.4. Скрипт GenerateArticle.pix

При вызове скрипта необходимо передать ряд параметров в окне *«Параметры запуска скрипта»*. (Рисунок 5.4.1)

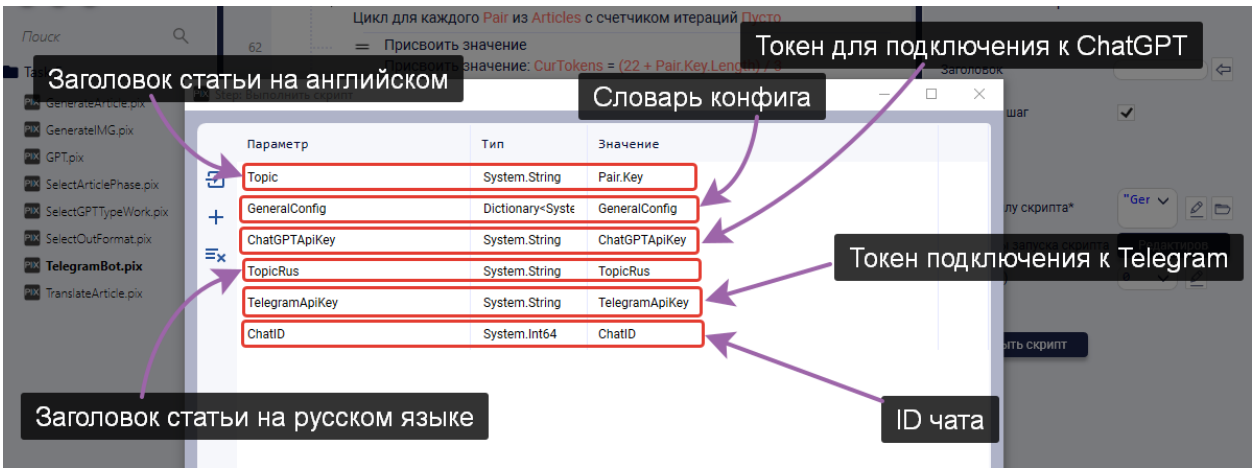


Рисунок 5.4.1. Передача параметров в скрипт

Скрипт аналогичен рисунку 5.3.4 с разницей, что HTTP запрос отправляется с просьбой сгенерировать статьи с темой текущей статьи.

После сохранения текста сгенерированной статьи в отдельную переменную активностью *«Файлы/Копировать файл/папку»* файл шаблона копируется в конечное место. Затем активностями *«Office/Word/Заменить текст в Word»* и *«Office/Word/Записать в файл Word»* записывается заголовок на русском языке и текст сгенерированной статьи на русском языке в результирующий файл.

После чего активностью *«Базовые/Выполнить скрипт»* вызывается скрипт *GenerateIMG.pix*. (Рисунок 5.4.2)

Рисунок 5.4.2. Сохранение данных в Word

5.5. Скрипт TranslateArticle.pix

При вызове скрипта необходимо передать ряд параметров в окне *«Параметры запуска скрипта»*. (Рисунок 5.5.1)

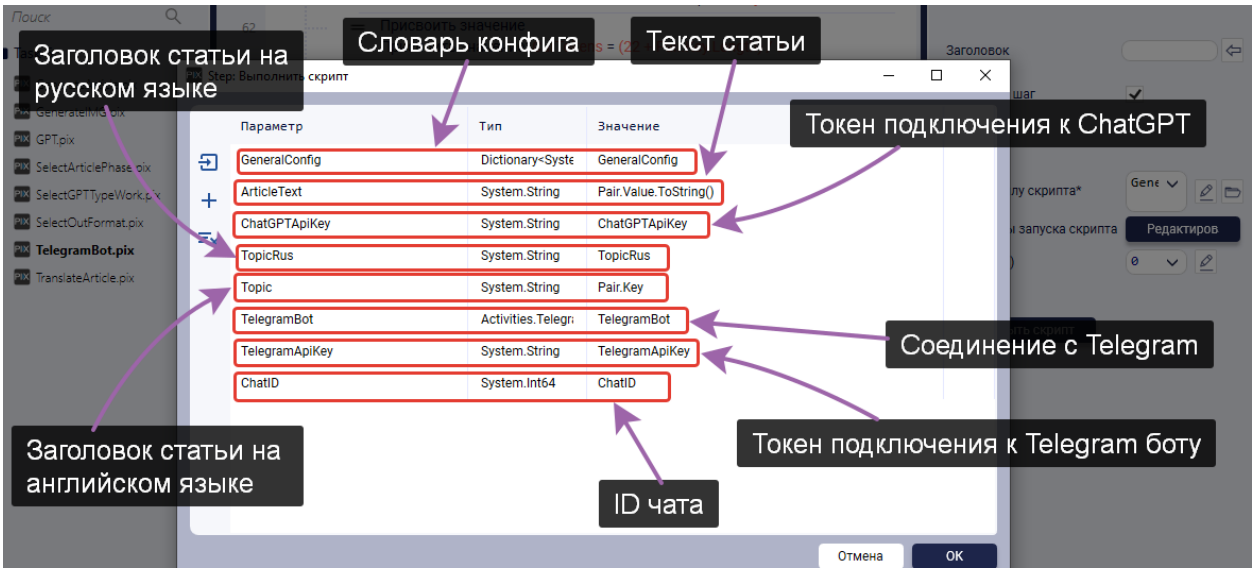


Рисунок 5.5.1. Передача параметров скрипта

В начале скрипта активностью «Коллекции/Список/Создать список» создаётся список для хранения кусков текста статьи. Далее активностью «Базовые/Циклы/Цикл с … по …» и C# функции Строка.Length запускается цикл кускам текста длиной 500 символов.

Активностью «Базовые/Условный оператор» итерация цикла проверяется на завершающую. В обоих случаях для записи куска текста в список используется активность «Коллекции/Список/Поместить объект в список» и в зависимости номера итерации C# функция Строка.Substring(Начало_подстроки, Количество_символов) и Строка.Substring(Начало_подстроки), если итерация завершающая. (Рисунок 5.5.2)

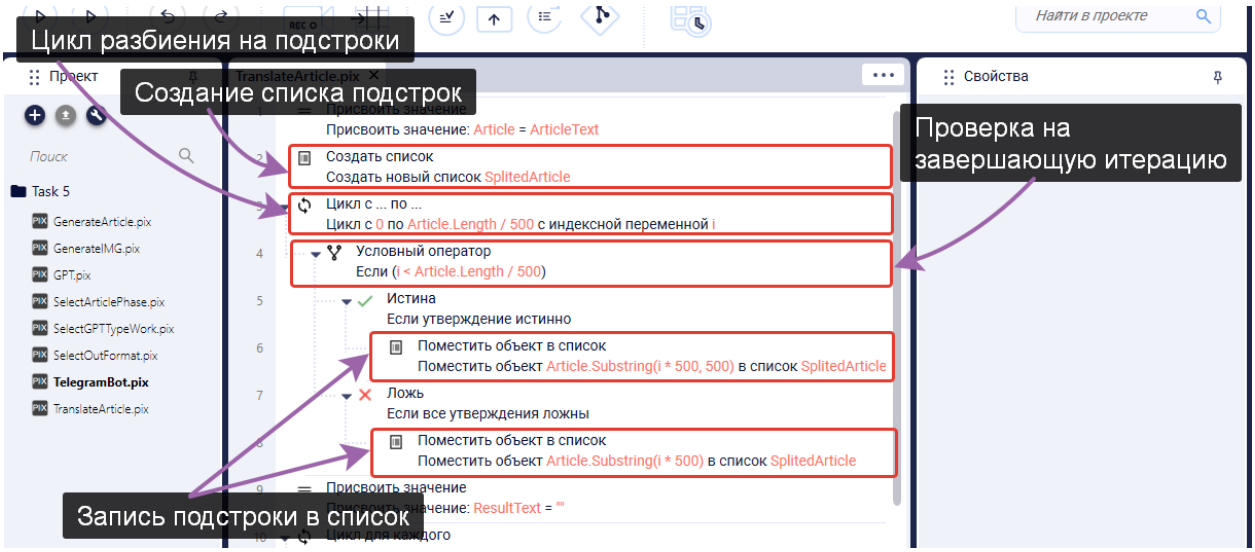


Рисунок 5.5.2. Разбиение статьи на подстроки.

Далее производятся действия аналогичные рисункам 5.3.4 и 5.4.2 с разницей, что статья отправляется не одним запросом, а частями.

5.6. Скрипт GenerateIMG.pix

При вызове скрипта необходимо передать ряд параметров в окне *«Параметры запуска скрипта»*. (Рисунок 5.6.1)

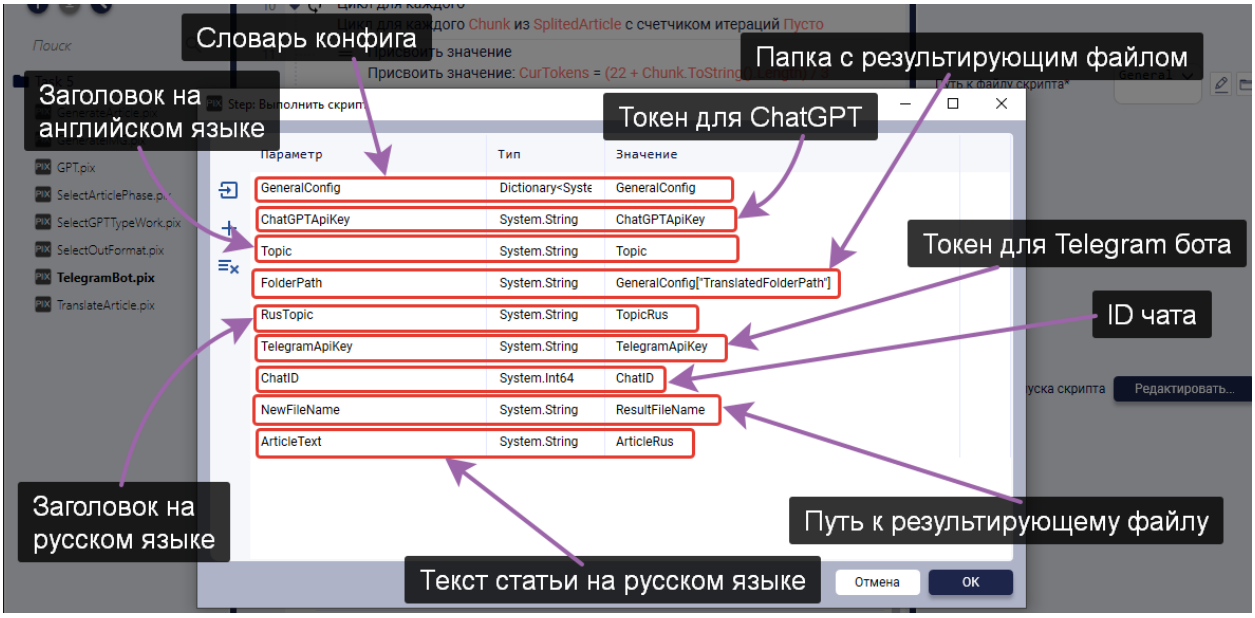


Рисунок 5.6.1. Передача параметров скрипта

В начале скрипта активностью «HTTP/Отправить HTTP запрос» отправляется запрос на генерацию изображения в соответствие с названием статьи в ChatGPT. В случае положительного ответа сервера. Из ответа способом аналогичным рисунку 5.3.4 достаётся ссылка на сгенерированное изображение. И активностью «HTTP/Отправить HTTP запрос» изображение

сохраняется в папку с результирующим файлом и отправляется пользователю в чат телеграмма.

Далее активностью «Базовые/Условный оператор» определяется выбранный способ вывода информации. В случае если информация выводится в чат, то активностью «HTTP/Отправить HTTP запрос» сначала отправляется заголовок статьи, а затем текст статьи. В ином случае с помощью активностей «Базовые/Присвоить значение» и «Базовые/Выполнить» и C# функций, связанных с отправкой запросов. В чат пользователю отправляется результирующий документ. (Рисунок 5.6.2)

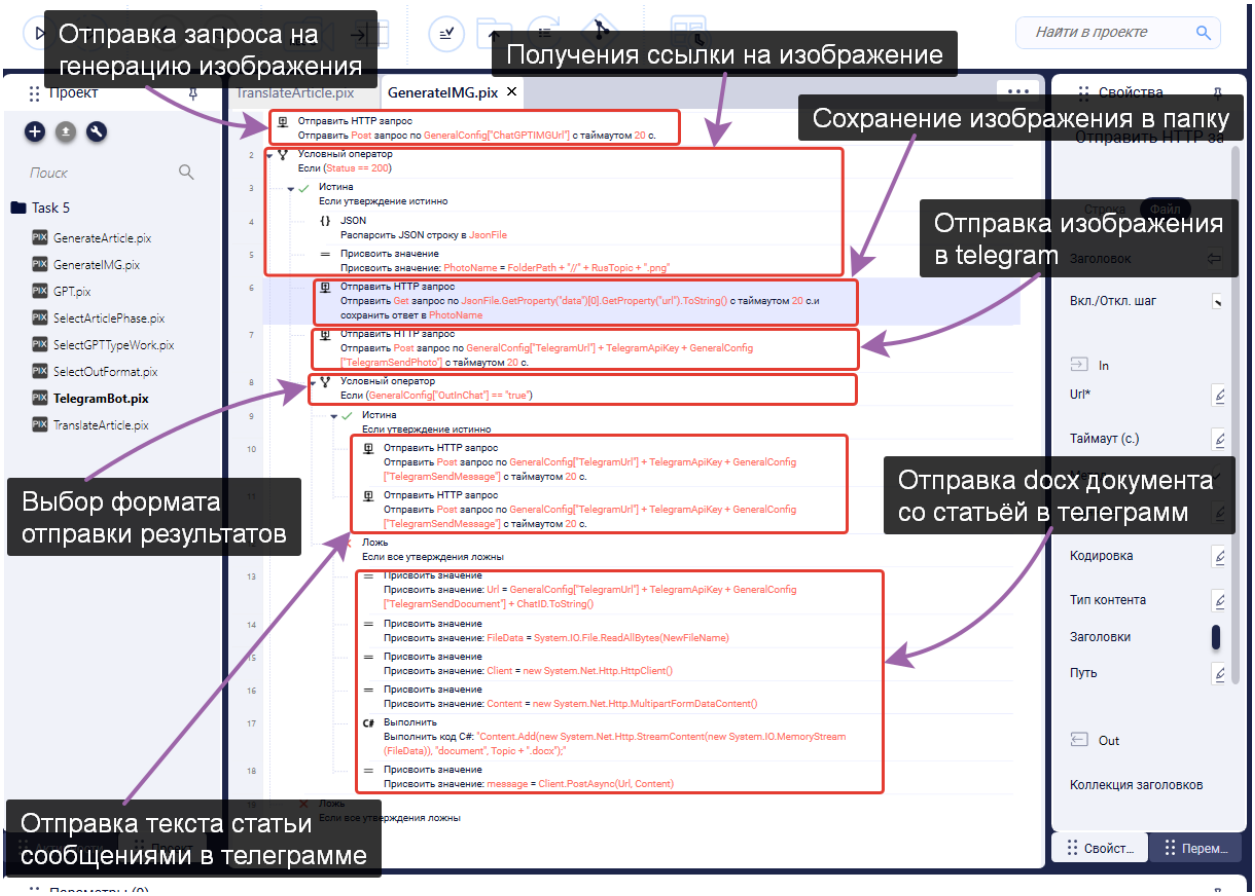


Рисунок 5.6.2. Отправка результатов

Литература

1. https://regex101.com/
2. http://2sql.ru/
3. https://core.telegram.org/
4. https://platform.openai.com/docs/models/gpt-3
5. https://knowledgebase.pixrpa.ru/actions/Excel/ReadRange
6. https://knowledgebase.pixrpa.ru/actions/Credentials/GetCredentials
7. https://knowledgebase.pixrpa.ru/actions/Base/DateTimeToString
8. https://knowledgebase.pixrpa.ru/actions/Base/GetDateTime
9. https://knowledgebase.pixrpa.ru/actions/Files/CreateFolder
10. https://knowledgebase.pixrpa.ru/actions/Files/PathExist
11. https://knowledgebase.pixrpa.ru/actions/Files/GetListFilesOrCatalogs
12. https://knowledgebase.pixrpa.ru/actions/DataBaseSQL/CreateConnection
13. https://knowledgebase.pixrpa.ru/actions/CSV/ReadCSV
14. https://knowledgebase.pixrpa.ru/actions/Base/Assign
15. https://knowledgebase.pixrpa.ru/actions/Files/CopyFileCatalog
16. https://knowledgebase.pixrpa.ru/actions/DataBaseSQL/RemoveConnection
17. https://knowledgebase.pixrpa.ru/actions/DataBaseSQL/SqlExecuteNonQuery
18. https://knowledgebase.pixrpa.ru/actions/Email/SendSMTPMailMessage
19. https://knowledgebase.pixrpa.ru/actions/Files/DeleteFile
20. https://knowledgebase.pixrpa.ru/actions/Base/LoopForEach
21. https://knowledgebase.pixrpa.ru/actions/Base/If
22. https://knowledgebase.pixrpa.ru/actions/Excel/WriteCell
23. https://knowledgebase.pixrpa.ru/actions/Telegram/CreateConnection

24. https://knowledgebase.pixrpa.ru/actions/Telegram/TelegramManager
25. https://knowledgebase.pixrpa.ru/actions/Base/TryCatch
26. https://knowledgebase.pixrpa.ru/actions/Telegram/SendMessage
27. https://knowledgebase.pixrpa.ru/actions/Strings/RegEx
28. https://knowledgebase.pixrpa.ru/actions/Base/ExecuteScript
29. https://knowledgebase.pixrpa.ru/actions/Telegram/SendPhoto
30. https://knowledgebase.pixrpa.ru/actions/Base/Return
31. https://knowledgebase.pixrpa.ru/actions/Tesseract/TesseractOCR
32. https://knowledgebase.pixrpa.ru/actions/Image/GetImage
33. http://knowledgebase.pixrpa.ru/actions/base/executescript
34. http://knowledgebase.pixrpa.ru/actions/base/executecscode
35. https://learn.microsoft.com/ru-ru/dotnet/api/system.security.securestring?cid=kerryherger&view=net-7.0
36. https://learn.microsoft.com/ru-ru/dotnet/api/system.data.datatable?view=net-7.0
37. https://learn.microsoft.com/ru-ru/dotnet/api/system.collections.generic.dictionary-2?view=net-5.0
38. https://learn.microsoft.com/ru-ru/dotnet/api/system.drawing.image?view=windowsdesktop-7.0
39. https://learn.microsoft.com/ru-ru/dotnet/api/system.drawing.bitmap?view=windowsdesktop-7.0
40. https://learn.microsoft.com/ru-RU/dotnet/api/system.datetime?view=net-5.0
41. https://learn.microsoft.com/en-us/dotnet/api/system.string?view=net-7.0
42. https://learn.microsoft.com/ru-ru/dotnet/api/system.runtime.interopservices.marshal.ptrtostringuni?view=net-7.0