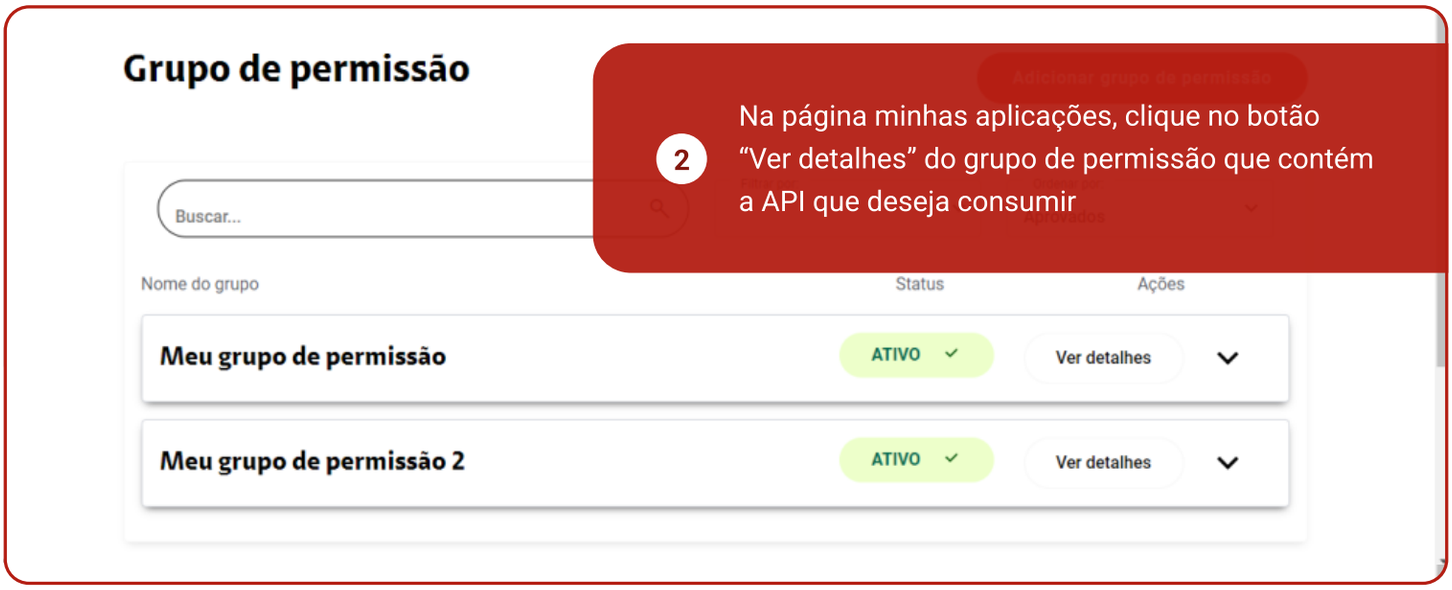
Tutorial de consumo da API Valida Endereço 2.0

Claro

Para iniciar o consumo da **API Valida Endereço 2.0**, é necessário adquirir o token de acesso através da chave pública e privada da aplicação contratada no portal.

» Como gerar um token na Base64



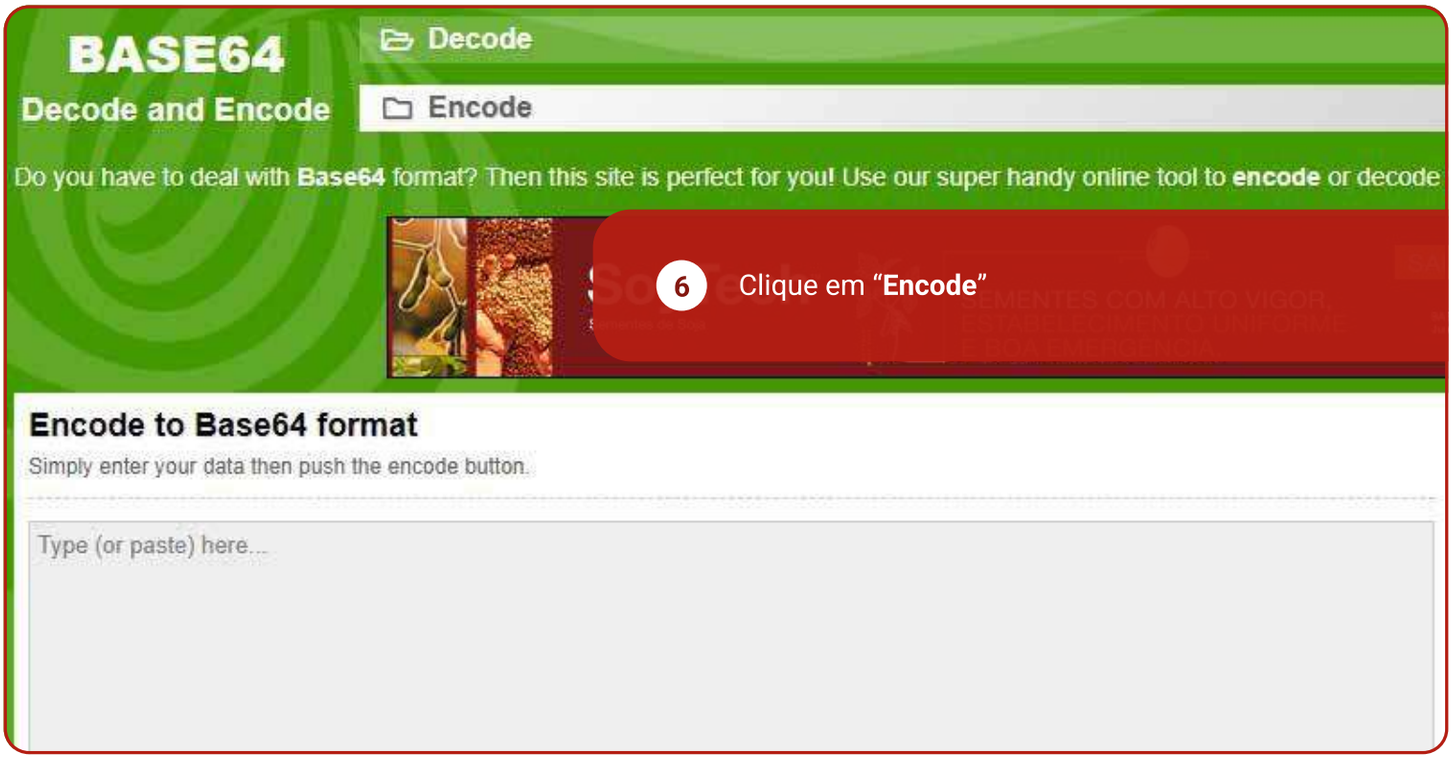
2 Na página minhas aplicações, clique no botão "Ver detalhes" do grupo de permissão que contém a API que deseja consumir

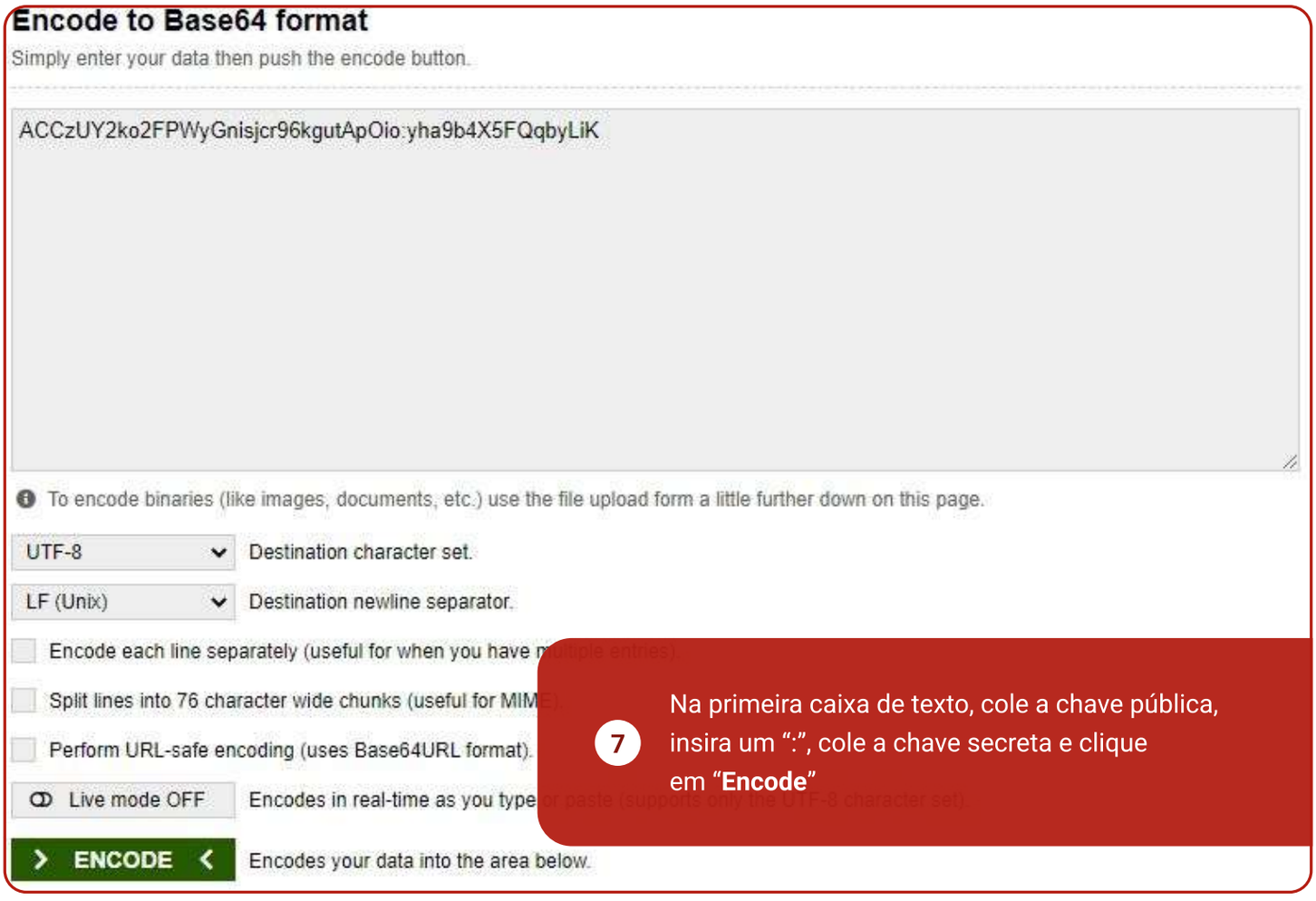
3 Clique na seta para ver os detalhes da aplicação desejada

4 Visualize as chaves pública e secreta, clique em "**Baixar arquivos**" e guarde a collection para utilizar mais tarde

Analytics

5 Acesse o site https://www.base64encode.org/.





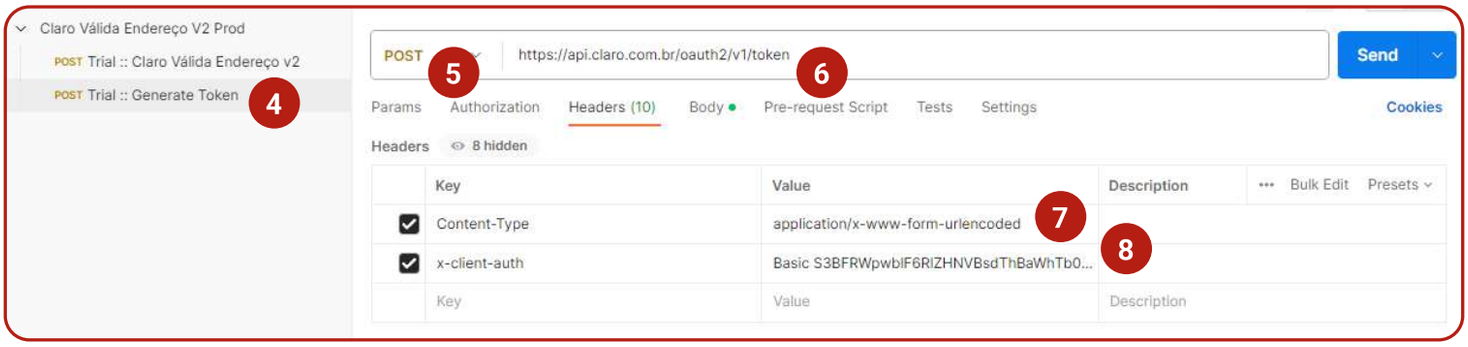
Ex: ACCzUY2ko2FPWyGnisjcr96kgutApOio : yha9b4X5FQqbyLiK

chave pública : **chave secreta**

O token não tem prazo de validade e pode ser utilizado por um período indeterminado, contanto que o status da aplicação permaneça **aprovado**

Usar o postman para obter o access token da API Valida Endereço 2.0

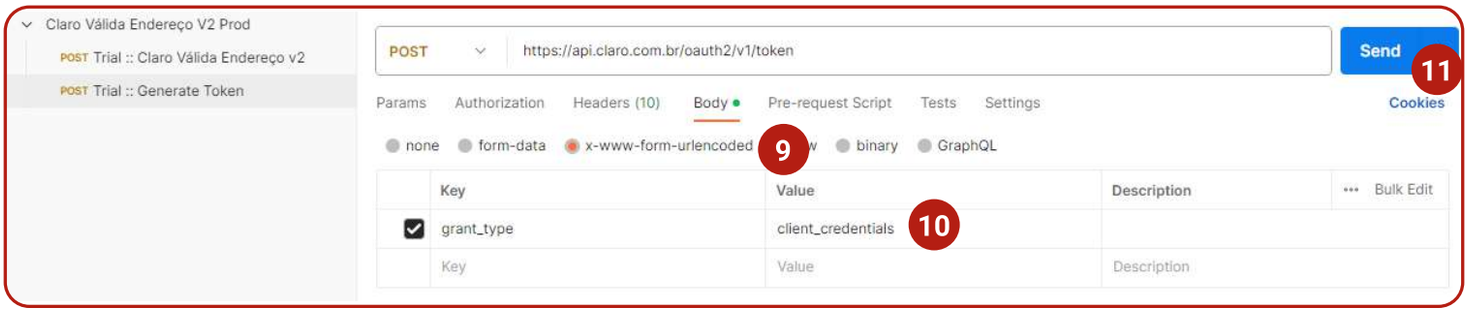
1. Inicie o postman
2. Clique no botão **Importar**
3. Selecione a **collection** baixada no passo anterior
4. Clique na opção **Generate Token**
5. Certifique-se que o método **POST** esteja selecionado
6. A URL inserida deverá ser **https://api.claro.com.br/oauth2/v1/token**
7. A guia **Headers** deve conter a chave **Content-Type** com o valor **application/x-www-form-urlencoded**
8. Troque o valor da chave **x-client-auth** colando o valor gerado na **base64** após o texto Basic



9 Na guia **Body**, a caixa **x-www-form-urlencoded** deve estar marcada

10 A guia **Body** deve conter a chave **grant-type** com o valor **client_credentials**

11 Envie a solicitação pressionando o botão **Envio**



12 Salve o **access token** para consumir a API

O **access token** gerado tem 24 horas de validade e deve ser solicitado novamente após esse período

» Usar o postman para consumir a API Valida Endereço 2.0

1 Na **collection** da API, clique na opção **Claro Valida Endereço v2**

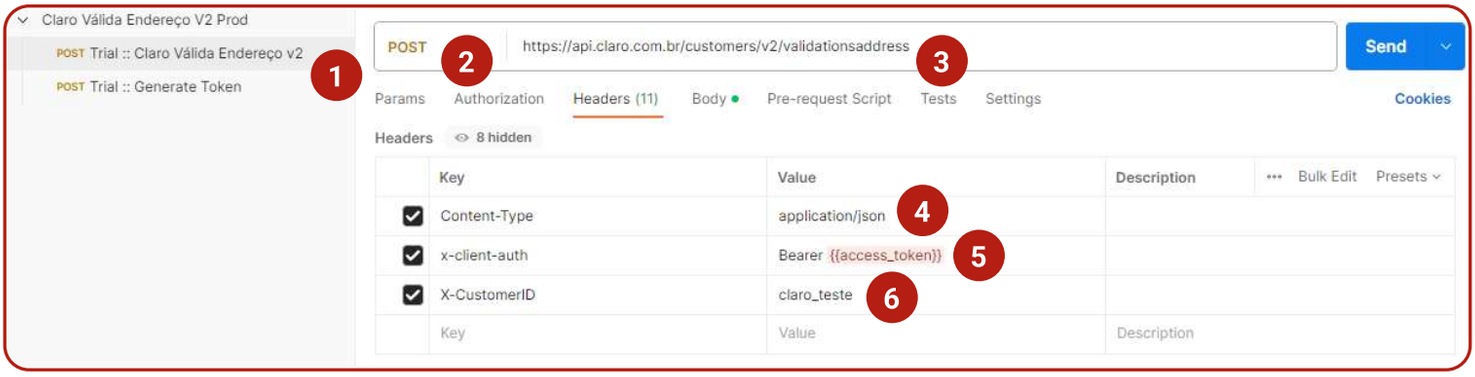
2 Certifique-se que o método **POST** esteja selecionado

3 A URL inserida deverá ser **https://api.claro.com.br/customers/v2/validationsaddress**

4 A guia **Headers** deve conter a chave **Content-Type** com o valor **application/json**

5 Troque o valor da chave **x-client-auth** colando o valor do **access token** após o texto Bearer

6 Na chave **X-CustomerID**, coloque o **nome da sua empresa** informado na criação do grupo de permissão



7 Na guia **Body**, a caixa **raw** e a seleção **JSON** devem estar marcados

8 Informe os todos valores que deseja consultar

9 Envie a solicitação pressionando o botão **Envio**

» Explicação dos dados da solicitação

```
Exemplo de json de request:
{
    "data": {
        "customer": {
            "addressZipCode": "string",
            "addressNumber": "string",
            "cpf": "string",
            "networkMsisdn": "string",
        }
    }
}
```

1 Campo **addressZipCode**, obrigatório, tipo string (5-8 caracteres) representa o CEP que deseja consultar

2. Campo **addressNumber**, opcional, tipo string (1-6 caracteres) representa o número do endereço que deseja consultar

3. Campo **cpf**, opcional, relacionado ao campo networkMsisdn, tipo string (11 caracteres) representa o CPF do cliente

4. Campo **networkMsisdn**, opcional, relacionado ao campo cpf, tipo string (1-15 caracteres) representa o número do telefone do cliente

> O campo CPF é uma string e precisa ser totalmente preenchido.
> Caso o CPF tenha menos de 11 caracteres deve ser sempre completado com zeros à esquerda.

» Dados de retorno com sucesso e erro

Status de retorno com sucesso

Exemplo de json de retorno:

```
{
    "data": {
        "result": {
            "message": "string",
            "score": "integer",
        }
    }
}
```

Explicação dos dados do json

1. Campo **message**, obrigatório, tipo string, retorna uma mensagem com interpretação descritiva do resultado obtido

2. Campo **score**, obrigatório, tipo int, retorna o score do CEP fornecido para o cliente encontrado

Status de retorno com erro

Exemplo de json de retorno:

```
{
    "apiVersion": “”,
    "transactionId": “”,
    "error": {
        "detailedMessage": "string",
        "errorCode": "string",
        "message": "string",
        "httpCode": "string",”link”: {
            "href": "string",
            "rel": "string"
        }
    }
}
```

Explicação dos dados do json

1. Campo **apiVersion**, obrigatório, tipo string, representa a versão da API
2. Campo **transactionId**, obrigatório, tipo string, representa o UUID da transação
3. Campo **detailedMessage**, obrigatório, tipo string, representa a Mensagem detalhada
4. Campo **errorCode**, obrigatório, tipo string, representa o código de erro Claro
5. Campo **message**, obrigatório, tipo string, representa a Mensagem de erro
6. Campo **httpCode**, obrigatório, tipo string, representa o código de retorno HTTP
7. Campo **href**, obrigatório, tipo string, representa o endereço para o detalhamento dos erros
8. Campo **rel**, obrigatório, tipo string, sempre “related”

» Status de response

Código de retorno 200 (Success)

Código de retorno 400 (Bad Request)

A solicitação não pôde ser entendida pelo servidor devido à sintaxe incorreta. O cliente NÃO DEVE repetir o pedido sem modificações

Código de retorno 401 (Unauthorized)

A solicitação requer autenticação do usuário

Código de retorno 403 (Forbidden)

O servidor entendeu a solicitação, mas está se recusando a atendê-la

Código de retorno 404 (Not Found)

O servidor não encontrou nada que corresponda ao request-URI (CPF não encontrado na base)

Código de retorno 405 (Not Found)

O método solicitado não é suportado pela API

Código de retorno 415 (Unsupported Media Type)

O servidor falhou ao atender uma solicitação. Por que o formato do payload não é um formato suportado

Código de retorno 406 (Not Acceptable)

Servidor não pode produzir uma resposta que combine com a lista de valores aceitáveis definidos

» Exemplo de código em curl

```
!/bin/bash

url_validationsaddress=https://api.claro.com.br/customers/v2/validationsaddress
url_token=https://api.claro.com.br/oauth2/v1/token
addressZipCode="XXXXXXXX"
addressNumber="XXXXX"
cpf="XXXXXXXXXXX"
networkMsisdn="XXXXXXXXXXXXX"
key=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
secret=XXXXXXXXXXXXXX

senha=$( echo -n $key:$secret | base64)

curl -X POST $url_token -H 'Content-Type:
application/x-www-form-urlencoded' -H 'cache-control: no-cache' -d
'grant_type=client_credentials' -H "x-client-auth: Basic $senha" >
json
grep 'access_token' json > atributoJson
token=$(cut -d"" -f4 atributoJson )
curl -X POST $url_validationsaddress -H 'Content-Type: application/json' -H
'cache-control: no-cache' -H "x-client-auth: Bearer $token" -H
"X-CustomerID: claro_exemplo" -d '{"data": {"customer":{"addressZipCode":
"$addressZipCode","addressNumber": "$addressNumber",
"networkMsisdn":"$networkMsisdn" ,"cpf": "$cpf"}}}'
```

» Exemplo de código em phyton

```
import requests, json
import base64

KEY="XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
SECRET="XXXXXXXXXXXXXX"

SENHA = base64.b64encode( (KEY + ':' + SECRET ).encode('ascii')
).decode('ascii')

GRANT_TYPE = "client_credentials";

URL = "https://api.claro.com.br";
END_POINT_ACCESS_TOKEN = "/oauth2/v1/token";
END_POINT_API_VALIDATIONSADDRESS = "/customers/v2/
validationsaddress";

headers={'Content-Type':'application/x-www-form-urlencoded','cach
e-control':'no-cache','x-client-auth': 'Basic {}'.format(SENHA)}

data = {'grant_type': 'client_credentials'}

respToken = requests.post(URL + END_POINT_ACCESS_TOKEN ,
headers=headers , data= data)
if respToken.status_code != 200:
   print('Erro: ' + str( respToken.status_code))
else:
   token = json.loads( respToken.text)['access_token']

   headers_app = {'Content-Type': 'application/json',
                  'X-CustomerID': 'claro_exemplo',
                  'cache-control':'no-cache',
                  'user-agent':'curl/7.60.0',
                  'x-client-auth': 'Bearer {}'.format(token)}

   addressZipCode="XXXXXXX"
   addressNumber="XXXXX"
   cpf="XXXXXXXXXXX"
   networkMsisdn="XXXXXX"

   data_app = {"data":{"customer":{"addressZipCode":
addressZipCode,"addressNumber": addressNumber ,"cpf": cpf ,
"networkMsisdn":networkMsisdn}}}

  resp = requests.post(URL +
END_POINT_API_VALIDATIONSADDRESS, headers=headers_app ,
data= json.dumps(data_app))

  if resp.status_code != 200:
      print('Erro: ' + str(resp.text))
   else:
      print('Valida ' + str( resp.text ))
```

» Exemplo de código em java

AJUSTAR DAQYU

```
package client.validationsAddress;

import java.io.IOException;
import java.net.URI;
import java.net.URISyntaxException;
import java.util.Base64;

import org.codehaus.jackson.JsonProcessingException;
import org.codehaus.jackson.map.ObjectMapper;
import org.springframework.http.HttpEntity;
import org.springframework.http.HttpHeaders;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import org.springframework.util.LinkedMultiValueMap;
import org.springframework.util.MultiValueMap;
import org.springframework.web.client.RestClientException;
import org.springframework.web.client.RestTemplate;


public class App {

        private static final String KEY =
"XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX";

        private static final String SECRET = "XXXXXXXXXXXXXX";

        private static final String GRANT_TYPE = "client_credentials";

        private static final String URL = "https://api.claro.com.br";

        private static final String END_POINT_ACCESS_TOKEN =
"/oauth2/v1/token";

        private static final String END_POINT_API_VALIDATIONS_ADDRESS =
"/customers/v2/validationsaddress";


   public static void main( String[] args ) throws URISyntaxException,
JsonProcessingException, IOException{

        App app = new App();

        String token = app.getAccessToken();
        String addressZipCode = ""XXXXXXXX";
        String addressNumber="XXXXX"
        String cpf = "XXXXXXXXXXX";
        String networkMsisdn="XXXXXXXXXXXXX";

        System.out.println("Válida Endereço: " + app.getValidationsAddress(
token , addressZipCode , addressNumber , cpf,networkMsisdn ));
   }

   public String getValidationsAddress(String acessToken, String
addressZipCode, String addressNumber, String cpf , String networkMsisdn)
throws RestClientException, URISyntaxException {

        HttpHeaders headers = new HttpHeaders();
        headers.setContentType(MediaType.APPLICATION_JSON);
        headers.add("cache-control","no-cache");
        headers.add("x-client-auth", "Bearer " + acessToken);
        headers.add("X-CustomerID", "claro_exemplo");

        HttpEntity<NetworkMsisdn> request = new HttpEntity<>(new
NetworkMsisdn(addressZipCode , addressNumber , cpf , networkMsisdn),
headers);

        RestTemplate restTemplate = new RestTemplate();
        ResponseEntity<String> result =
restTemplate.postForEntity(new URI(URL +
END_POINT_API_VALIDATIONS_ADDRESS),
                                request, String.class);
```

```
        return result.getBody();
    }

    public String getAccessToken() throws URISyntaxException,
JsonProcessingException, IOException {

        String senha = Base64.getEncoder().encodeToString(
App.KEY.concat(":").concat(App.SECRET).getBytes() );

        HttpHeaders headers = new HttpHeaders();

headers.setContentType(MediaType.APPLICATION_FORM_URLENCODED);
        headers.add("cache-control","no-cache");
        headers.add("x-client-auth", "Basic " + senha);

        MultiValueMap<String, String> parameters = new
LinkedMultiValueMap<>();
        parameters.add("grant_type", GRANT_TYPE);

        HttpEntity<MultiValueMap<String, String>> request = new
HttpEntity<>(parameters, headers);

        RestTemplate restTemplate = new RestTemplate();
        ResponseEntity<String> result =
restTemplate.postForEntity(new URI(URL +
END_POINT_ACCESS_TOKEN),
                        request, String.class);

        return new
ObjectMapper().readTree(result.getBody()).get("access_token").asTex
t() ;

    }

    class NetworkMsisdn{

        public NetworkMsisdn(String addressZipCode,String addressNumber
, String cpf, String networkMsisdn) {
                this.data = new Data(addressZipCode, addressNumber ,
cpf,networkMsisdn);
                }

                private Data data;

                public Data getData() {
                        return data;
                }

                public void setData(Data data) {
                        this.data = data;
                }
    }

    class Data{

        public Data(String addressZipCode, String addressNumber, String
cpf, String networkMsisdn) {
                        this.customer = new Customer(addressZipCode ,
addressNumber , cpf, networkMsisdn);
                }

                private Customer customer;

        public Customer getCustomer() {
                        return customer;
                }

        public void setCustomer(Customer customer) {
                        this.customer = customer;
                }

                class Customer{
```

```
                public Customer(String addressZipCode,String
addressNumber , String cpf, String networkMsisdn) {
                        this.addressZipCode = addressZipCode;
                        this.addressNumber = addressNumber;
                        this.cpf = cpf;
                        this.networkMsisdn = networkMsisdn;
                }
                private String cpf;

                private String networkMsisdn;

                        private String addressNumber;

                        private String addressZipCode;

                        public String getCpf() {
                                return cpf;
                        }

                        public void setCpf(String cpf) {
                                this.cpf = cpf;
                        }

                        public String getAddressNumber() {
                                return addressNumber;
                        }

                        public void setAddressNumber(String
addressNumber) {
                                this.addressNumber = addressNumber;
                        }

                        public String getAddressZipCode() {
                                return addressZipCode;
                        }

                        public void setAddressZipCode(String
addressZipCode) {
                                this.addressZipCode = addressZipCode;
                        }
        }
    }
}
```

» Exemplo de código em php

```
<?php

$key = 'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX';

$secret = 'XXXXXXXXXXXX';

$addressZipCode = 'XXXXXXX';

$addressNumber = 'XXXX';

$cpf = 'XXXXXXXXXXX';

$networkMsisdn = 'XXXXXXXXXXXXX';

$token = getAccessToken();

$validationsaddress = getValidationsAddress($token ,
$addressZipCode, $addressNumber , $cpf, $networkMsisdn);
```

```
echo $validationsaddress;

function getAccessToken() {

    try{

            $senha = base64_encode($GLOBALS['key'] . ':' .
$GLOBALS['secret']);

            $ch = curl_init();
            curl_setopt($ch, CURLOPT_URL,
"https://api.claro.com.br/oauth2/v1/token");
            curl_setopt( $ch, CURLOPT_HTTPHEADER, [
                'Content-Type: application/x-www-form-urlencoded',
                'x-client-auth: Basic ' . $senha

            ]);
            curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
            curl_setopt($ch, CURLINFO_HEADER_OUT, true);
            curl_setopt($ch, CURLOPT_POST, true);
            curl_setopt($ch, CURLOPT_POSTFIELDS,
'grant_type=client_credentials');

            $response = curl_exec($ch);

            if (!$response){
                $error = curl_error($ch);
                $info = curl_getinfo($ch);
                die("cURL request failed, error = {$error}; info = " .
print_r($info, true));
            }
            if(curl_errno($ch)){
                curl_close($ch);
                echo 'error:' . curl_error($ch);
            } else {
                curl_close($ch);
                return json_decode($response)->access_token;
            }

    } catch (Exception $e) {
        return 'Erro '. $e;
    }
}


function getValidationsAddress($token , $addressZipCode,
$addressNumber , $cpf , $networkMsisdn){

    try {

        $payload = '{"data":{""customer": {"addressZipCode":' .
$addressZipCode . ","addressNumber": ' . $addressNumber
.","networkMsisdn":'.$networkMsisdn.' , "cpf": ' . $cpf . '}}}';
        $ch = curl_init();
        curl_setopt($ch, CURLOPT_URL,
"https://api.claro.com.br/customers/v2/validationsaddress");
        curl_setopt( $ch, CURLOPT_HTTPHEADER, [
            'Content-Type: application/json',
                'X-CustomerID: claro_exemplo',
            'x-client-auth: Bearer ' . $token

        ]);
        curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
        curl_setopt($ch, CURLINFO_HEADER_OUT, true);
        curl_setopt($ch, CURLOPT_POST, true);
        curl_setopt($ch, CURLOPT_POSTFIELDS, $payload);

        $response = curl_exec($ch);
```

```
        if (!$response){
            $error = curl_error($ch);
            $info = curl_getinfo($ch);
            die("cURL request failed, error = {$error}; info = " .
print_r($info, true));
        }
        if(curl_errno($ch)){
            curl_close($ch);
            echo 'error:' . curl_error($ch);
        } else {
            curl_close($ch);
            return $response;
        }

    } catch (Exception $e) {
        return 'Erro '. $e;
    }
}
?>
```

Claro