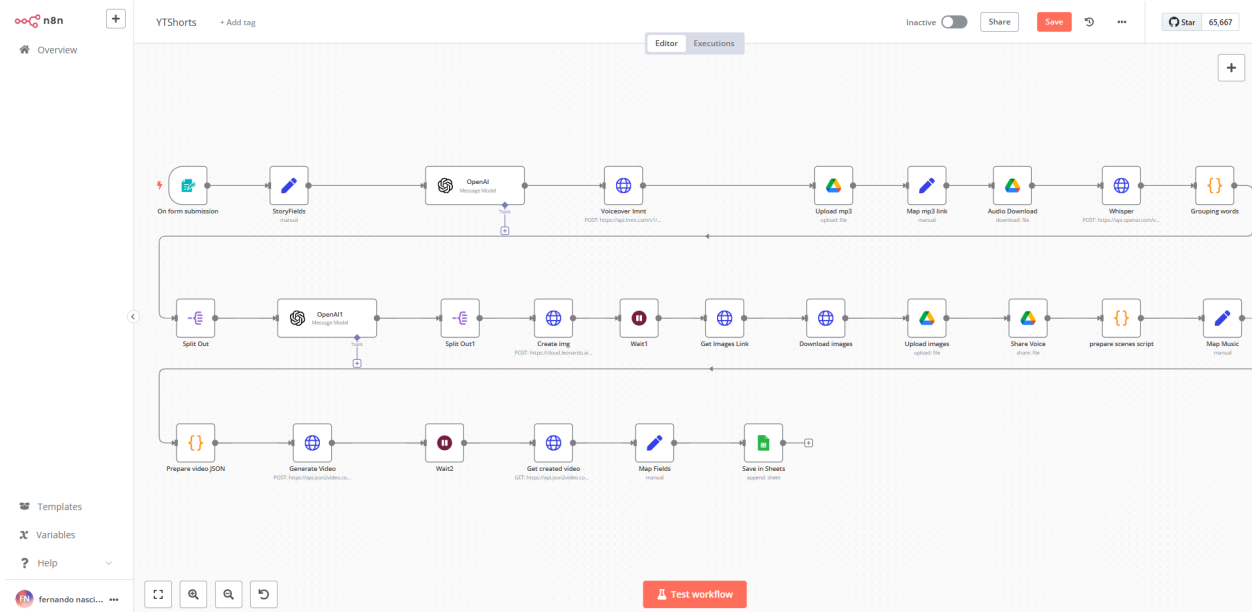


Automacao de Youtube Shorts

Imagine voce criando videos no formato youtube shorts apenas informando o assunto do video desejado! Isso mesmo, esse dia chegou.

Com essa automacao que acabei de desenvolver voce sera capaz de criar seus videos para postar no Instagram, Tiktok, youtube, etc. quase que num piscar de olhos. Veja esse exemplo!



Veremos agora as tecnologias que voce vai precisar para seguir esse tutorial:

Links:

<https://pixabay.com/music>

<https://www.lmmt.com>

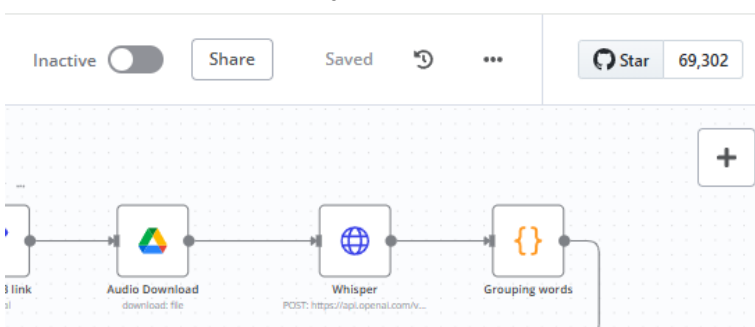
<https://json2video.com>

<https://workspace.google.com/products/drive>

<https://openai.com>

<https://leonardo.ai>

Para começar precisaremos de um node para o formulario onde o usuario informara o assunto do video e o nome do projeto. Basta vir aqui no simbolo de + e buscar por **form**.



Para facilitar esse tutorial nao vou colocar controle de unicidade no nome do projeto, isso significa que se voce colocar o mesmo nome para diferentes projetos nao vai dar legal. Ok?

Entao, abrindo o **formulario**, vamos da-lo um titulo(que pode ser qualquer titulo que voce quiser) e adicionar 2 campos: o primeiro vamos chamar de **Subject** que significa 'Assunto', sendo do tipo **TextArea** e **mandatorio**, precisando dar um check nessa botao escrito **Required**; o segundo vamos chamar de **Project Name** que significa 'Nome do Projeto', que vai ser do tipo **Text** e tambem marcamos o botao de **required**.

On form submission

Parameters Settings Docs

Form URLs

Test URL Production URL

https://n8n.saasland.shop/form-test/cee8391662-42ed-99a2-e26e4d014996

Authentication

None

Form Title

My Story

Form Description

Write the subject of your story

Form Elements

Field Name

Subject

Element Type

Textarea

Placeholder

Type the subject here

Required Field

☒

Field Name

Project Name

Element Type

OUTPUT

1 item

Subject	Project Name	submittedAt	formMode
A looser soldier that becomes the most powerful dictator.	theDictator	2025-03-10T06:23:36.387-04:00	test

On form submission

Parameters Settings Docs

Form Elements

Field Name

Subject

Element Type

Textarea

Placeholder

Type the subject here

Required Field

☒

Field Name

Project Name

Element Type

Text

Placeholder

Write a project name without spaces or special char

Required Field

☒

Add Form Element

Respond When

Form is Submitted

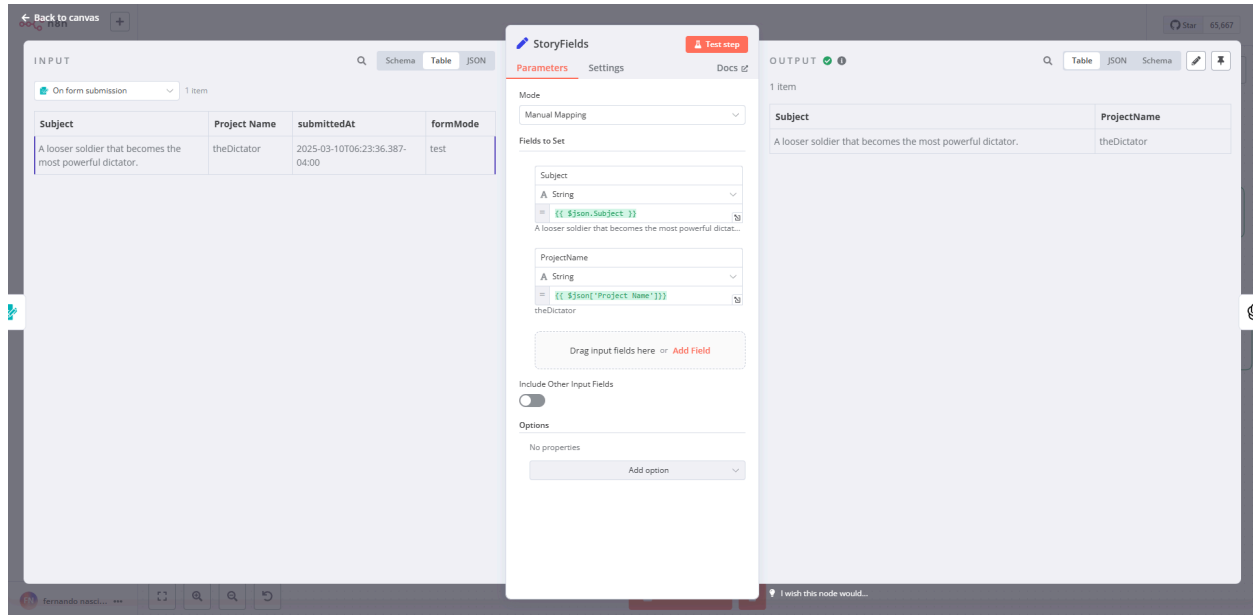
OUTPUT

1 item

Subject	Project Name	submittedAt	formMode
A looser soldier that becomes the most powerful dictator.	theDictator	2025-03-10T06:23:36.387-04:00	test

Pronto, nosso formulario esta preparado com os nossos 2 campos sendo **mandatorios**.

O próximo node que vamos usar é o **Edit Fields**, apenas para pegarmos os campos que criamos anteriormente e mapeá-los para nomes sem espaço. Nesse caso fica Subject/‘Assunto’, que não muda nada e ProjectName/‘NomeProjeto’. A razão para isso é que se colocássemos este nome sem o espaço, quando o formulário aparecesse iria, também, mostrar o texto do campo sem espaço, o que ficaria feio para um formulário.



Agora a coisa começa a ficar interessante, pois já vamos ter contato com a API da OpenAi. Portanto adicionaremos agora um novo node, basta pesquisarmos pela palavra **open** que aparece para nós o node OpenAi. Uma vez clicado, ele mostra uma lista de ações que podemos fazer com a openAi. O que serve para nossa proposta é o **Message a Model**. Abrindo as configurações desse node OpenAi, temos que selecionar a nossa credencial; a opção **Resource** fica como **Text**; o **Operation** fica como está. Para o modelo tem a opção **From List** e nessa opção vamos selecionar **GPT-4O-mini**, porque é mais barato 😊.

A opção Prompt é onde devemos colocar o nosso **prompt** para a **Ai da OpenAi**, que nesse caso vai ser esse:

“You are an youtuber video script writer. Your task is to write an interesting story based on the subject bellow.

The duration of the video should be about 20 seconds.

The tone should be of a interesting storytelling.

I want only the text as the response without any guiding text.

Do not add any closing scene, opening scene or action text, I want only voiceover text.

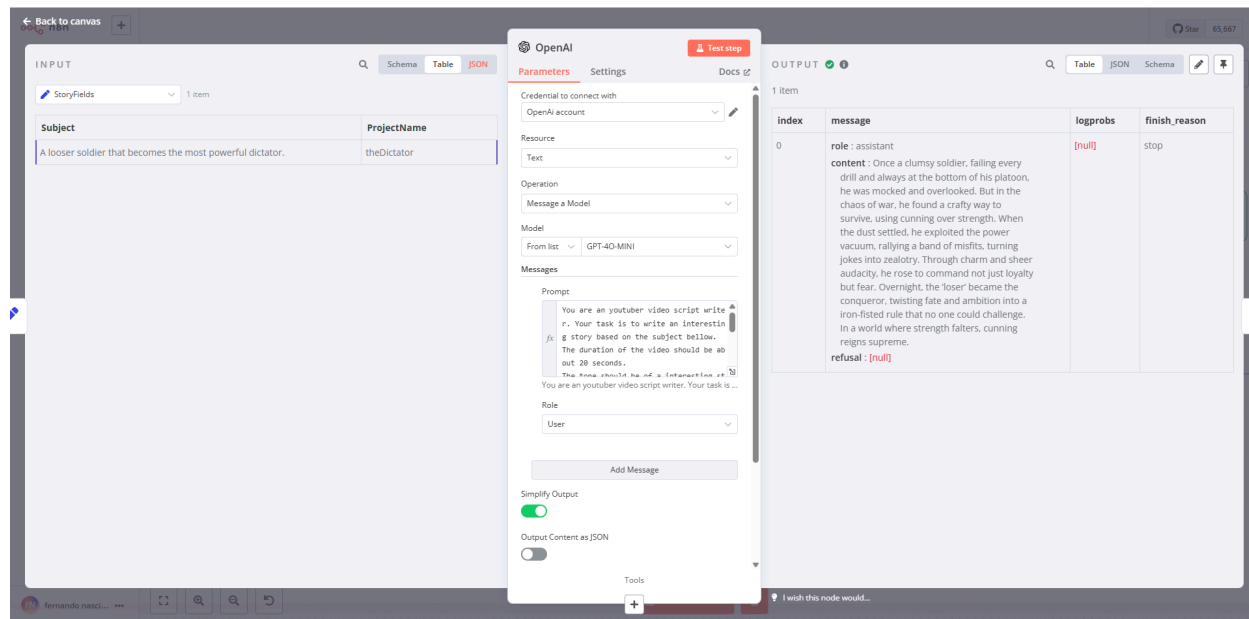
The subject is:

```
{{ $('StoryFields').item.json.Subject }}
```

”

Três informações importantes:

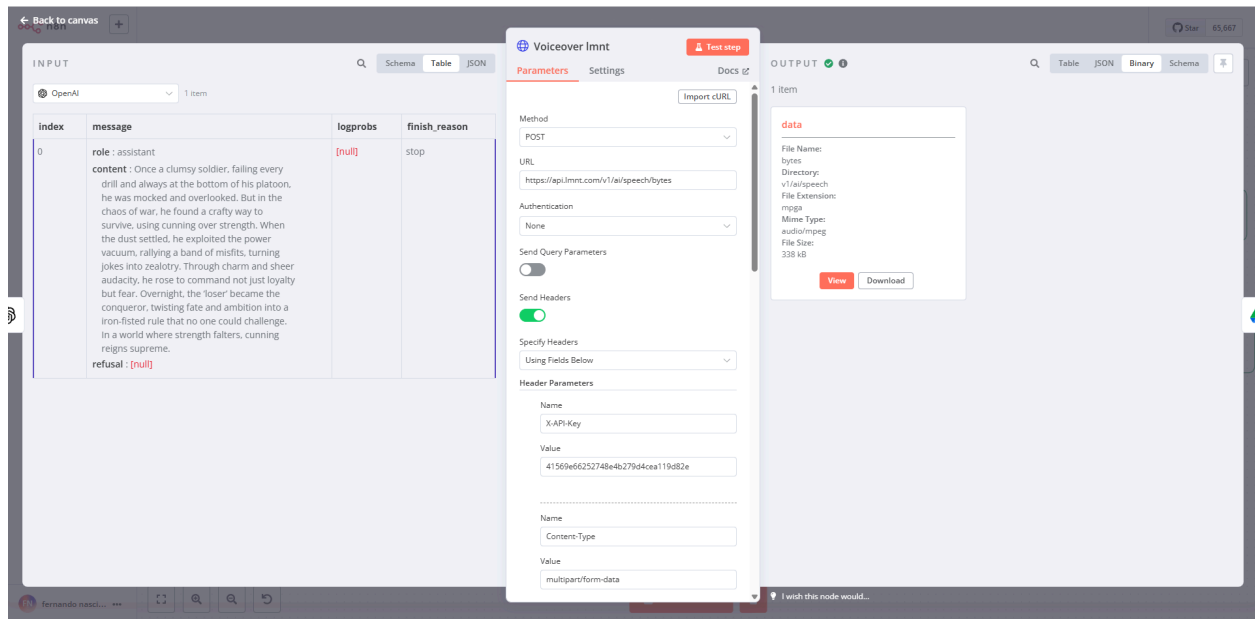
- 1 - o prompt está em inglês mas você pode usar o google tradutor para traduzi-lo, adaptar conforme o teu querer e usa-lo em português pois o ChatGPT irá entender.
- 2 - no prompt, note que tem incluído um pseudo código que está pegando o valor 'Subject' do **Edit Fields** que incluímos anteriormente.
- 3 - o chatGPT vai retornar uma história baseada no Subject/Assunto que informamos.



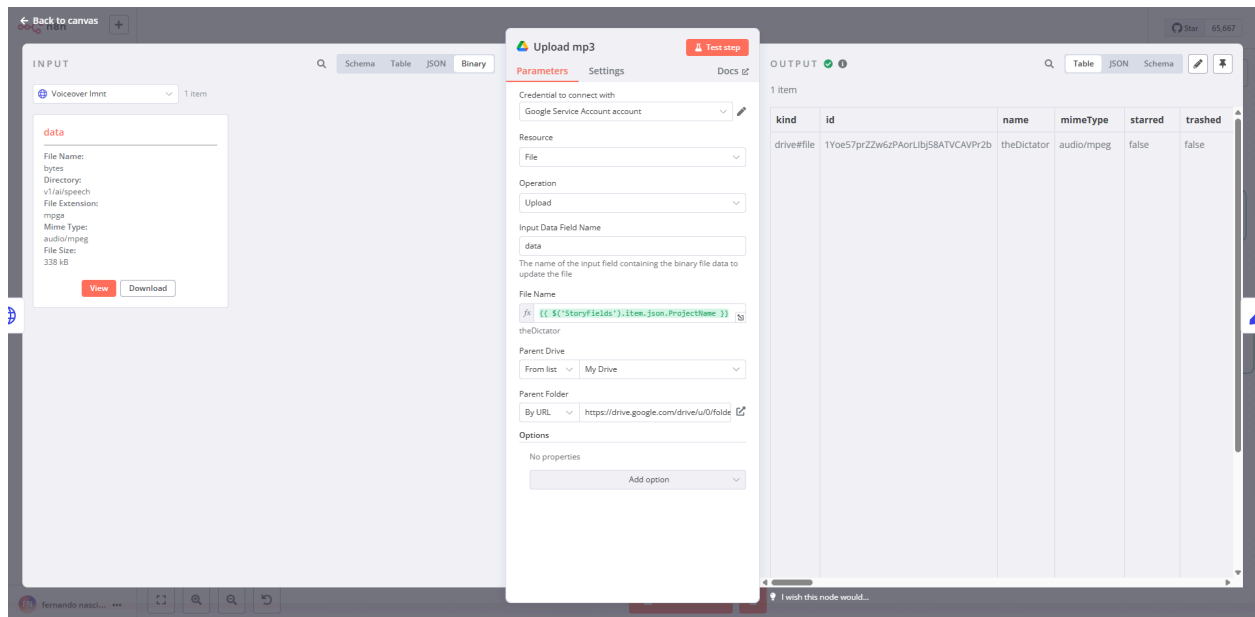
Note os valores(que nesse caso é só deixar como Role igual a User e Simplify Output marcado).

O próximo passo, agora, é enviar o script da história criada no node anterior para um serviço de geração de áudio através de texto (TTS). Vamos usar um serviço chamado LMNT. Para isso precisamos adicionar um novo node, sendo esse node de **requisição HTTP (HTTP Request)**; a opção **method** será **POST**; a **URL** será a informada no website da Api da LMNT; em **authentication** deixaremos **None**; em **Send query** deixa como está; ativaremos a opção **Send Headers**.

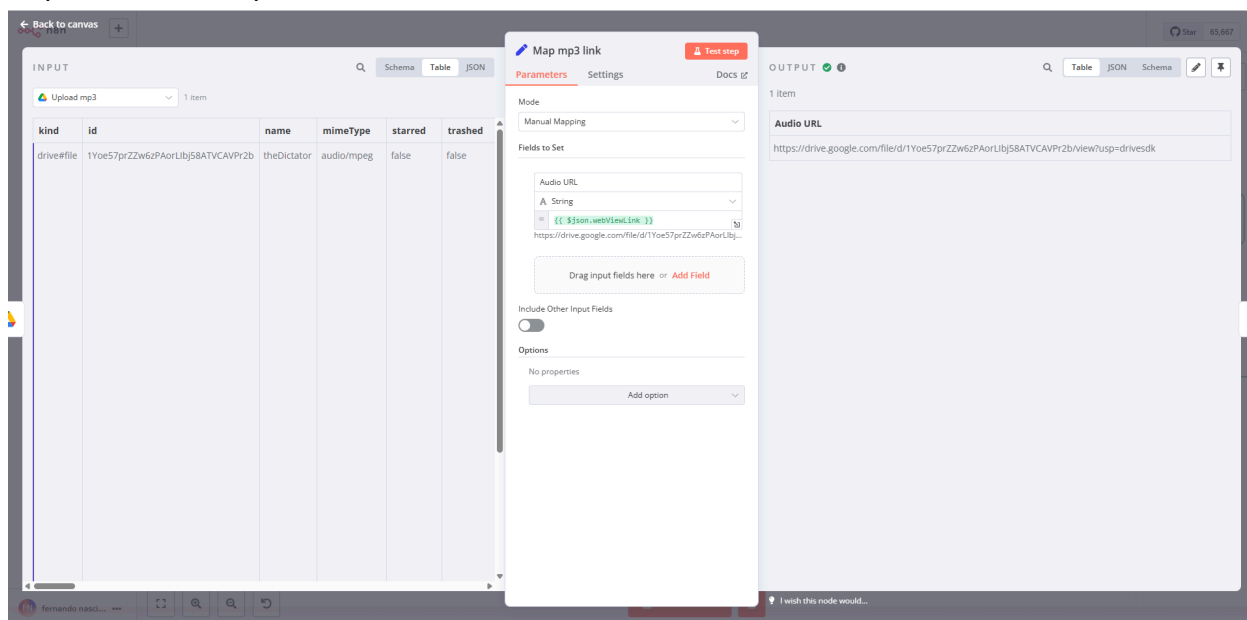
Agora precisamos informar **2 parametros** nesse **Send Headers**: o primeiro é **X-API-Key** e o valor tem que ser a tua **chave API**; o segundo é o **Content-Type** com o valor **multipart/form-data** que informa a API sobre o conteúdo **multimidia** envolvido nesse request. E para finalizar, devemos ativar a opção de **Send Body** e setar **Body Content Type** como **Form-Data** e informar os parametros para ser enviados no corpo da requisicao ao servico: **voice** como **noah** (esse é o tom de voz do narrador); para **text** temos que pegar o resultado/texto que foi gerado pelo **chatGPT** no node anterior, para isso basta colocarmos esse pseudo código **{{ \$json.message.content }}** e setar como **Expression**; o param **model** como **aurora**; e por fim **format** como **mp3**.



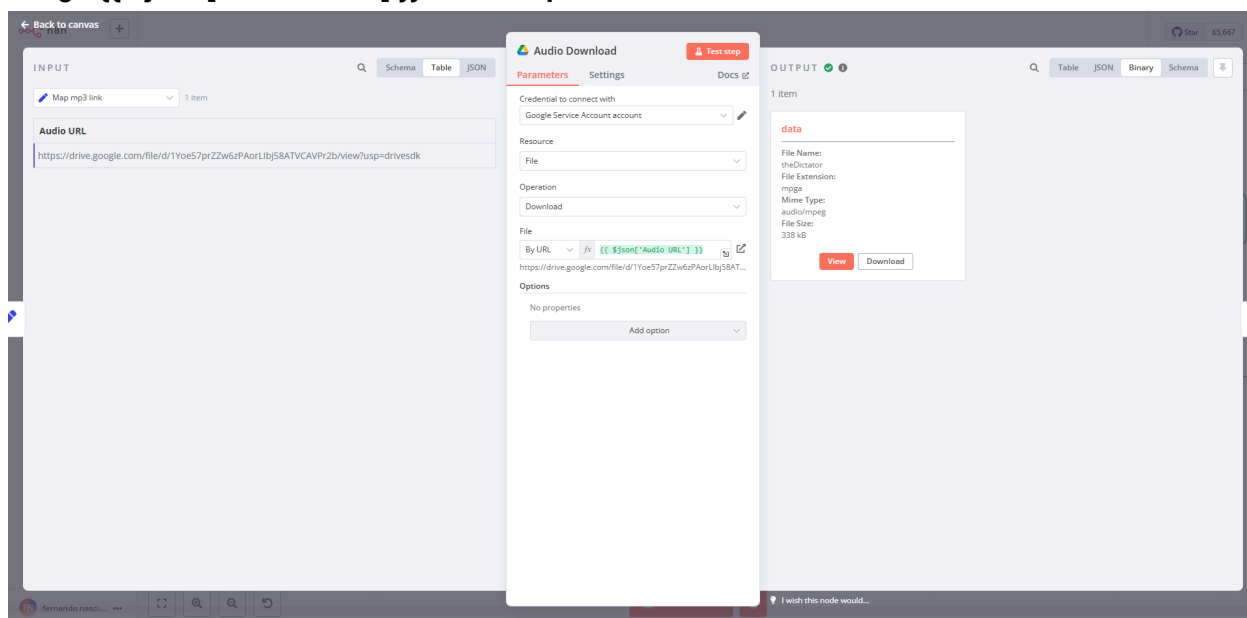
Uma vez o arquivo mp3 pronto, então vamos fazer o upload desse arquivo para a nossa pasta no google Driver. Então para isso vamos adicionar um node do **Google Driver**. Vai ser um node com a **ação** de **upload**. A configuracao eh a seguinte: selecionamos a nossa **credencial**; **Resource** como **File**; **Operation** deixa como **upload**; **Input Data** como **data**; em **Field Name** vamos colocar o nome do projeto usando esse pseudo code de origem do **Edit Field** node **{{ \$('StoryFields').item.json.ProjectName }}**; o **Parent Drive** deixamos **From List** e **My Drive**; e o **Parent Folder** vamos usar of **By URL** e colamos aqui o link do folder onde desejamos salvar o arquivo.



Ao fazer o upload, o node do Drive retorna um valor chamado **webViewLink**. Então vamos, so, mapear esse valor para um field com o nome de Audio URL usando mais um node **Edit Field**.



Uma vez que já temos o áudio na nossa pasta do Drive, só precisamos usar o field **Audio URL** para referenciar e fazer uso do arquivo mp3. Podemos, então, usar o node do Drive para download desse arquivo. Novamente selecionamos a nossa **credencial**; deixando as outras opções como estão e modificando o field **Field** com a opção **By URL** e o valor com o pseudo código `{{ $json['Audio URL'] }}` como **Expression**.



Com o Audio em mãos, vamos fazer o transcript usando o Whisper da openAi. O Whisper vai converter o audio em texto mas com as anotações de captions(que são aquelas marcações de

tempo para cada sentença). Então usaremos um node HTTP com a seguinte configuração: **Method** igual a **POST**; **URL** de transcricao da openAi <https://api.openai.com/v1/audio/transcriptions>; **Authentication** igual a **Predefined Credential Type**; **Credential Type** igual a **OpenAi**; selecionamos a nossa **openAi account**; **send query** *desativado*; **send headers** *desativado*; e **Body Parameters** incluiremos **timestamps_granularities[]** igual a **word**; **model** igual a **whisper-1**; **response_format** igual a **verbose_json**; um **parametro** do tipo **binario** com o nome **file** e **input data** como **data**, referindo-se o arquivo mp3 vindo do node anterior.

The screenshot shows the n8n interface with the Whisper node configuration. The 'Parameters' tab is active, showing the following settings:

- Method: POST
- URL: <https://api.openai.com/v1/audio/transcriptions>
- Authentication: Predefined Credential Type
- Credential Type: OpenAi
- OpenAi account: [selected]
- Send Query Parameters: ☐
- Send Headers: ☐
- Send Body: ☒
- Body Content Type: Form-Data
- Body Parameters:
 - Parameter Type: Form Data
 - Name: timestamps_granularities[]
 - Value: word

The 'OUTPUT' tab shows the result of the transcription, including task, language, duration, text, and segments.

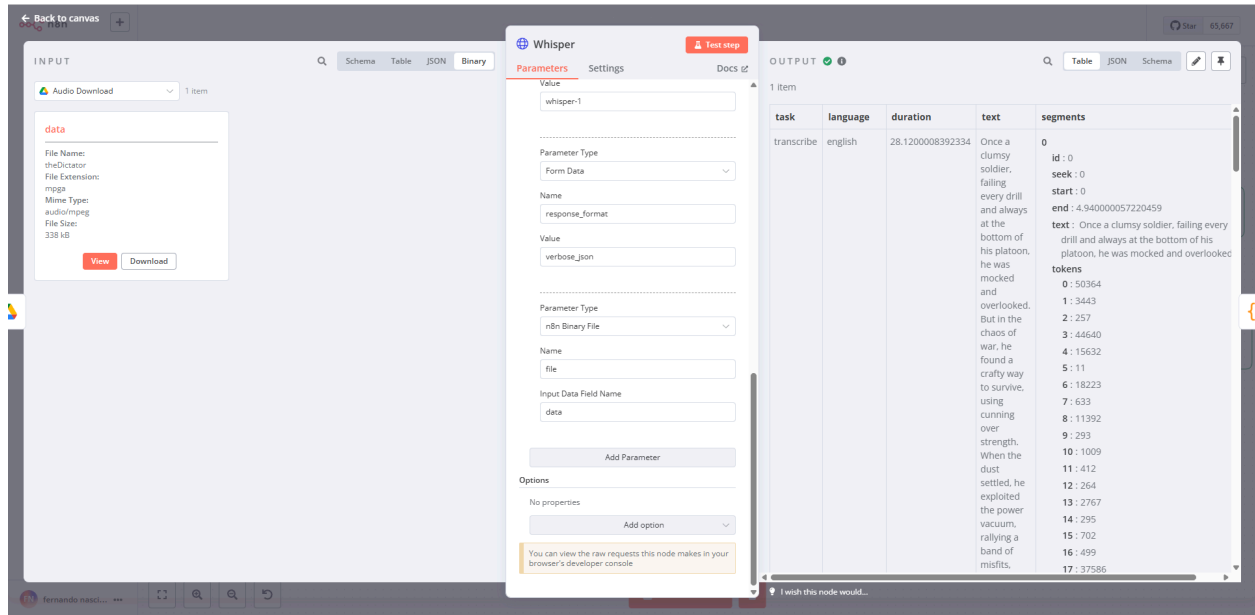
task	language	duration	text	segments
transcribe	english	28.1200008392334	Once a clumsy soldier, failing every drill and always at the bottom of his platoon, he was mocked and overlooked. But in the chaos of war, he found a crafty way to survive, using cunning over strength. When the dust settled, he exploited the power vacuum, rallying a band of misfits.	0 : 50364 1 : 3443 2 : 257 3 : 44640 4 : 15632 5 : 11 6 : 18223 7 : 633 8 : 11392 9 : 293 10 : 1009 11 : 412 12 : 264 13 : 2767 14 : 295 15 : 702 16 : 499 17 : 37586

The screenshot shows the n8n interface with the Whisper node configuration. The 'Parameters' tab is active, showing the following settings:

- Method: POST
- URL: <https://api.openai.com/v1/audio/transcriptions>
- Authentication: Predefined Credential Type
- Credential Type: OpenAi
- OpenAi account: [selected]
- Send Query Parameters: ☐
- Send Headers: ☐
- Send Body: ☒
- Body Content Type: Form-Data
- Body Parameters:
 - Parameter Type: Form Data
 - Name: timestamps_granularities[]
 - Value: word

The 'OUTPUT' tab shows the result of the transcription, including task, language, duration, text, and segments.

task	language	duration	text	segments
transcribe	english	28.1200008392334	Once a clumsy soldier, failing every drill and always at the bottom of his platoon, he was mocked and overlooked. But in the chaos of war, he found a crafty way to survive, using cunning over strength. When the dust settled, he exploited the power vacuum, rallying a band of misfits.	0 : 50364 1 : 3443 2 : 257 3 : 44640 4 : 15632 5 : 11 6 : 18223 7 : 633 8 : 11392 9 : 293 10 : 1009 11 : 412 12 : 264 13 : 2767 14 : 295 15 : 702 16 : 499 17 : 37586



Quando o Whisper retornar com o valor precisaremos dar uma formatada nele. Então agora incluiremos um node Code e nele adicionaremos esse código javascript:

```
return items.map(item => {

  const words = item.json.segments;

  const result = [];

  let previousStart = words[0].start;

  let currentGroup = [];

  let index = 0;
  for(const word of words){
    if(word.id % 2 === 0 || word.id === 0){
      currentGroup = [word.text];
      previousStart = word.start;
    } else {
      index++;
      currentGroup.push(word.text);
      result.push({
        text: currentGroup.join(' '),
        start: previousStart,
        end: word.end,
        index: index
      });
    }
  }
}
```



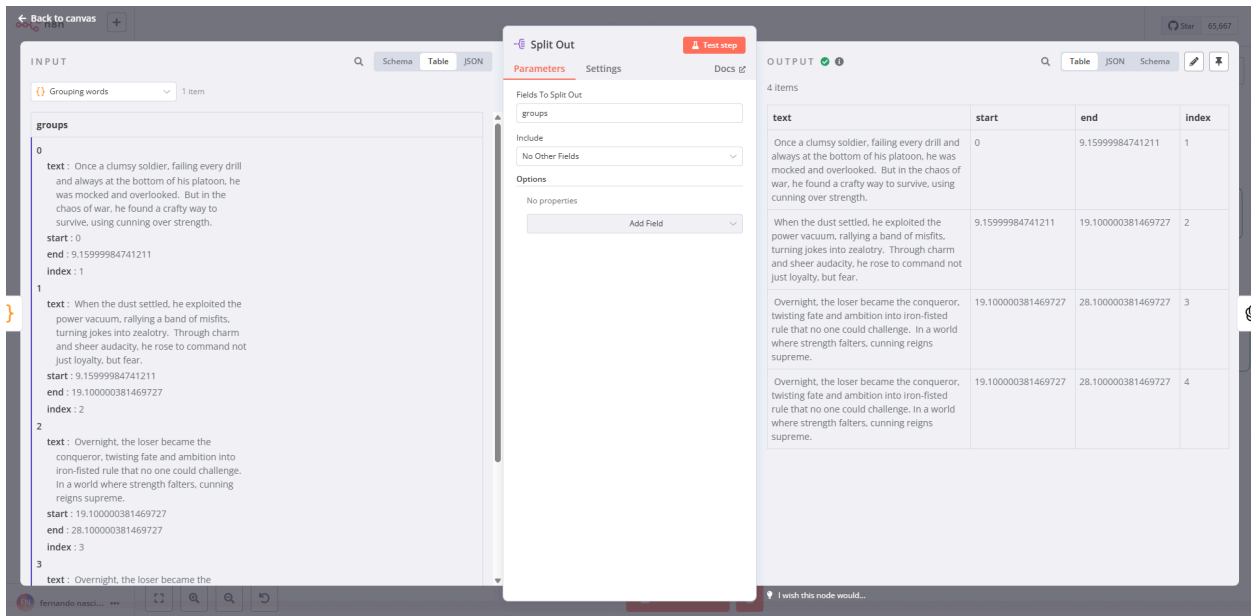
```

if(currentGroup.length > 0){
  index++;
  result.push({
    text: currentGroup.join(""),
    start: previousStart,
    end: words[words.length - 1].end,
    index: index
  });
}

return {json: {groups: result}}
});

```

Após a formatação precisamos dar um split nos valores. Para isso vamos adicionar um node **Split** e informar que o field usado no split deve ser o groups.



The screenshot shows the Node-RED interface with a 'Split Out' node configuration. The 'INPUT' section displays a 'groups' array with four items. The 'Parameters' section shows 'Fields To Split Out' set to 'groups'. The 'OUTPUT' section displays a table with 4 items, each containing 'text', 'start', 'end', and 'index'.

text	start	end	index
Once a clumsy soldier, failing every drill and always at the bottom of his platoon, he was mocked and overlooked. But in the chaos of war, he found a crafty way to survive, using cunning over strength.	0	9.15999984741211	1
When the dust settled, he exploited the power vacuum, rallying a band of misfits, turning jokes into zealotry. Through charm and sheer audacity, he rose to command not just loyalty, but fear.	9.15999984741211	19.100000381469727	2
Overnight, the loser became the conqueror, twisting fate and ambition into iron-fisted rule that no one could challenge. In a world where strength falters, cunning reigns supreme.	19.100000381469727	28.100000381469727	3
Overnight, the loser became the conqueror, twisting fate and ambition into iron-fisted rule that no one could challenge. In a world where strength falters, cunning reigns supreme.	19.100000381469727	28.100000381469727	4

Nesta etapa vamos criar os prompts para as imagens. Vamos fazer com que a própria openAi gere esses prompts para nós. Então adicionamos mais um **openAi** node com ação de **message a modelo** e a configuração será a mesma como anteriormente sendo que a única mudança será no **prompt**. Fiquem a vontade para melhorar esse prompt de acordo com a vossa vontade:

You are an image prompt designer, your task is to convert the following transcript text into image prompt for Leonardo AI image generation model.

Output the prompt directly without things like:

*****Do not include '\",', double quotes and single quotes".**

*****Keep the personage or character consistency between the scenes.**

Here is the transcript: `{{ $json.text }}`

The screenshot shows a workflow editor with an OpenAI1 node. The INPUT table has the following data:

text	start	end	index
Once a clumsy soldier, falling every drill and always at the bottom of his platoon, he was mocked and overlooked. But in the chaos of war, he found a crafty way to survive, using cunning over strength.	0	9.15999984741211	1
When the dust settled, he exploited the power vacuum, rallying a band of misfits, turning jokes into zealotry. Through charm and sheer audacity, he rose to command not just loyalty, but fear.	9.15999984741211	19.100000381469727	2
Overnight, the loser became the conqueror, twisting fate and ambition into iron-fisted rule that no one could challenge. In a world where strength falters, cunning reigns supreme.	19.100000381469727	28.100000381469727	3
Overnight, the loser became the conqueror, twisting fate and ambition into iron-fisted rule that no one could challenge. In a world where strength falters, cunning reigns supreme.	19.100000381469727	28.100000381469727	4

The OUTPUT table shows the result of the OpenAI1 node:

index	message	logprobs	finish_reason
0	role : assistant content : A clumsy soldier in a war-torn environment, wearing a tattered uniform, looking frustrated and out of place among his well-trained platoon members. The scene is filled with chaos: explosions in the background, smoke rising, and soldiers maneuvering strategically. The soldier, with a determined yet resourceful expression, is cleverly using a rucksack and makeshift tools to outsmart an enemy. He embodies an underdog spirit, showcasing his cunning and adaptability amidst the turmoil of battle. The atmosphere is tense and dynamic, capturing the essence of survival through ingenuity in warfare. refusal : [null]	[null]	stop
0	role : assistant content : A charismatic leader standing confidently amidst a cloud of dust, rallying a diverse group of misfits around him. The scene is vibrant and dynamic, with expressions of determination and admiration on the faces of his followers. The leader, with an audacious grin, exudes a magnetic charm, blending humor and intensity. The atmosphere is charged with a sense of rebellion and transformation, illustrating the shift from laughter to fervent loyalty. In the background, hints of a chaotic environment symbolize the power vacuum he has seized.	[null]	stop

Nota-se que esse `json.text` eh o script já separado com o captions e o output está marcado como Simplify Output.

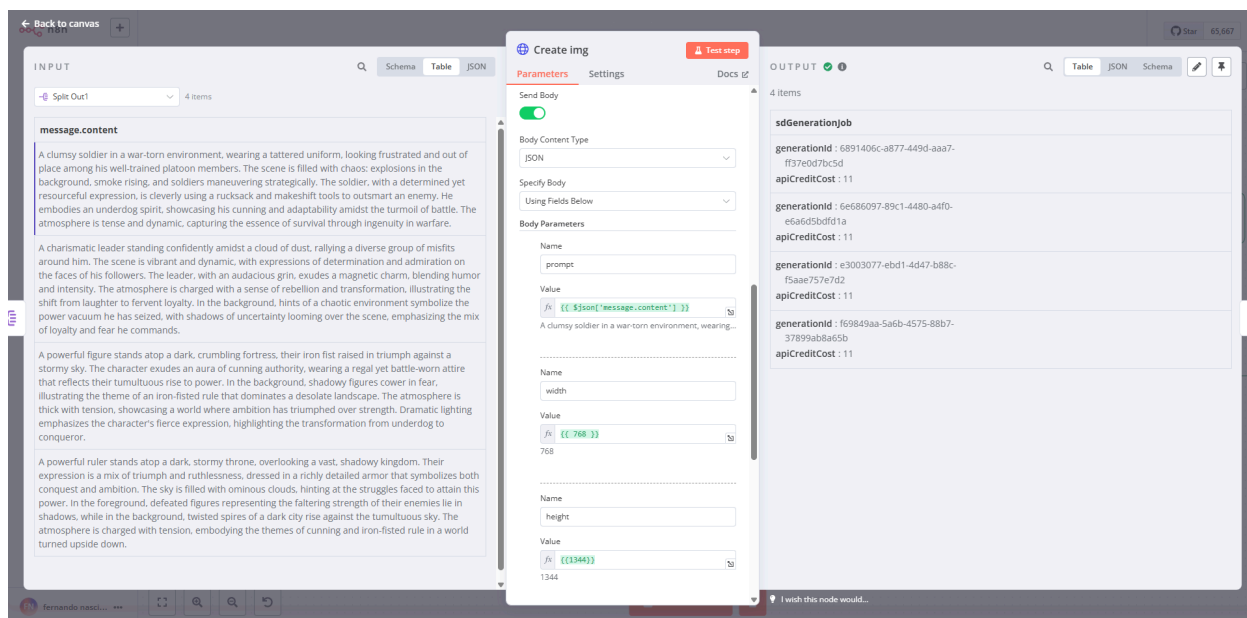
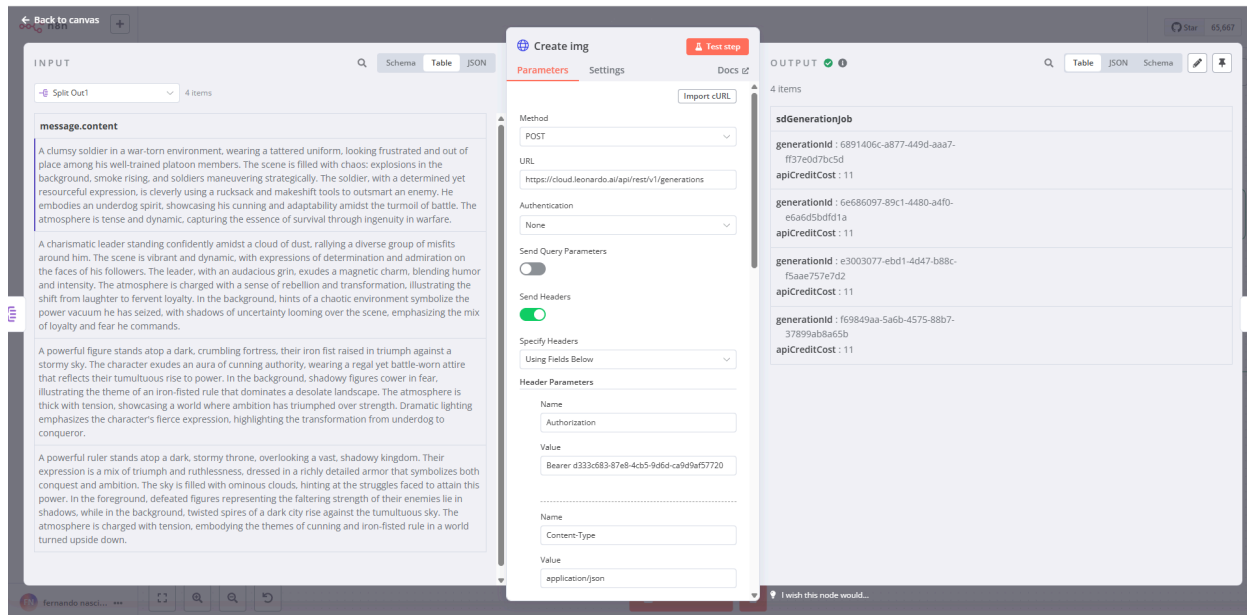
Agora incluiremos mais um **Split** node e usaremos como referência para split o campo `message.content` provindo do resultado da geração dos **images prompts** da openAi.

The screenshot shows the workflow editor with a Split Out1 node. The INPUT table is the same as in the previous screenshot. The OUTPUT table shows the result of the Split Out1 node:

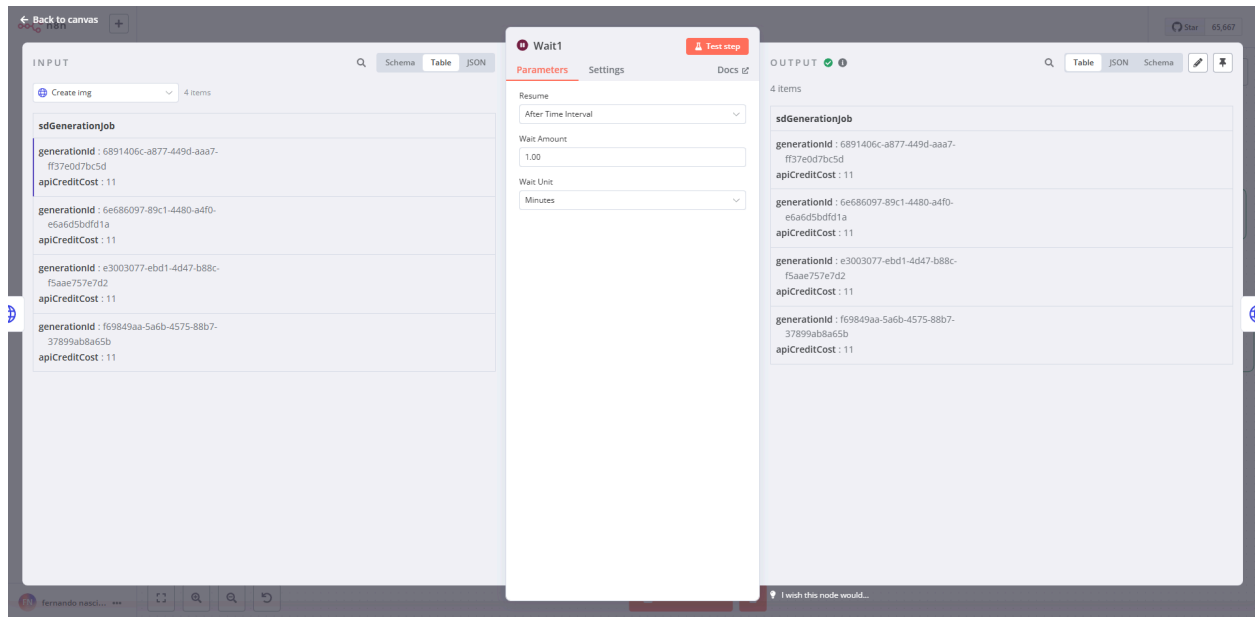
index	message	logprobs	finish_reason
0	role : assistant content : A clumsy soldier in a war-torn environment, wearing a tattered uniform, looking frustrated and out of place among his well-trained platoon members. The scene is filled with chaos: explosions in the background, smoke rising, and soldiers maneuvering strategically. The soldier, with a determined yet resourceful expression, is cleverly using a rucksack and makeshift tools to outsmart an enemy. He embodies an underdog spirit, showcasing his cunning and adaptability amidst the turmoil of battle. The atmosphere is tense and dynamic, capturing the essence of survival through ingenuity in warfare. refusal : [null]	[null]	stop
0	role : assistant content : A charismatic leader standing confidently amidst a cloud of dust, rallying a diverse group of misfits around him. The scene is vibrant and dynamic, with expressions of determination and admiration on the faces of his followers. The leader, with an audacious grin, exudes a magnetic charm, blending humor and intensity. The atmosphere is charged with a sense of rebellion and transformation, illustrating the shift from laughter to fervent loyalty. In the background, hints of a chaotic environment symbolize the power vacuum he has seized.	[null]	stop

Agora que já temos os prompts para as imagens, vamos envia-los para a nossa API do **LeonardoAi**. Para isso adicionaremos um node **HTTP Request** com o method **Post**, URL <https://cloud.leonardo.ai/api/rest/v1/generations> , **Authentication** igual a **none**, **Send Query**

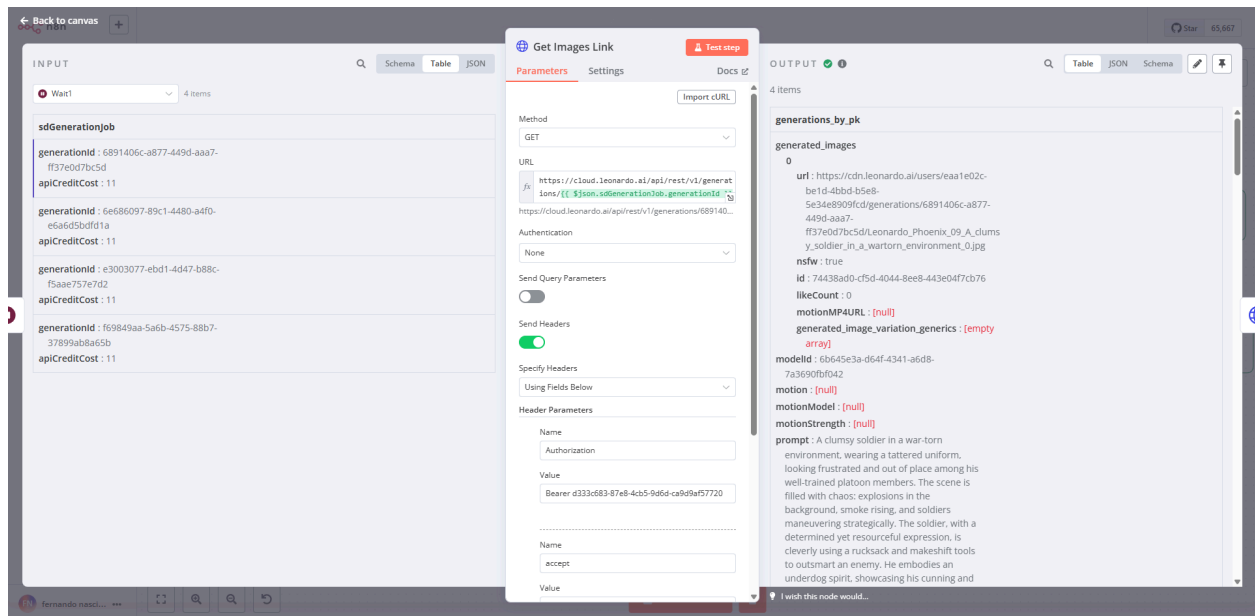
desativado, **Send Headers** com 2 parametros (**Authorization** igual ao teu **token** com a palavra **Bearer** na frente, e **Content-Type** como **application/json**). Também temos que ativar o **Send Body** selecionando **Body Content Type** como **Json** com 5 attributes (**prompt** = **message.content** do node anterior, **width** = **768**, **height** = **1344**, **num_images** = **1** e **modelId** = **6b645e3a-d64f-4341-a6d8-7a3690fbf042**).

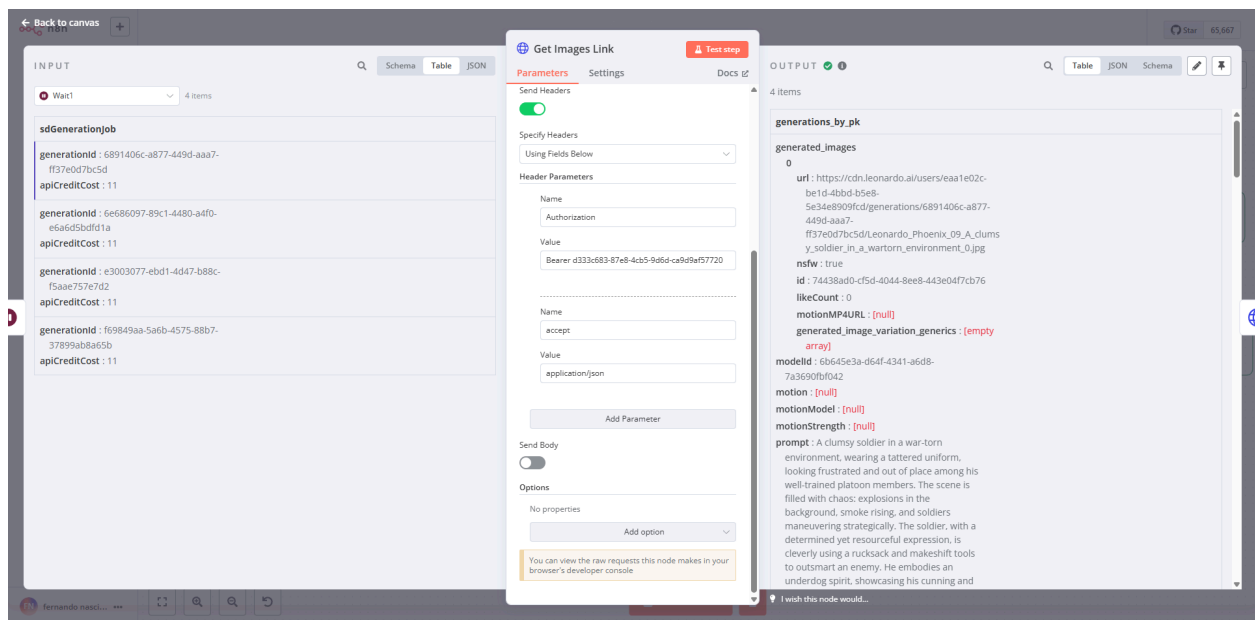


Temos que esperar até que o LeonardoAi termine de gerar as imagens, então vamos adicionar um node **Wait** com **1** minute de espera.

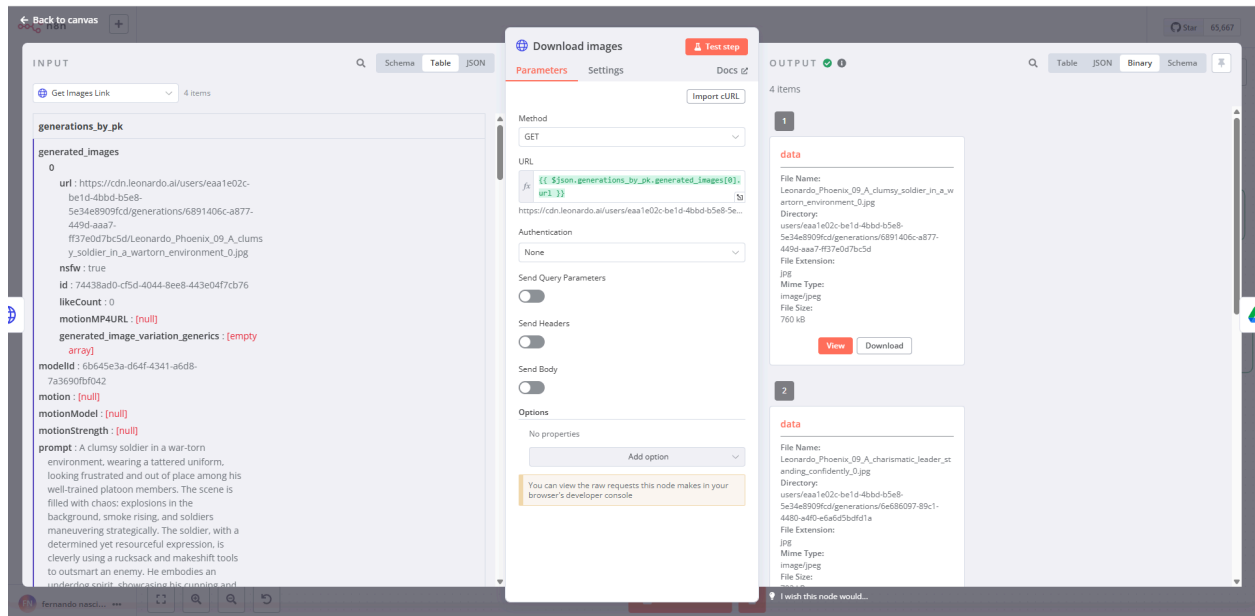


Agora temos que buscar as imagens geradas, então vamos adicionar novamente um node **HTTP Request**. Podemos copiar e colar esse mesmo node HTTP que usamos para gerar as imagens pois o esquema de **autenticacao** com o token já está pronto. Devemos fazer pequenas alterações, como trocar o method de **POST** para **GET**; adicionar esse pseudo código apos o link de conexão **/{{ \$json.sdGenerationJob.generationId }}**; no **headers** deixar a **Authorization** com o token e adicionar **accept = application/json**; e por fim **desativar o Send Body**.





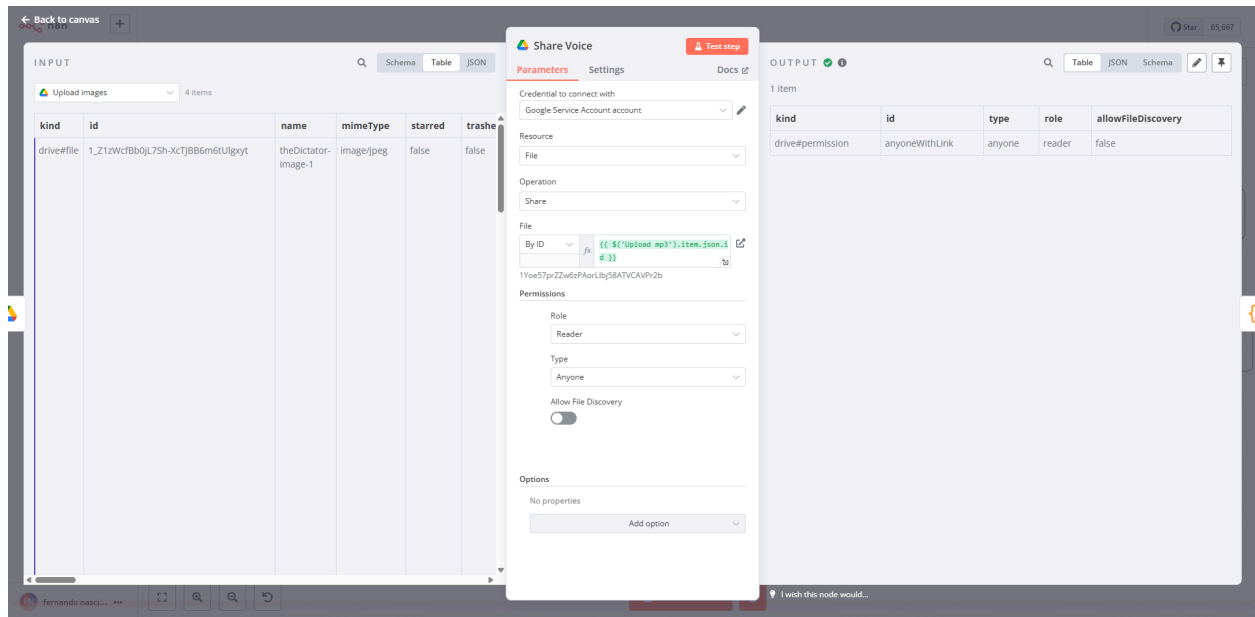
E de novo adicionar uma vez mais um node **HTTP Request** para realizar o download das imagens baseando-se nos links das imagens geradas no node anterior que acessou o LeonardoAi. Para isso, usaremos o **method** como **GET** e em **URL** vamos usar esse pseudo código que faz um loop no atributo generated_images retornado anteriormente **{{ \$json.generations_by_pk.generated_images[0].url }}**. Lembre-se que sempre que usamos esses pseudo códigos devemos alterar a configuração do campo para **Expression**.



Fizemos o download das imagens e agora vamos fazer o upload delas para a nossa pastinha no Drive. Então adicionamos um node **Drive do Google** com a ação de **upload**. Na configuração do mesmo colocamos: a nossa **credencial**, **Resource** = file, **operation** deixa como **upload**, **Input Data** igual a **data**, **File Name** usamos esse pseudo código com o nome do projeto-image-essa variavel com o index do loop `{{ $('StoryFields').item.json.ProjectName }}-image-{{ $itemIndex + 1 }}`. From list = **My drive** e **Parent Folder** by **URL** com o nosso link da pasta <https://drive.google.com/drive/u/0/folders/4-08yblUGJDGBFDOIFHDUIDVLrh>

The screenshot displays the n8n workflow editor. On the left, the 'INPUT' section shows two data items. Each item contains a 'data' object with file metadata such as 'File Name', 'Directory', 'File Extension', 'Mime Type', 'File Size', and 'File Size'. The first item's file name is 'Leonardo_Phoenix_09_A_dumy_soldier_in_a_w...'. The second item's file name is 'Leonardo_Phoenix_09_A_charismatic_leader_st...'. In the center, the 'Upload images' node is configured. The 'Parameters' tab is selected, showing settings for connecting to a Google Service Account, uploading files, and specifying the file name using a template string: `{{ $('StoryFields').item.json.ProjectName }}-image-{{ $itemIndex + 1 }}`. The parent folder is set to 'By URL' with the link `https://drive.google.com/drive/u/0/folders/4-08yblUGJDGBFDOIFHDUIDVLrh`. On the right, the 'OUTPUT' section shows a table with 4 items. The first item is a drive file with the ID '1_Z12Wcf8bQJL7Sh-XCTJB6m6tUgxyt' and the name 'theDictator-image-1'.

Para a geração do video utilizaremos uma outra API, então lembrei que o nosso audio mp3 que guardamos na nossa pasta no Drive não está acessível. Então devemos incluir mais um node do **Google Drive** com a Action **Share**. Selecionamos a nossa **credencial** e em **file** colocamos **By ID** e indicamos o id do nosso arquivo mp3 gerado com o LMNT. Em permissões colocamos **Role** = Reader and **Type** = Anyone.



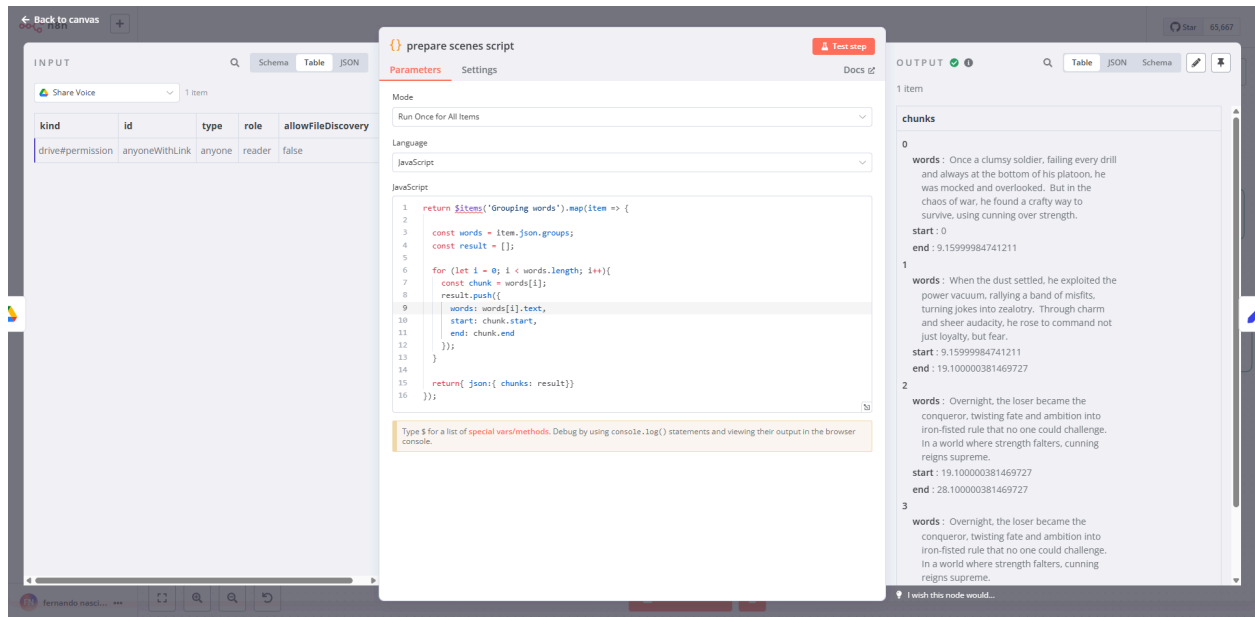
Chegou a hora de separarmos nosso texto da história de acordo com a mudança das cenas. Isso quer dizer que vamos deixar organizado em que momento um texto e uma imagem vai aparecer ou desaparecer no vídeo, a sincronização das cenas.

Basta copiar e colar esse **javascript** nesse próximo node que é um **Code**:

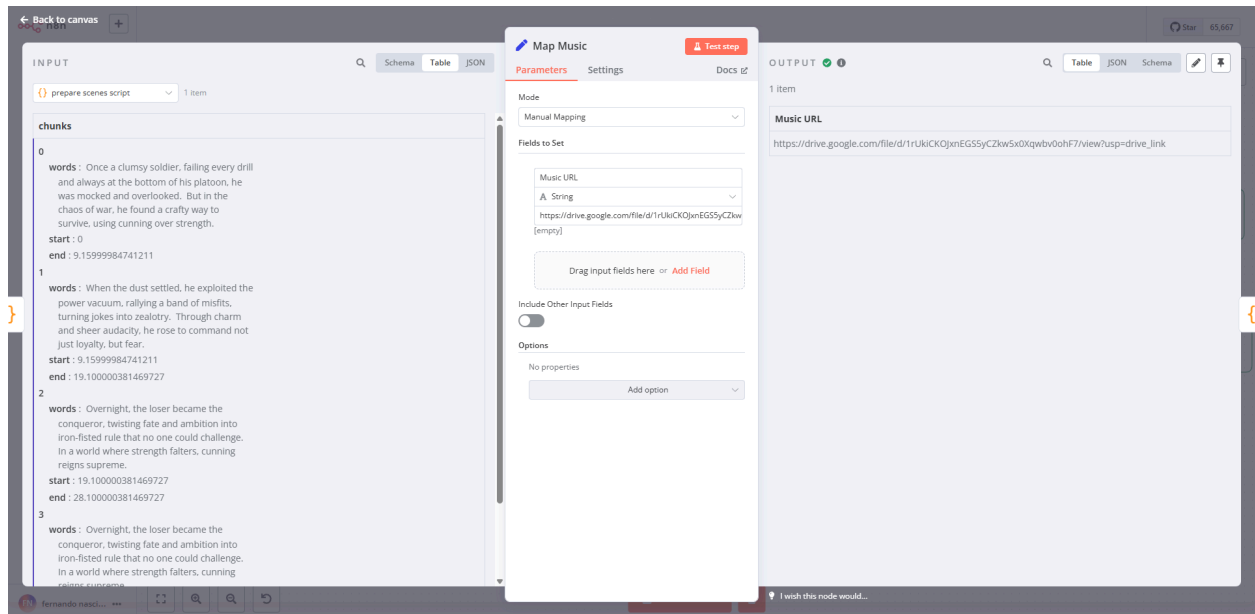
```
return $items('Grouping words').map(item => {
```

```
    //const words = item.json.segments; // originary would be from whisper
    const words = item.json.groups;
    const result = [];
    console.log(words);
    for (let i = 0; i < words.length; i++) { //i+= 3){
        //const chunk = words.slice(i, i + 3); in case it came as words only
        const chunk = words[i];
        result.push({
            words: words[i].text, //chunk.map(word => word.word.charAt(0).toUpperCase() +
            word.word.slice(1)).join(' '),
            start: chunk.start, //chunk[0].start,
            end: chunk.end //chunk[chunk.length - 1].end
        });
    }

    return { json: { chunks: result } }
});
```



Quanto a música de background, essa não será dinâmica. Precisamos adicionar um mp3 na nossa pasta no Drive, copiar o link do arquivo e colar nesse novo node do tipo **Edit Field** com a única e exclusiva tarefa de guardar esse link para nós. No mode colocamos **Manual Mapping** e em **Fields to Set** damos o nome de **Music URL**.



Estamos quase no final, então já é a hora de prepararmos a geração do vídeo. Para isso precisamos de um novo node **Code** e com esse **javascript** mapeamos todos os dados necessários para o vídeo, como: a lista de imagens, a música de fundo, os texto das cenas e o áudio da narração.


```

var imageIndex = 0;
var result = {};
var scenes = [];
const projName = $('StoryFields').first().json.ProjectName;

const itemArray = $items('Download images');

$node['prepare scenes script'].json.chunks.map(item => {
  const imageURL = itemArray[imageIndex].json.generations_by_pk.generated_images[0].url;
  const sceneDescription = item.words;
  const duration = item.end - item.start;
  const sceneld = projName + "_Scene_" + imageIndex;
  scenes.push(
    {
      "id": sceneld,
      "comment": "Scene " + imageIndex,
      "elements": [
        {
          "id": sceneld + "_Element_1",
          "type": "image",
          "src": imageURL,
          "scale": {
            "width": 1080,
            "height": 1920
          },
          "x": 0,
          "y": 0,
          "position": "center-center",
          "zoom": 5
        },
        {
          "id": sceneld + "_Element_2",
          "type": "component",
          "component": "basic/000",
          "settings": {
            "headline": {
              "text": sceneDescription,
              "color": "white",
              "font-family": "EB Garamond",
              "text-align": "center",
              "font-size": "8vw",
              "padding": "3vw 0"
            },
            "body": {

```

```

        "color": "white",
        "text-align": "center",
        "font-family": "EB Garamond",
        "font-size": "5vw"
    },
    "card": {
        "vertical-align": "bottom",
        "margin": "5vw",
        "background-color": "rgba(0,0,0,0.5)",
        "border-radius": "2vw"
    }
},
"width": 1080,
"height": 1800,
"x": 0,
"y": 0,
"duration": Math.trunc(duration),
"comment": "Simple card",
"position": "custom"
}
]
}
);

```

```

    imageIndex++;

```

```

});

```

```

result = {
    id: projName,
    comment: "Default movie",
    width: 1080,
    height: 1920,
    quality: "medium",
    draft: true,
    scenes: scenes,
    "elements": [
        {
            "id": projName + "_bg_music",
            "type": "audio",
            "volume": 1,
            "fade-out": 1,
            "src": $input.first().json["Music URL"],
            "duration": -2,

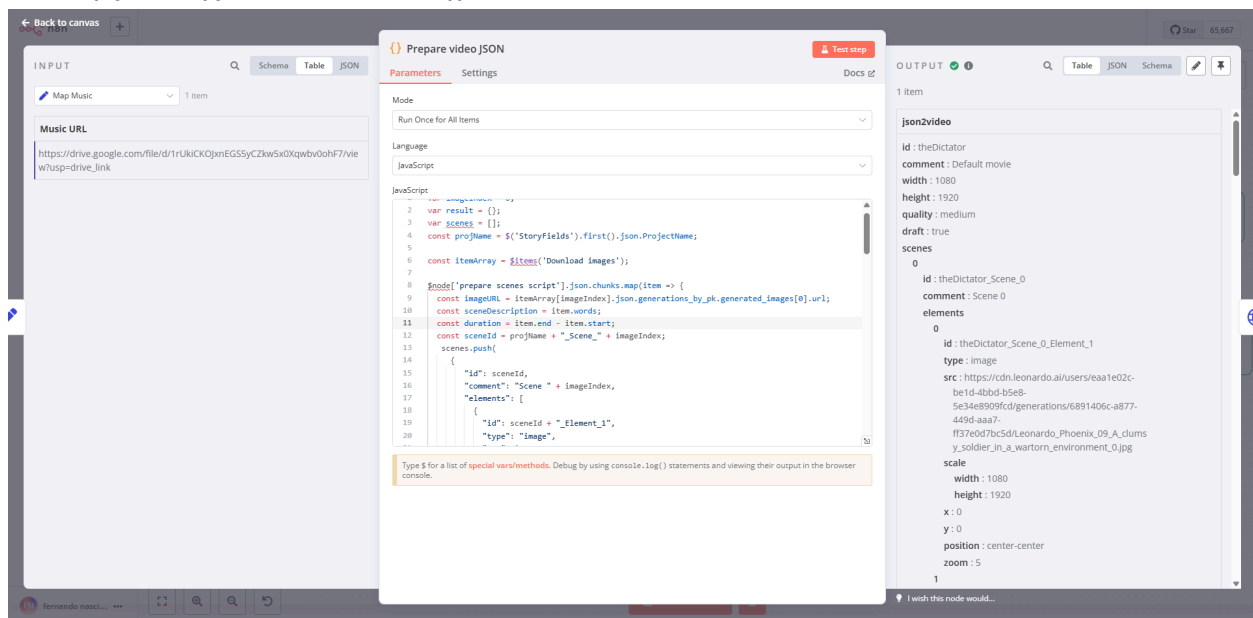
```

```

    "fade-out": 2,
    "fade-in": 1
  },
  {
    "id": projName + "_voiceover",
    "type": "audio",
    "src": $('Audio Download').first().json['Audio URL']
  }
],
"resolution": "instagram-story",
"fps": 25,
"settings": {},
"cache": true,
"variables": {}
};

```

```
return { "json": {"json2video":result}}
```



Com os dados para o vídeo mapeados então incluiremos um node de **HTTP Request** para efetuarmos o request para a API do gerador de vídeo. A configuração é muito simples, basta alterar o **Method** para **POST**, usar essa url <https://api.json2video.com/v2/movies>, **Authentication** = none e sem **Send Query**. Em **Send Headers** usamos o **x-api-key** = ao token, **Content-Type** = application/json. Em **Send Body** colocamos como **JSON using JSON** e no campo **json** informamos o conteúdo do node anterior, de preparação do json de video `{{ $json.json2video }}`.

INPUT

```
{} Prepare video JSON 1 item
```

json2video

```
{
  id: theDictator
  comment: Default movie
  width: 1080
  height: 1920
  quality: medium
  draft: true
  scenes:
    0:
      id: theDictator_Scene_0
      comment: Scene 0
      elements:
        0:
          id: theDictator_Scene_0_Element_1
          type: image
          src: https://cdn.leonardo.ai/users/eea1e02c-be1d-4b0d-b5e8-5e34e8909fcd/generations/6891406c-a877-449d-aaa7-ff37e0d7bc5d/Leonardo_Phoenix_09_A_clumsy_soldier_in_a_warrior_environment_0.jpg
          scale:
            width: 1080
            height: 1920
            x: 0
            y: 0
            position: center-center
            zoom: 5
      }
    }
}
```

Generate Video Parameters Settings Docs

Method: POST

URL: https://api.json2video.com/v2/movies

Authentication: None

Send Query Parameters: ☐

Send Headers: ☒

Specify Headers: Using Fields Below

Header Parameters:

Name	Value
x-api-key	dxrgBt8NUpYVDF76HCQpEBw78TFOL2he3PZ5f7
Content-Type	application/json

OUTPUT

1 item

success	project	timestamp
true	7afQZegE7bQ1mUdr	2025-03-10T10:25:17.905Z

INPUT

```
{} Prepare video JSON 1 item
```

json2video

```
{
  id: theDictator
  comment: Default movie
  width: 1080
  height: 1920
  quality: medium
  draft: true
  scenes:
    0:
      id: theDictator_Scene_0
      comment: Scene 0
      elements:
        0:
          id: theDictator_Scene_0_Element_1
          type: image
          src: https://cdn.leonardo.ai/users/eea1e02c-be1d-4b0d-b5e8-5e34e8909fcd/generations/6891406c-a877-449d-aaa7-ff37e0d7bc5d/Leonardo_Phoenix_09_A_clumsy_soldier_in_a_warrior_environment_0.jpg
          scale:
            width: 1080
            height: 1920
            x: 0
            y: 0
            position: center-center
            zoom: 5
      }
    }
}
```

Generate Video Parameters Settings Docs

Value: dxrgBt8NUpYVDF76HCQpEBw78TFOL2he3PZ5f7

Name: Content-Type

Value: application/json

Add Parameter

Send Body: ☒

Body Content Type: JSON

Specify Body: Using JSON

JSON:

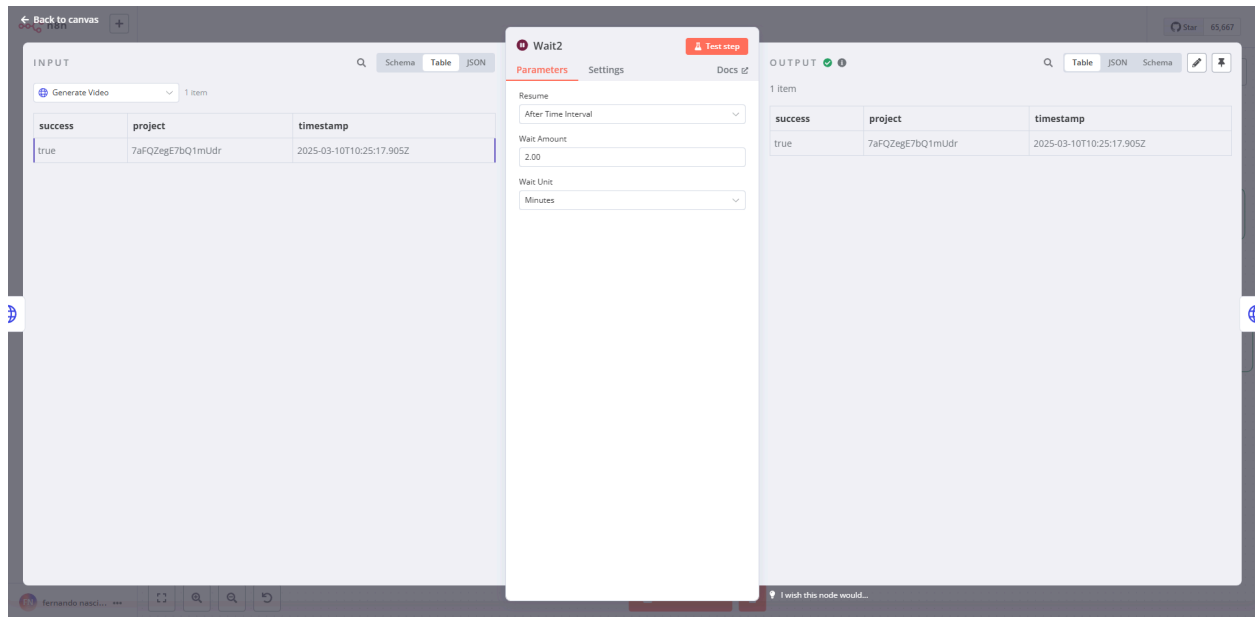
```
{ "id": "theDictator", "comment": "Default movie", ... }
```

OUTPUT

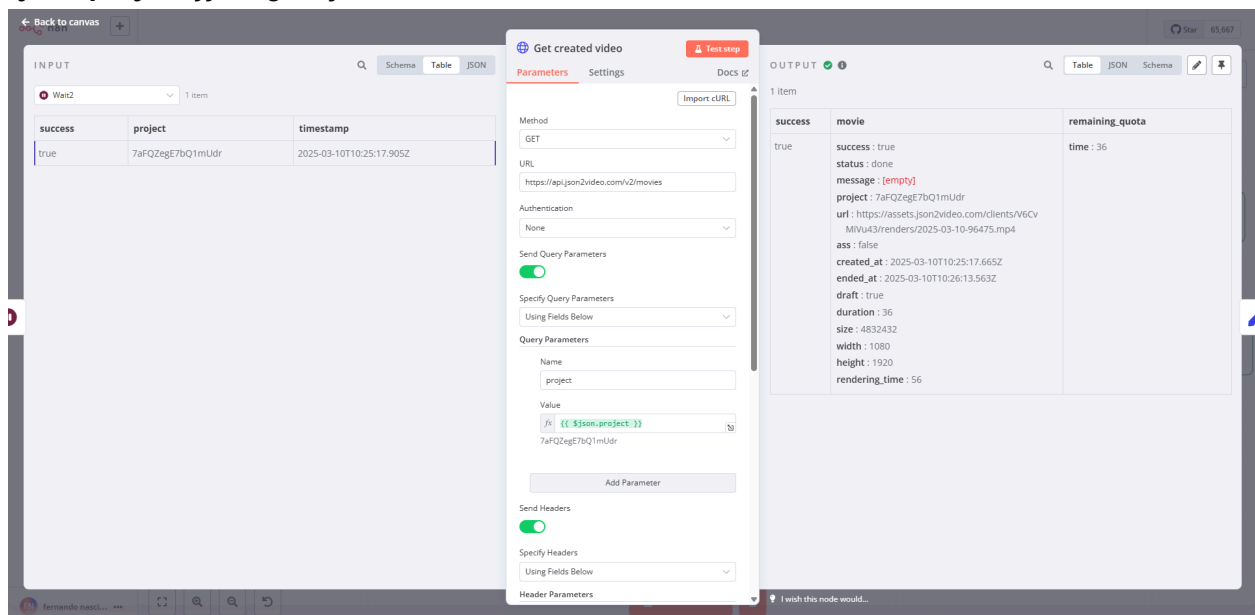
1 item

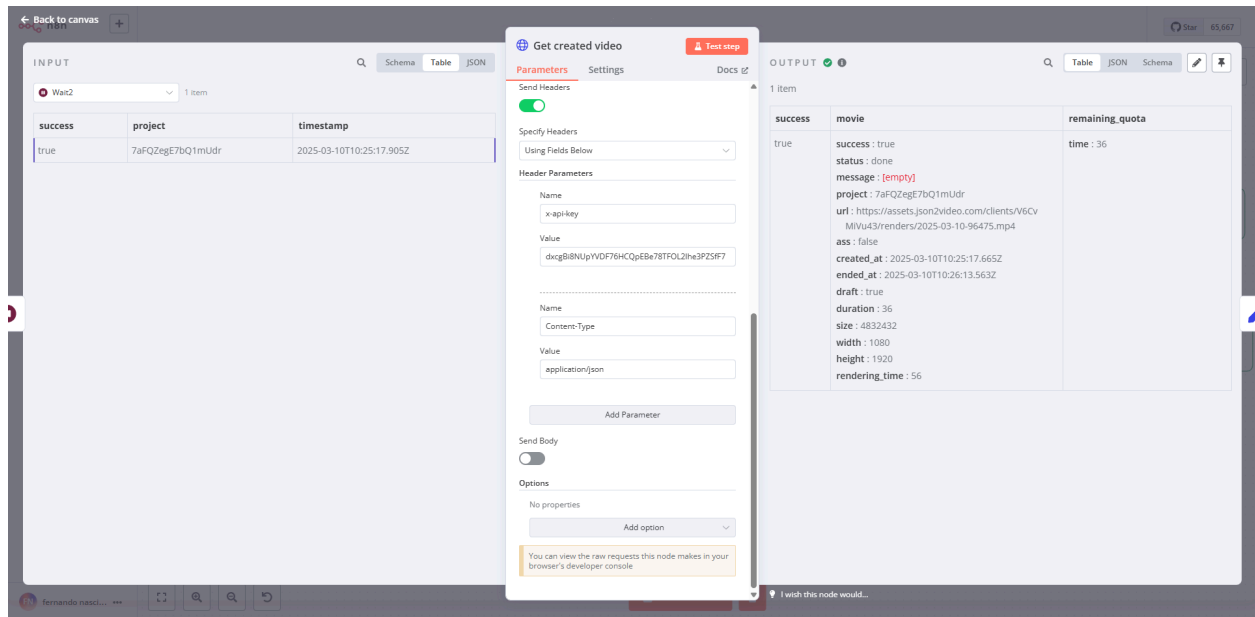
success	project	timestamp
true	7afQZegE7bQ1mUdr	2025-03-10T10:25:17.905Z

Adicionamos um novo node **Wait** e configuramos ele para esperar **2** minutos pois a geração de vídeo costuma requerer um tempo maior para o processamento.

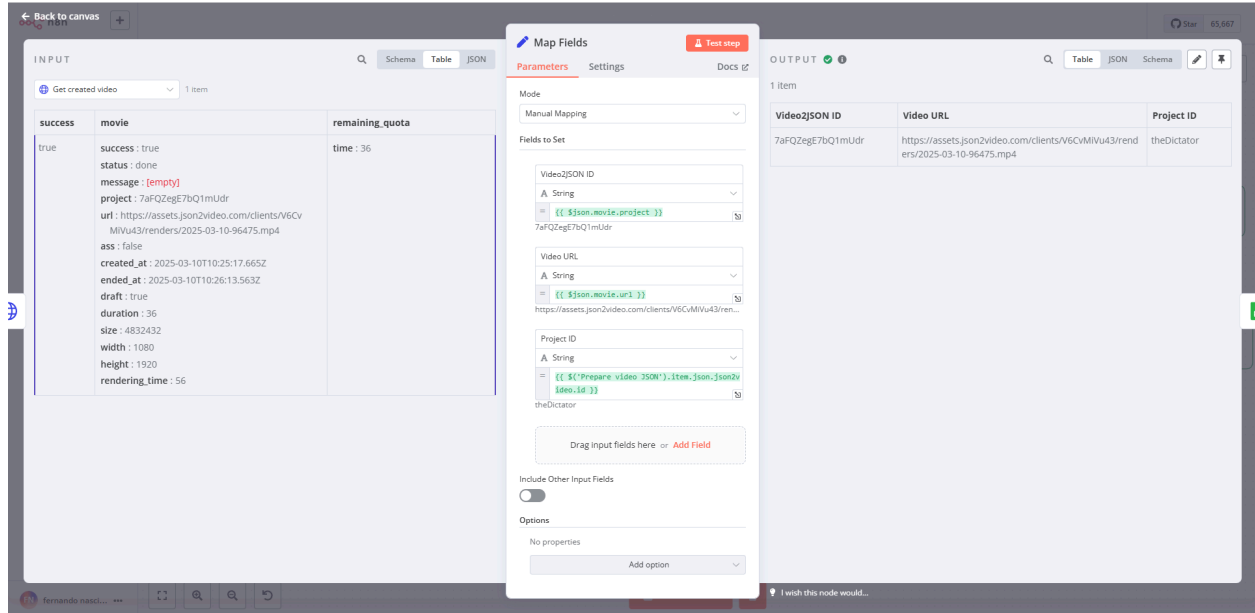


Uma vez criado o vídeo precisamos fazer um novo http request para acessar a URL do vídeo gerado. Para tanto temos que incluir um novo node de **HTTP**, então basta copiar e colar o node **HTTP** usado para a **geração** do vídeo e fazer algumas pequenas modificações, como: trocar **POST** por **GET**; em **Send Query** colocar um parâmetro chamado **project** referenciando **{{ \$json.project }}** da geração do video; e **Send Headers** fica a mesma coisa.





Com a url do vídeo “em mãos”, basta-nos agora pegar as informações que achamos essenciais e mapear em mais um node **Edit Fields**, só por questões de simplicidade, ou filtragem se acharem melhor. Nesse caso estão sendo mapeados 3 campos: o projeto do vídeo, a url do vídeo e o id do vídeo dentro da api do Json2video.



E finalmente salvamos essas informações numa planilha no nosso Drive. Então adicionamos um novo node Drive-Sheets com ação de **append Row**. Selecionamos nossa **credencial**, colocamos o valor **By Id**, selecionamos **by name** e informamos o nome da planilha que temos no Drive. E concluindo mapeamos as informações do vídeo com as colunas da planilha.

E chegamos ao final do tutorial. Espero que tenham gostado. De um Like e comente dando sugestões, críticas, elogios, etc..