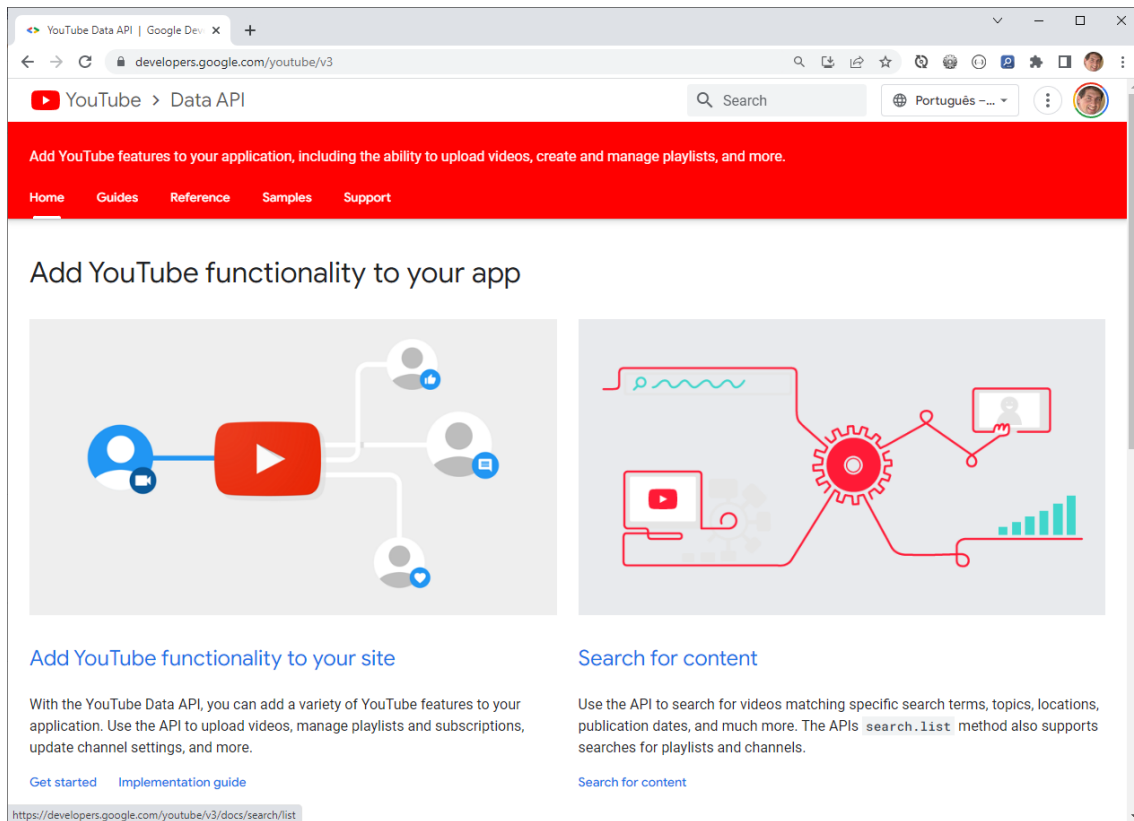


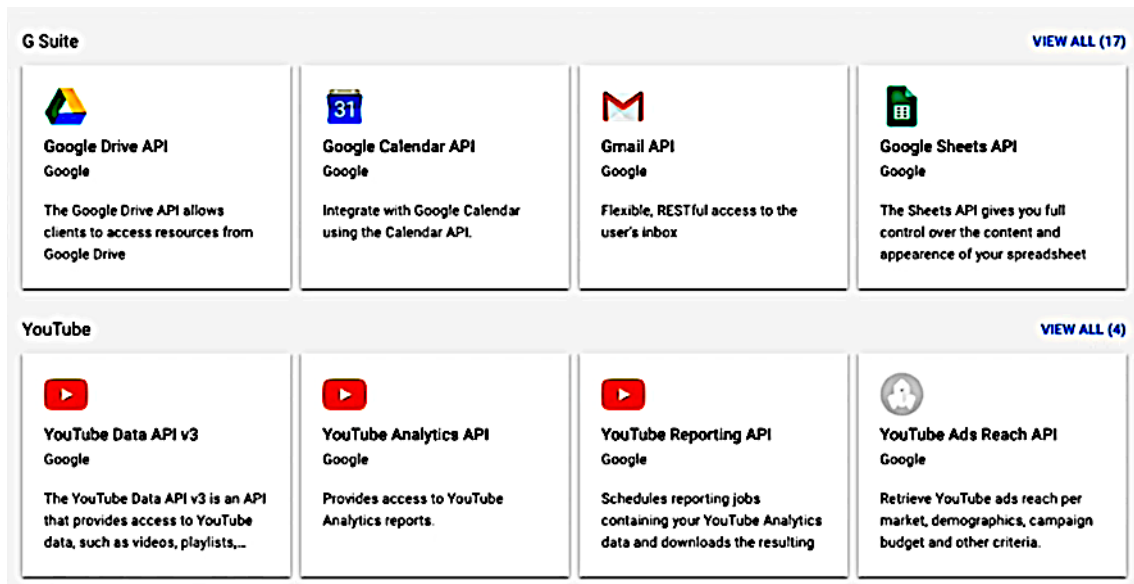
Lista 15 – API do YouTube

Utilização da API do YouTube.



Para utilizar a API do YouTube será necessário seguir os seguintes passos:

- Criar uma credencial de acesso à API de desenvolvedores: Isso é feito no *site* da Google.
- Habilitar qual ou quais serviços estarão vinculados à esta credencial: Mais de um serviço (API) está disponível.



- Criar a chave de acesso: A chave identifica a qual desenvolvedor pertence a aplicação criada.
- Definir qual plataforma será utilizada: Cliente/navegador (JavaScript), servidor (Node.js/Tomcat), Android, iOS, etc.
- Definir quais dados serão acessados (público ou privado): Dados privados, como relatórios de acesso, necessitam de aprovação do proprietário do canal.
- Definir se poderá funcionar em qualquer URL ou não: É possível liberar acesso para qualquer URL ou então especificar quais URLs poderão rodar. Normalmente uma URL local (localhost) é utilizada em desenvolvimento e um domínio na internet é utilizado em produção.

Criando as credenciais. A página [abaixo](#) fornece um passo a passo com as explicações.

The screenshot shows the 'Getting Started with the YouTube Data API' page on the YouTube Developers site. The main heading is 'Getting Started with the YouTube Data API'. Below it, the 'Introdução' (Introduction) section explains that the document is for developers who want to create applications that interact with the YouTube API. It also mentions that the document provides a general overview of the different functions compatible with the API. The 'Antes de começar' (Before getting started) section lists three steps: 1. You need a Google account to access the Google API Console, request an API key, and register your application. 2. Register your application on Google so it can send API requests. 3. After registering your application, select an API from the YouTube data services used by your application. The steps are numbered 1 through 4, with sub-points i, ii, and iii under step 3. A video thumbnail titled 'Obtaining a developer key for the...' is shown. The right sidebar contains a table of contents for the page, including 'Introdução', 'Antes de começar', 'Recursos e tipos de recursos', 'Operações compatíveis', 'Uso de quotas', 'Recursos parciais', 'Compreender o parâmetro part', 'Compreender o parâmetro fields', 'Exemplos de solicitações parciais', and 'Recommended for you'.

Operações disponíveis:

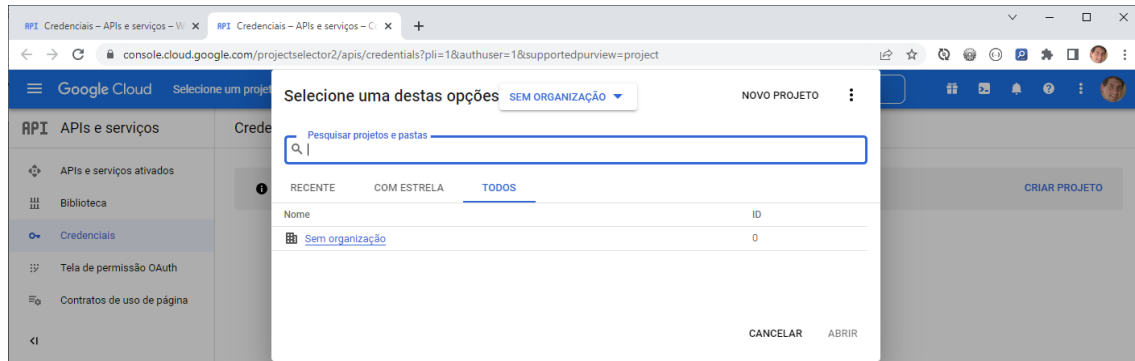
The screenshot shows the 'Getting Started with the YouTube Data API' page on the YouTube Developers site, specifically the 'Operações compatíveis' (Compatible operations) section. The text explains that the table below shows the most common methods compatible with the API. Some resources also have specific functions, such as the 'videos.rate' method, which associates a video with a user, and the 'thumbnails.set' method, which sends a video thumbnail to YouTube. A table titled 'Operações' (Operations) lists four methods: 'list' (Recupera (GET) uma lista vazia ou com recursos.), 'insert' (Cria (POST) um novo recurso.), 'update' (Modifica (PUT) um recurso existente para propagar os dados em sua solicitação.), and 'delete' (Remove (DELETE) um recurso específico.). The right sidebar contains a table of contents for the page, including 'Introdução', 'Antes de começar', 'Recursos e tipos de recursos', 'Operações compatíveis', 'Uso de quotas', 'Recursos parciais', 'Compreender o parâmetro part', 'Compreender o parâmetro fields', 'Exemplos de solicitações parciais', and 'Recommended for you'.

É necessário ter uma conta no Google. Caso não tenha clique [aqui](#) para criar uma.

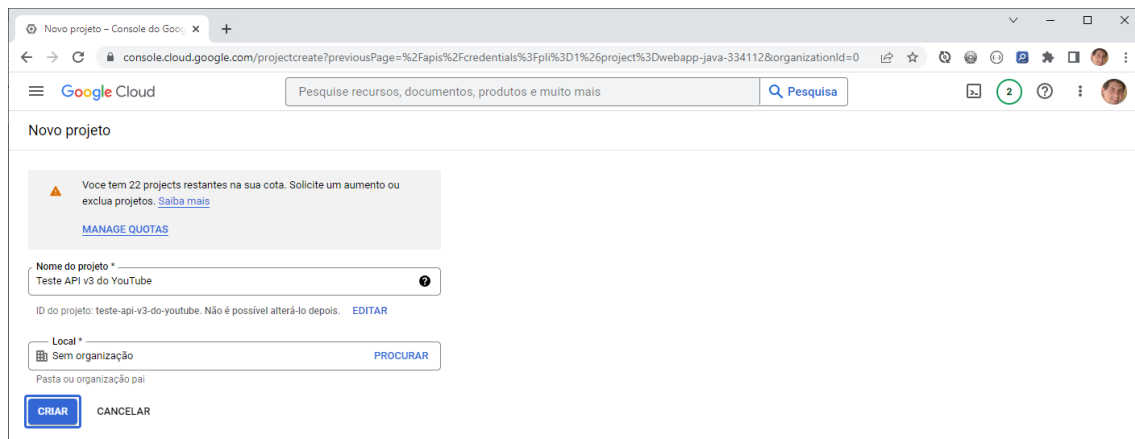
A página [abaixo](#) mostra as informações de como conseguir as credenciais.

The screenshot shows the 'Getting Started with the YouTube Data API' page on the YouTube Developers site, specifically the 'Conseguir credenciais de autorização' (Getting authorization credentials) section. The text explains that to use the YouTube Data API, your application must have authorization credentials. The document describes the different types of authorization credentials that the Google API Console supports. It also explains how to locate or create authorization credentials for your project. The 'Crie seu projeto e selecione serviços de API' (Create your project and select API services) section lists two steps: 1. Open the 'Credentials' page in the API Console. 2. The API is compatible with two types of credentials. Create the appropriate credentials for your project. The steps are numbered 1 through 2, with sub-points i, ii, and iii under step 2. A video thumbnail titled 'Obtaining a developer key for the...' is shown. The right sidebar contains a table of contents for the page, including 'Introdução', 'Antes de começar', 'Recursos e tipos de recursos', 'Operações compatíveis', 'Uso de quotas', 'Recursos parciais', 'Compreender o parâmetro part', 'Compreender o parâmetro fields', 'Exemplos de solicitações parciais', and 'Recommended for you'.

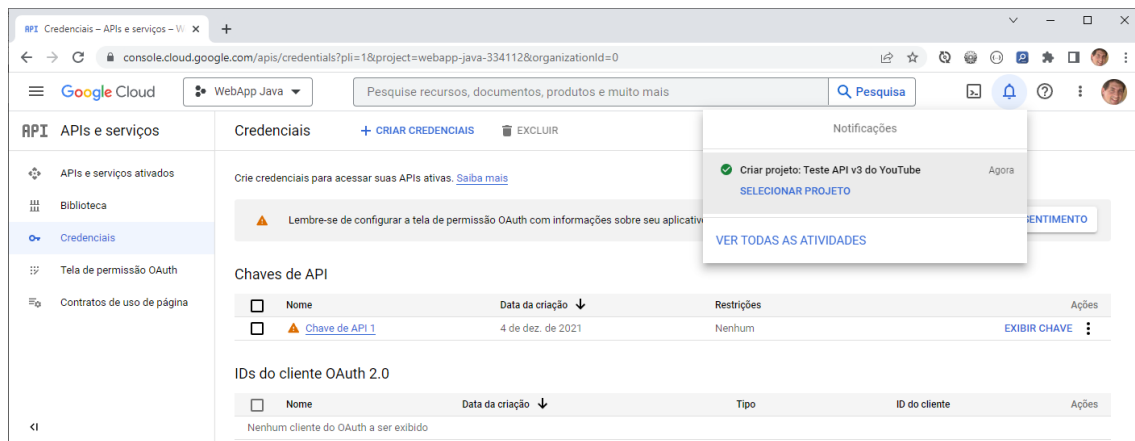
Clique [aqui](#) para abrir a página de credenciais. Clique no botão *Selecione um projeto* e então em *Novo projeto*.



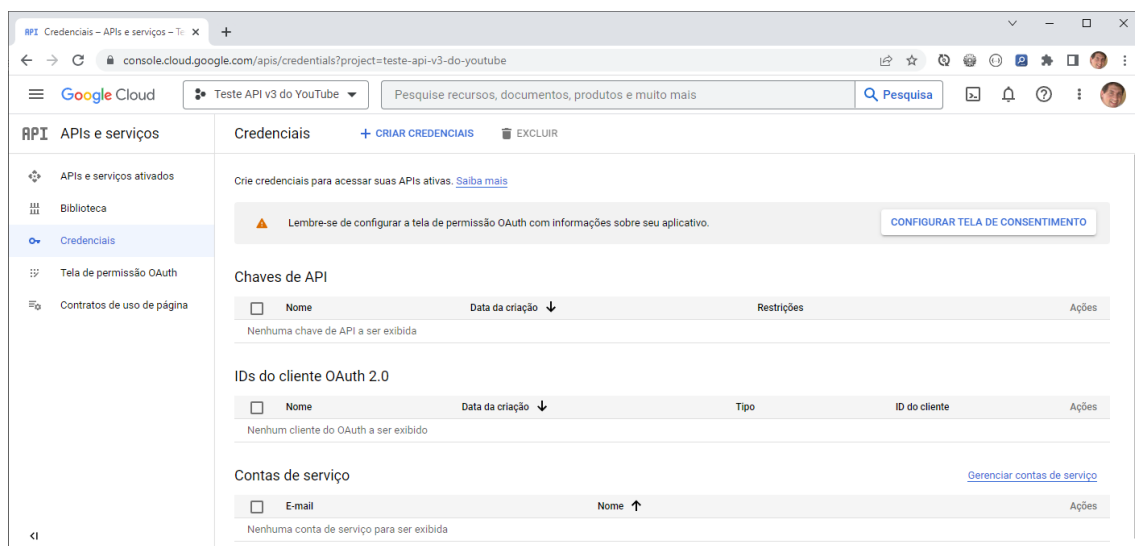
Insira os dados como mostrado abaixo e clique no botão *Criar*.



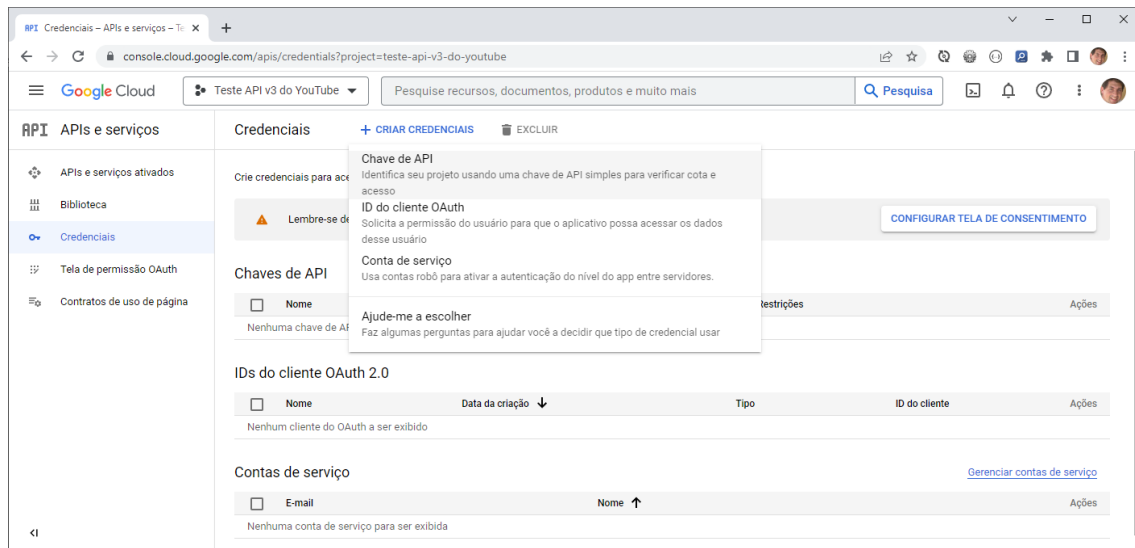
Clique no botão *Selecionar projeto*.




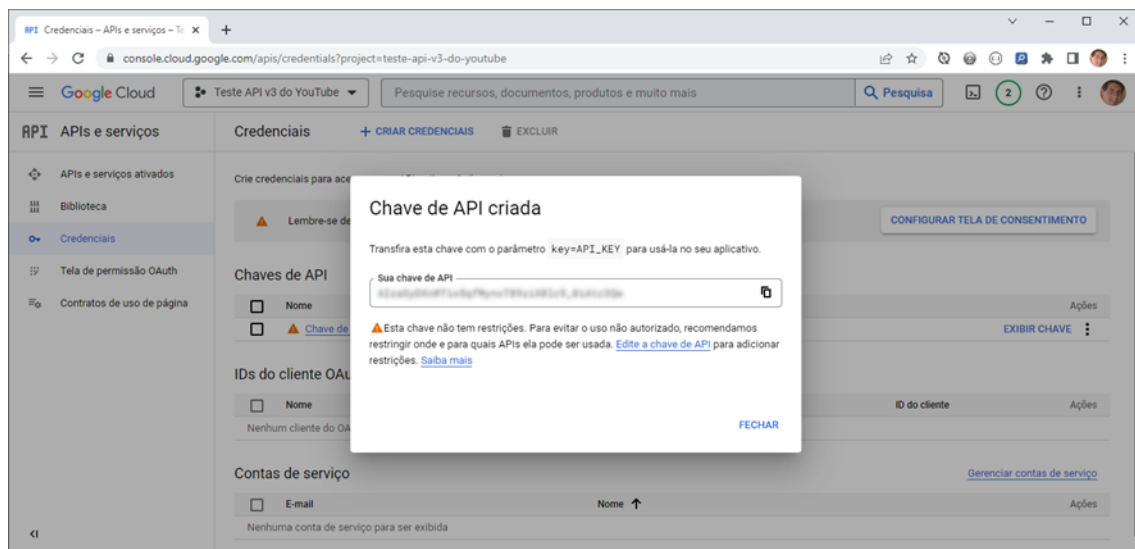
O nome do projeto irá aparecer no canto superior esquerdo na janela.



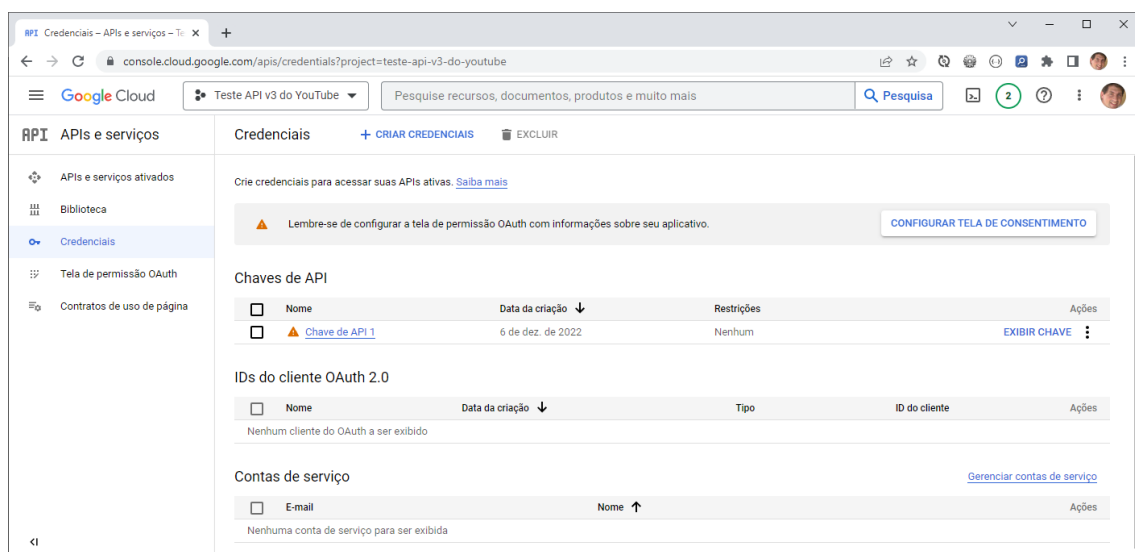
A seguir escolha o menu *Credenciais* do lado esquerdo e clique no botão *Criar credenciais* e então *Chave de API*.



A chave será criada. Clique no botão  para copiar a chave para a área de transferência. Cole a chave em algum lugar, como no bloco de notas, pois será preciso inseri-la mais tarde no código fonte.



Clique no *link* abaixo (Chave de API 1) para configurar a chave.



No campo *Restrições de aplicativo* escolha *Nenhum* e clique no botão *Salvar*.

The screenshot shows the 'Editar chave de API' page in the Google Cloud console. The left sidebar has 'APIs e serviços' expanded, with 'Credenciais' selected. The main content area is titled 'Editar chave de API' and includes a 'GERAR CHAVE NOVAMENTE' button and an 'EXCLUIR' button. The 'Nome' field is 'Chave de API 1'. The 'API Key' field contains a long alphanumeric string. The 'Restrições de chave' section has a warning icon and text. The 'Restrições de aplicativo' section has radio buttons for 'Nenhum' (selected), 'Referenciadores HTTP (sites da Web)', 'Endereços IP (servidores da Web, cron jobs etc.)', 'Apps Android', and 'Apps iOS'. The 'Restrições da API' section has radio buttons for 'Não restringir a chave' (selected) and 'Restringir chave'. The 'Salvar' button is highlighted in blue.

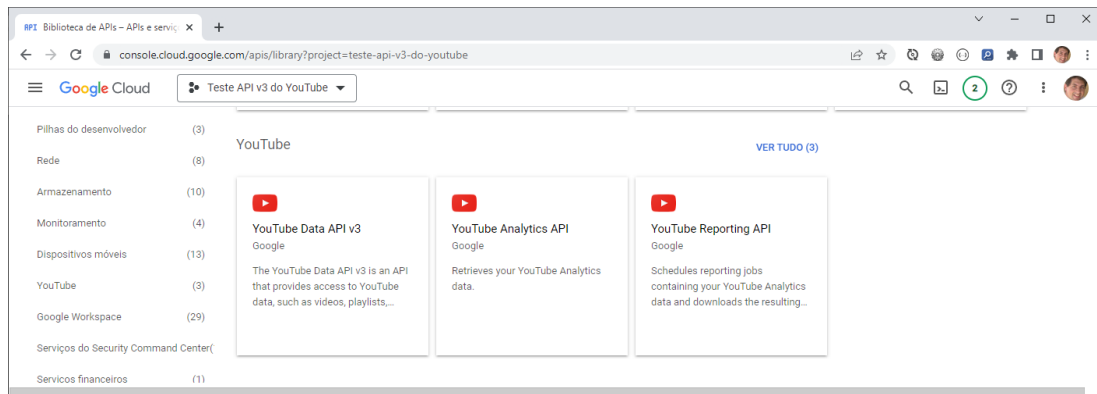
Para habilitar a API do YouTube para o aplicativo clique agora no item *Biblioteca* no menu lateral.

The screenshot shows the 'Credenciais' page in the Google Cloud console. The left sidebar has 'APIs e serviços' expanded, with 'Credenciais' selected. The main content area is titled 'Credenciais' and includes a 'CRIAR CREDENCIAIS' button and an 'EXCLUIR' button. The 'Chaves de API' section is visible, showing a table with one API key. The 'Biblioteca' item is highlighted in the left sidebar.

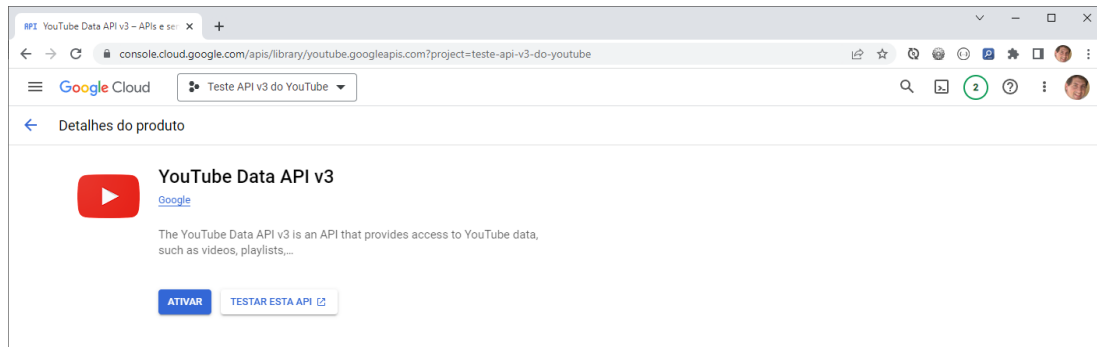
Você será levado à página de biblioteca de APIs.

The screenshot shows the 'Biblioteca de APIs' page in the Google Cloud console. The page displays a search bar and a list of APIs. The 'Filtro' section on the left has 'Visibilidade' set to 'Público' and 'Categoria' set to 'Mapas'. The 'Mapas' section shows four API cards: 'Maps SDK for Android', 'Maps SDK for iOS', 'Maps JavaScript API', and 'Places API'. The 'Biblioteca de APIs' title is at the top, and the 'VER TUDO (17)' link is on the right.

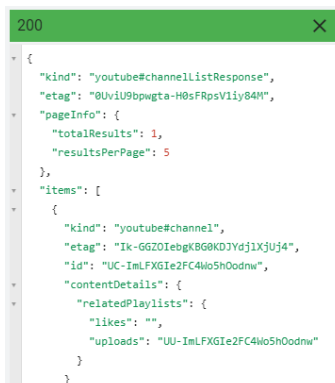
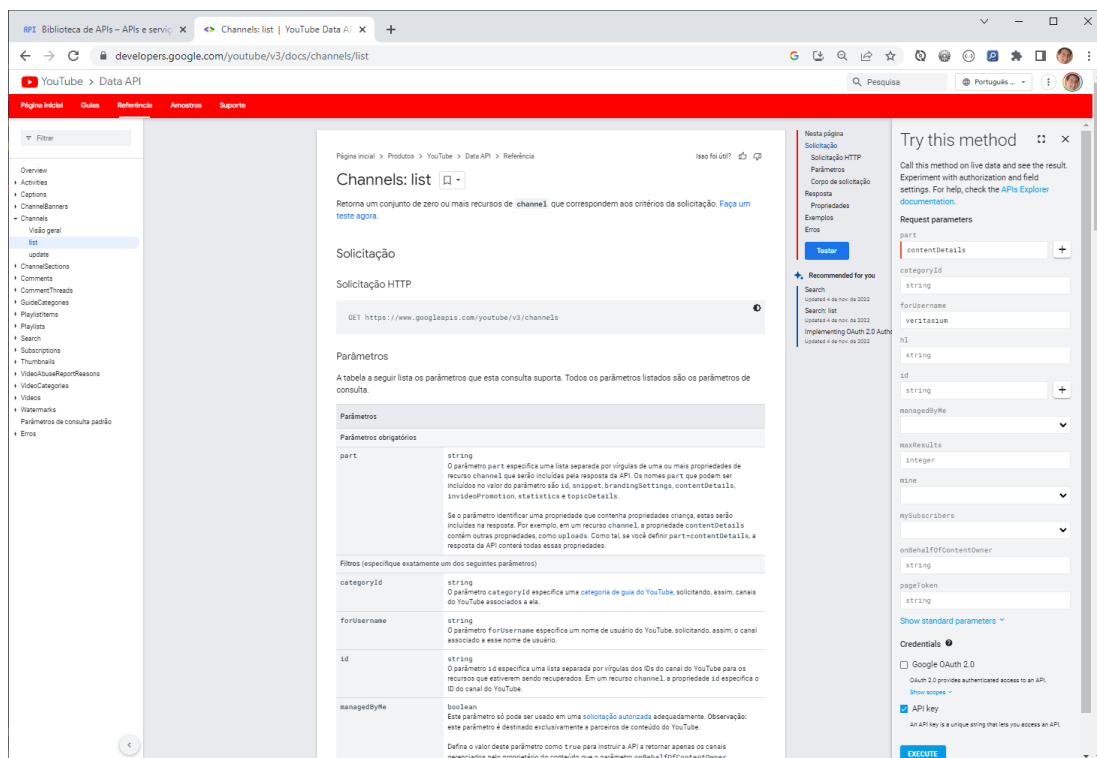
Role a página até que apareça a opção YouTube Data API v3. Clique nela.



Clique no botão *Ativar*.



A página abaixo mostra a solicitação HTTP para obter uma lista dos canais. É possível testar o método na barra lateral.



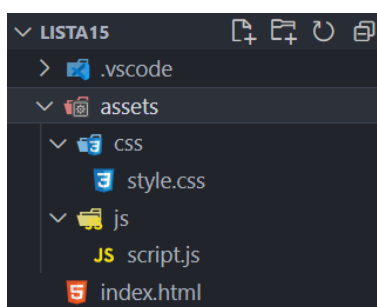
Crie uma pasta chamada *lista15* e dentro dela uma pasta chamada *assets*. Dentro da pasta *assets* crie outras duas pastas: *css* e *js*.

O arquivo *index.html* deverá ficar diretamente dentro da pasta *lista15*.

```
lista15 - index.html
1  <!DOCTYPE html>
2  <html lang="pt-BR">
3
4  <head>
5      <meta charset="UTF-8">
6      <meta http-equiv="X-UA-Compatible" content="IE=edge">
7      <meta name="viewport" content="width=device-width, initial-scale=1.0">
8      <title>YouTube API v3</title>
9
10     <!-- Arquivos/bibliotecas CSS -->
11     <link rel="stylesheet" href="./assets/css/style.css">
12
13     <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/@fancyapps/ui@4.0/dist/fancybox.css" />
14
15     <!-- Arquivos/bibliotecas JavaScript -->
16     <script src="https://code.jquery.com/jquery-3.6.1.min.js" integrity="sha256-o88AwQnZB+VDvE9tvIXrM
QaPlFFSUTR+nldQm1LuPXQ=" crossorigin="anonymous"></script>
17
18     <script src="https://momentjs.com/downloads/moment-with-locales.min.js"></script>
19
20     <script src="https://cdn.jsdelivr.net/npm/@fancyapps/ui@4.0/dist/fancybox.umd.js"></script>
21
22     <script src="./assets/js/script.js"></script>
23 </head>
24
25 <body>
26     <div id="janela">
27         <ul></ul>
28     </div>
29 </body>
30
31 </html>
```

Crie o arquivo *script.js* dentro da pasta *assets/js*.

A organização final das pastas deverá ser a seguinte:



Não se preocupe com a pasta *.vscode*, ela é criada automaticamente dependendo as extensões instaladas.

O arquivo *style.css* deverá ficar diretamente dentro da pasta *assets/css*.

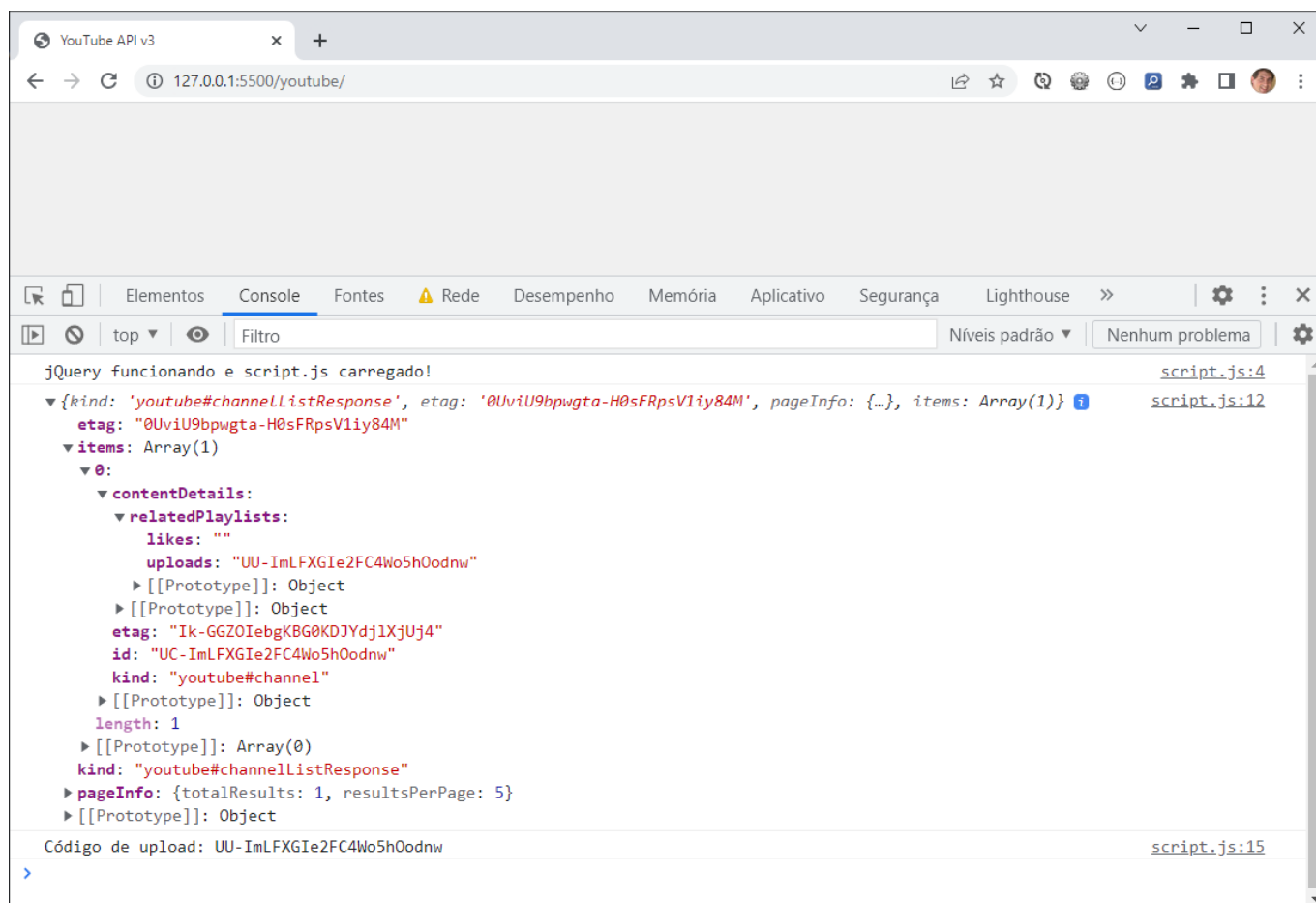
```
lista15 - style.css

1  body {
2      background: #F0F0F0;
3  }
4
5  div#janela ul {
6      width: 1080px;
7      margin: 0 auto;
8      padding: 0;
9  }
10
11 div#janela ul li.principal {
12     position: relative;
13     list-style: none;
14     display: inline-block;
15     margin: 20px;
16     cursor: pointer;
17     width: 320px;
18     height: 180px;
19 }
20
21 div#janela ul li.principal div.foto {
22     position: absolute;
23     top: 0;
24     left: 0;
25     width: 320px;
26     height: 180px;
27     z-index: 1;
28     clip: rect(0 320px 180px 0);
29 }
30
31 div#janela ul li.principal div.foto div.legenda {
32     position: absolute;
33     top: 160px;
34     left: 0;
35     width: 320px;
36     height: 150px;
37     padding: 5px;
38     z-index: 2;
39     cursor: pointer;
40     background-color: rgba(0, 0, 0, 0.5);
41 }
42
43 div#janela ul li.principal div.foto div.legenda:hover {
44     top: 60px;
45     transition: all 0.5s ease-out 0s;
46 }
47
48 div#janela ul li.principal div.foto div.legenda h5 {
49     font-family: sans-serif;
50     font-size: small;
51     text-transform: uppercase;
52     font-weight: 100;
53     color: aqua;
54     margin: 0;
55 }
56
57 div#janela ul li.principal div.foto div.legenda p {
58     font-family: sans-serif;
59     font-size: 0.7rem;
60     color: aliceblue;
61 }
```


O jQuery será utilizado para fazer uma solicitação GET no endereço <https://www.googleapis.com/youtube/v3/channels> com o parâmetro `forUsername` configurado para o canal oficial da [Unesco](#) ('unescooficial'). Na linha 2, dentro das aspas simples, cole a sua chave de API criada na página 4 deste roteiro.

```
lista16 - script.js
1 let nomeCanal = 'unescooficial';
2 let chave = 'AIzaSyDm8T1u8qPqWt7BwU188U8_808w8q';
3
4 $(document).ready(function() {
5     console.log('jQuery funcionando e script.js carregado!');
6
7     $.get('https://www.googleapis.com/youtube/v3/channels', {
8         part: 'contentDetails',
9         forUsername: nomeCanal,
10        key: chave
11    },
12    dados => {
13        let codigoUpload = dados.items[0].contentDetails.relatedPlaylists.uploads;
14        console.log(dados, 'Código de upload:', codigoUpload);
15
16        pegarVideos(codigoUpload);
17    })
18 });
19 });
```

Resultado mostrando o código de *upload*.

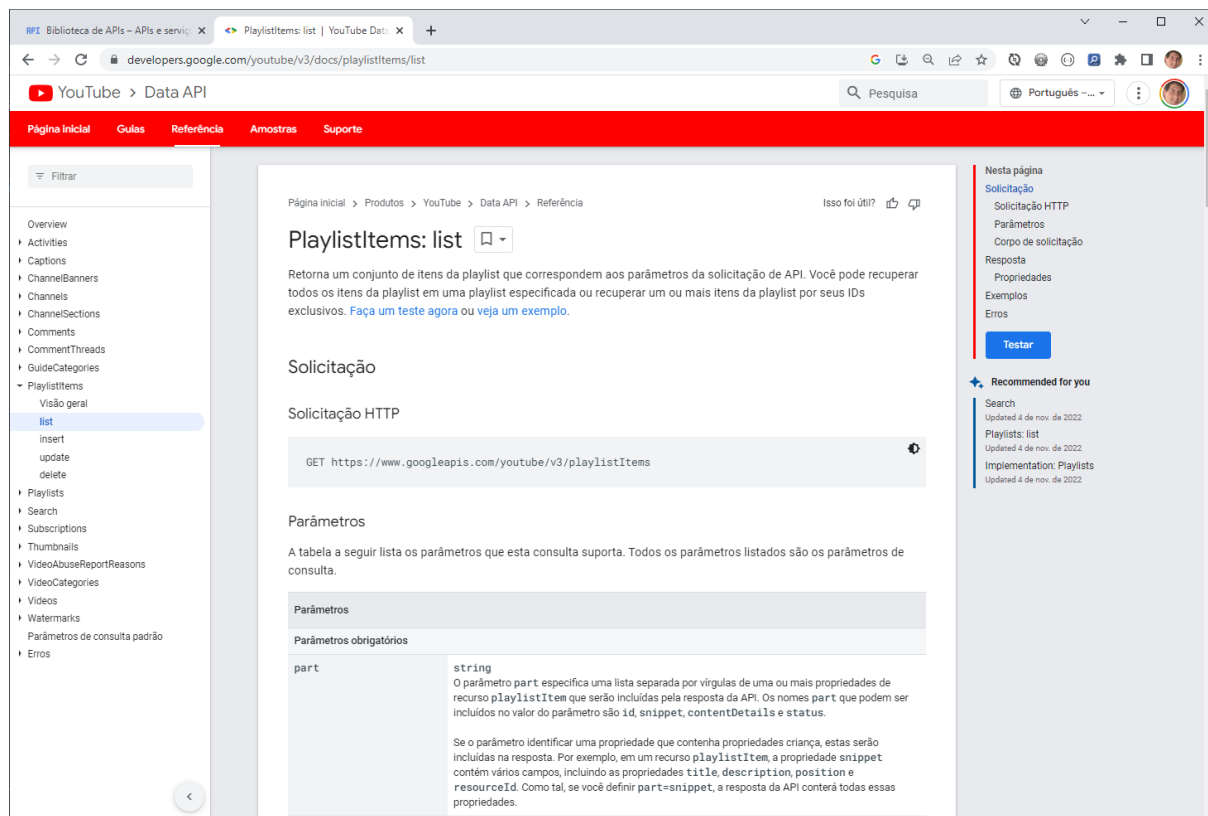


The screenshot shows a web browser window with the address bar displaying '127.0.0.1:5500/youtube/'. The browser's developer tools are open, showing the 'Console' tab. The console displays the following log messages:

- jQuery funcionando e script.js carregado! (script.js:4)
- A JSON object representing the YouTube API response (script.js:12). The object has the following structure:
 - `kind`: 'youtube#channelListResponse'
 - `etag`: '0UviU9bpwgta-H0sFRpsV1iy84M'
 - `pageInfo`: {totalResults: 1, resultsPerPage: 5}
 - `items`: Array(1)
 - `0`: {
 - `contentDetails`: {
 - `relatedPlaylists`: {
 - `likes`: ''
 - `uploads`: 'UU-ImLFXGie2FC4Wo5h0odnw'

The console also shows the log message 'Código de upload: UU-ImLFXGie2FC4Wo5h0odnw' (script.js:15).

Abaixo o método para retornar os itens de uma *playlist*.



The screenshot shows the YouTube Data API documentation page for the 'PlaylistItems: list' endpoint. The page is in Portuguese and includes a sidebar with a navigation menu. The main content area is titled 'PlaylistItems: list' and contains the following sections:

- Solicitação**: Returns a set of items from the playlist that correspond to the API request parameters. You can recover all items from the playlist in a specific playlist or recover one or more items from the playlist by their IDs exclusively. [Faça um teste agora](#) ou [veja um exemplo](#).
- Solicitação HTTP**: GET `https://www.googleapis.com/youtube/v3/playlistItems`
- Parâmetros**: A table listing the parameters supported by this query. All listed parameters are query parameters.

Parâmetros obrigatórios	
part	string O parâmetro part especifica uma lista separada por vírgulas de uma ou mais propriedades de recurso <code>playlistItem</code> que serão incluídas pela resposta da API. Os nomes part que podem ser incluídos no valor do parâmetro são <code>id</code> , <code>snippet</code> , <code>contentDetails</code> e <code>status</code> . Se o parâmetro identificar uma propriedade que contenha propriedades criancas, estas serão incluídas na resposta. Por exemplo, em um recurso <code>playlistItem</code> , a propriedade <code>snippet</code> contém vários campos, incluindo as propriedades <code>title</code> , <code>description</code> , <code>position</code> e <code>resourceId</code> . Como tal, se você definir <code>part=snippet</code> , a resposta da API conterá todas essas propriedades.

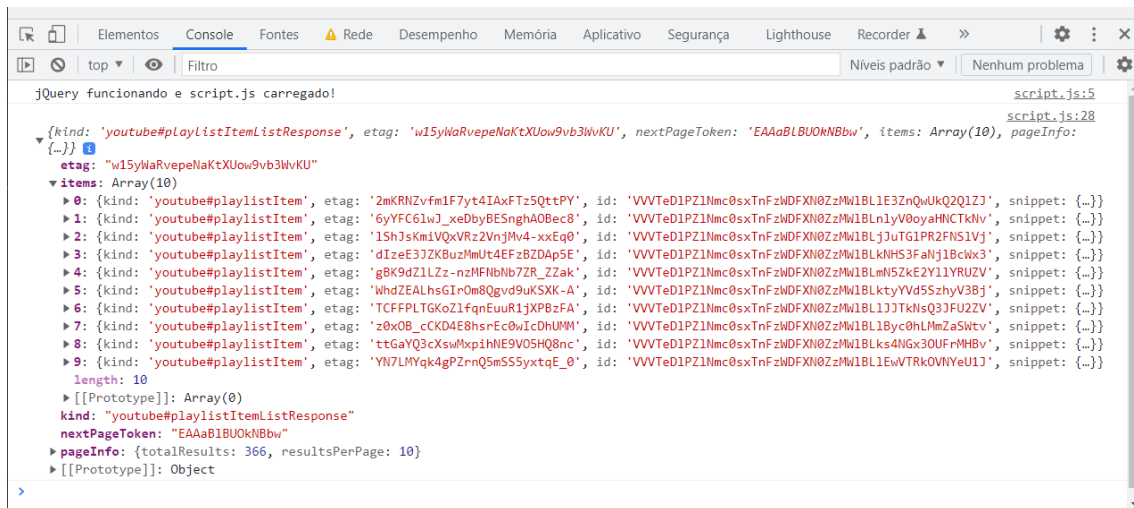
O próximo código irá obter a lista de itens da *playlist*.

As linhas 7 e 21 utilizam o método `$.get()` do jQuery (semelhante às requisições, Ajax, Fetch API ou Axios) para fazer uma requisição do tipo `GET` à API.

```
lista16 - script.js

1 let nomeCanal = 'unoescoficial';
2 let chave = 'AIzaSyDXnRTioSqfMynxT89ziX8lc9_0iAtc3Qw';
3
4 $(document).ready(function() {
5     console.log('jQuery funcionando e script.js carregado!');
6
7     $.get('https://www.googleapis.com/youtube/v3/channels', {
8         part: 'contentDetails',
9         forUsername: nomeCanal,
10        key: chave
11    },
12    dados => {
13        let codigoUpload = dados.items[0].contentDetails.relatedPlaylists.uploads;
14        // console.log(dados, 'Código de upload:', codigoUpload);
15
16        pegarVideos(codigoUpload);
17    }
18    );
19
20    function pegarVideos(codigo) {
21        $.get('https://www.googleapis.com/youtube/v3/playlistItems', {
22            part: 'snippet',
23            maxResults: 10,
24            playlistId: codigo,
25            key: chave
26        },
27        dados => {
28            console.log(dados);
29        }
30        );
31    }
32 });
```

Resultados:



Modifique o código abaixo para exibir os *thumbnails* dos últimos 12 vídeos.

A linha 30 utiliza o método `$.each()` do jQuery, que é equivalente ao método `forEach()` do JavaScript. O método `append()` da linha 40 é equivalente ao método `appendChild()` do DOM.

```
20 function pegarVideos(codigo) {
21     $.get('https://www.googleapis.com/youtube/v3/playlistItems', {
22         part: 'snippet',
23         maxResults: 12,
24         playlistId: codigo,
25         key: chave
26     },
27     dados => {
28         // console.Log(dados);
29
30         $.each(dados.items, (_, item) => {
31             let imagem, arquivo;
32
33             imagem = item.snippet.thumbnails.medium.url;
34
35             arquivo = `<li>
36                 
37             </li>`;
38
39             // console.Log(arquivo);
40             $('#janela ul').append(arquivo);
41         });
42     });
43 }
44 }
```

Melhorando a visualização e aplicando efeito de animação ao passar o *mouse* por cima da miniatura. O título, descrição e data de publicação do vídeo são mostrados.

A biblioteca [Moment.js](https://momentjs.com/) está sendo carregada no arquivo *index.html*, na linha 18.

```
10 <!-- Arquivos/bibliotecas CSS -->
11 <link rel="stylesheet" href="./assets/css/style.css">
12
13 <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/@fancyapps/ui@4.0/dist/fancybox.css" />
14
15 <!-- Arquivos/bibliotecas JavaScript -->
16 <script src="https://code.jquery.com/jquery-3.6.1.min.js" integrity="sha256-o88AwQnZB+VDvE9tIXrM
17 QaPlFFSUTR+nldQm1LuPXQ=" crossorigin="anonymous"></script>
18
19 <script src="https://momentjs.com/downloads/moment-with-locales.min.js"></script>
20
21 <script src="https://cdn.jsdelivr.net/npm/@fancyapps/ui@4.0/dist/fancybox.umd.js"></script>
22
23 <script src="./assets/js/script.js"></script>
24 </head>
```

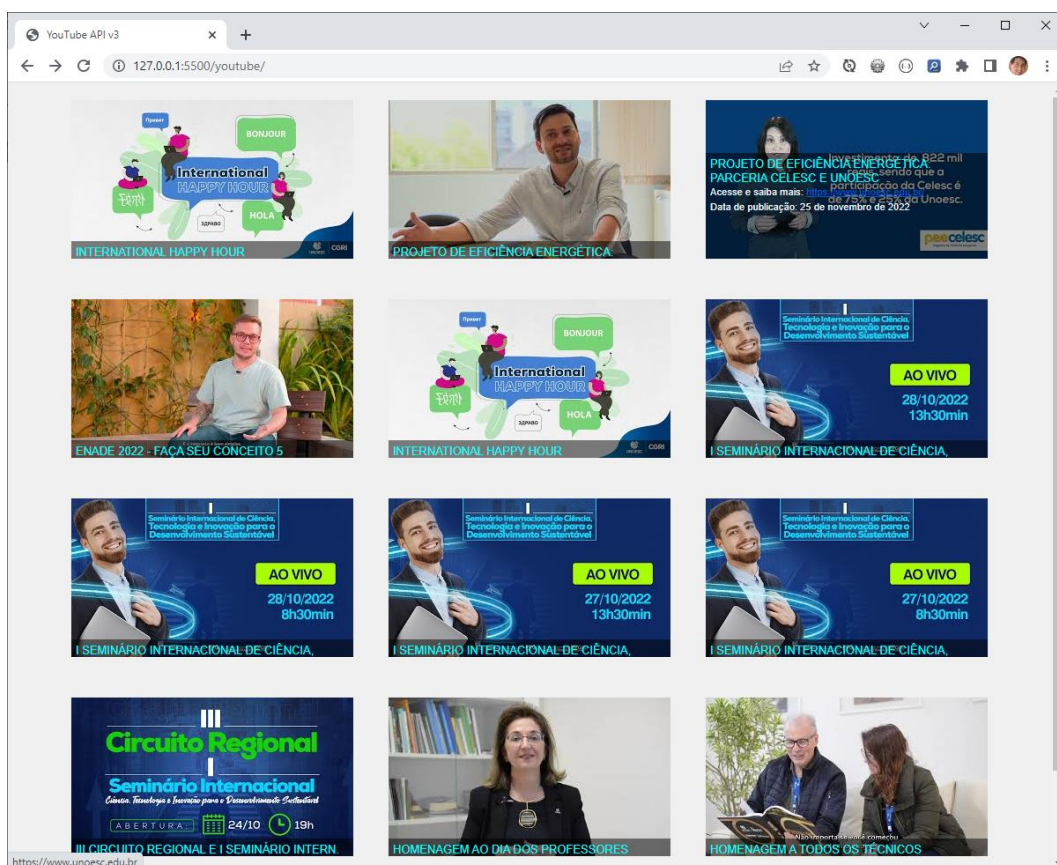
Modificações no arquivo `script.js`. São acrescentadas as variáveis para armazenar a data de publicação e a descrição (linha 30). As linhas 35 e 36 usam as funções da biblioteca Moment.js é utilizada para o tratamento das datas. As linhas 39-40 utilizam uma expressão regular ([regex](#)) para identificar uma URL dentro da descrição e transformá-la em um hiperlink. As linhas 42-54 adicionam a legenda e a descrição em cada miniatura.

```

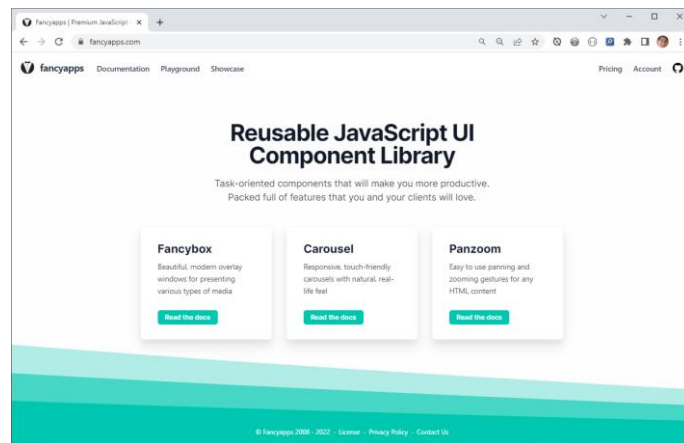
20 function pegarVideos(codigo) {
21   $.get('https://www.googleapis.com/youtube/v3/playlistItems', {
22     part: 'snippet',
23     maxResults: 12,
24     playlistId: codigo,
25     key: chave
26   },
27   dados => {
28     // console.log(dados);
29     $.each(dados.items, (_, item) => {
30       let imagem, arquivo, dataPublicacao, descricao;
31
32       imagem = item.snippet.thumbnails.medium.url;
33       titulo = item.snippet.title;
34
35       moment.locale('pt');
36       dataPublicacao = new moment(item.snippet.publishedAt).format('DD [de] MMMM [de] YYYY');
37
38       descricao = item.snippet.description;
39       descricao = descricao.replace(/((https?|file|ftp):\/\/[^\s?=&.\|/;#~%-]+(?:![^\s?=&.\|/;#~%]*=)))/g,
40       | | '<a href="$1" target="_blank">$1</a>');
41
42       arquivo = `<li class="principal">
43         <div class="foto">
44           
45           <div class="legenda">
46             <h5>${titulo}</h5>
47             <p>
48               ${descricao}
49               <br>
50               Data de publicação: ${dataPublicacao}
51             </p>
52           </div>
53         </div>
54       </li>`;
55
56       // console.log(arquivo);
57       $('#janela ul').append(arquivo);
58     });
59   });
60 }
61

```

Estado atual.



Para mostrar o vídeo em uma janela modal será utilizada o componente [Fancybox](#) da biblioteca [Fancyapps](#).



O suporte à biblioteca fancyapps pode ser visto nas linhas 13 e 20 do arquivo *index.html*.

```
4 <head>
5   <meta charset="UTF-8">
6   <meta http-equiv="X-UA-Compatible" content="IE=edge">
7   <meta name="viewport" content="width=device-width, initial-scale=1.0">
8   <title>YouTube API v3</title>
9
10  <!-- Arquivos/bibliotecas CSS -->
11  <link rel="stylesheet" href="./assets/css/style.css">
12
13  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/@fancyapps/ui@4.0/dist/fancybox.css" />
14
15  <!-- Arquivos/bibliotecas JavaScript -->
16  <script src="https://code.jquery.com/jquery-3.6.1.min.js" integrity="sha256-o88AwQnZB+VDvE9tvIXrM
17  QaPlFFSUTR+nldQm1LuPXQ=" crossorigin="anonymous"></script>
18
19  <script src="https://momentjs.com/downloads/moment-with-locales.min.js"></script>
20  <script src="https://cdn.jsdelivr.net/npm/@fancyapps/ui@4.0/dist/fancybox.umd.js"></script>
21
22  <script src="./assets/js/script.js"></script>
23 </head>
```

No início do código em *script.js* acrescente a configuração do componente (linha 5):

```
4 $(document).ready(function () {
5   Fancybox.bind("[data-fancybox-plt]", {});
```

Modifique agora o método *dados()*, linhas 30, 42, 46 a 48.

```
27 dados => {
28   // console.log(dados);
29   $.each(dados.items, (_, item) => {
30     let imagem, arquivo, dataPublicacao, descricao, video;
31
32     imagem = item.snippet.thumbnails.medium.url;
33     titulo = item.snippet.title;
34
35     moment.locale('pt');
36     dataPublicacao = new moment(item.snippet.publishedAt).format('DD [de] MMMM [de] YYYY');
37
38     descricao = item.snippet.description;
39     descricao = descricao.replace(/((https?|file|ftp):\\\/\\\/\\w?=.\\\/-;#~%-[?!(\\w\\s?&\\.\\\/;#~%=-]*)*)/g,
40       '|<a href="$1" target="_blank">$1</a>');
41
42     video = item.snippet.resourceId.videoId;
43
44     arquivo = `<li class="principal">
45       <div class="foto">
46         <a data-fancybox-plt href="http://www.youtube.com/watch?v=${video}">
47           
48         </a>
49       <div class="legenda">
50         <h5>${titulo}</h5>
51         <p>
52           ${descricao}
53           <br>
54           Data de publicação: ${dataPublicacao}
55         </p>
56       </div>
57     </div>
58   </li>`;
59
60   console.log(arquivo);
61   $('#janela ul').append(arquivo);
62   });
63 }
```