Ministério do Planejamento
Secretaria de Tecnologia da Informação e Comunicação

Acesso Digital Único da Plataforma de Cidadania Digital (Brasil Cidadão)

Roteiro de Integração (SDK)

Sumário

Contexto

O Decreto n° 8.936, de 19 de dezembro de 2016, pemititu o inicio do projeto da plataforma de cidadania digital, que contempla diversas diretrizes para a prestação de serviços públicos digitais, das quais fazem parte a convergência autoritativa e a federação dos processos de autenticação dos serviços digitais. Para essa diretriz foi concebido o conceito da Plataforma de Autenticação Digital do Cidadão, o projeto Brasil Cidadão, tendo, como destaque no decreto, o mecanismo de acesso digital único.

Dentro deste contexto, podemos destacar as diversas dificuldades com múltiplas contas de acesso sob responsabilidade do cidadão e variados bancos de dados cadastrais, tais como a duplicidade e inconsistência de informações, falta de integração, dados dispersos e diversas formas de autenticação. Problemas enfrentados por cidadãos ao tentar consumir um serviço público digital oferecido pelo governo federal. Analisando essas dificuldades, o Ministério do Planejamento, Desenvolvimento e Gestão (MP), em parceria com o Serviço Federal de Processamento de Dados (Serpro), disponibilizou a plataforma central de autenticação digital do cidadão, o Brasil Cidadão.

Essa é a nova proposta do Governo federal, para facilitar a identificação e autenticação do cidadão, privilegiando a governança e a convergência autoritativa, e finalmente o controle de acesso unificado. A Plataforma de Cidadania Digital chega para ampliar e simplificar o acesso dos cidadãos brasileiros aos serviços públicos digitais, inclusive por meio de dispositivos móveis.

Introdução

Este documento é o elemento para orientar a integração da Plataforma de Autenticação Digital do Cidadão – Brasil Cidadão a qualquer ambiente. A partir de agora, será feita uma revisão sobre a arquitetura de serviço e alguns conceitos utilizados pela Plataforma, além de uma explicação sobre procedimentos administrativos essenciais para autorizar o acesso à Plataforma.

Este documento contém as formas de chamadas a operações, parâmetros e métodos de integração, e, por último, os procedimentos para permitir a conectividade entre os ambientes de implantação.

Arquitetura de Serviço e Protocolos

OpenID Connect (OIDC)

O Openid Connect é um protocolo baseado no OAuth 2.0 que especifica autorização e autenticação. Define como implementar o gerenciamento de autorizações de acesso, gerenciamento de sessão, fornecimento de informações sobre usuário logado. O OIDC permite executar o logon único dos usuários e apresenta o conceito de um *id token*: um *token* de segurança que permite verificar a identidade do usuário e obter informações básicas sobre o usuário. Tem característica de ser interoperável, porque segue o protocolo *RestFull* e usa o formato de saída de dados: JSON (*JavaScript Object Notation*).

Além disso, o OIDC suporta vários tipos de clientes, como aplicações que utilizam o *browser*, clientes *javascript*, aplicações mobile e outros. A Figura 1 ilustra as requisições da autenticação entre cliente e servidor.

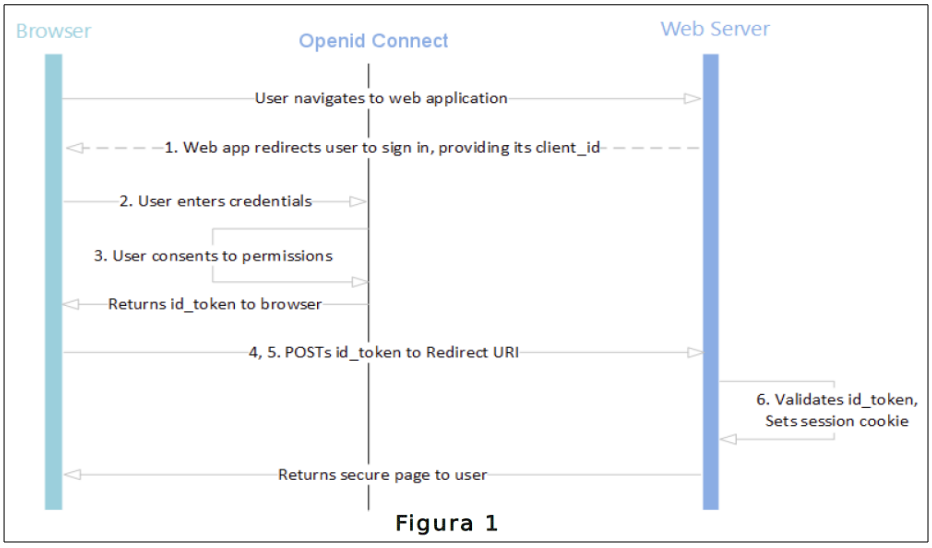

Figura 1

OAUTH2

OAuth2 é um protocolo aberto para autorização que permite aos clientes obterem acesso a recursos protegidos do servidor em nome do proprietário do recurso. O proprietário pode ser um cliente ou usuário final. Também especifica como um usuário final pode autorizar o acesso de terceiros aos seus recursos do servidor sem precisar compartilhar suas credenciais. Atualmente ele está sendo usado por grandes empresas como Google, Facebook, Microsoft, Twitter, e outros.

O protocolo fornece 4 estratégias para concessão de autorização: código de autorização, implícita, credenciais de senha do proprietário do recurso e credenciais do cliente. A estratégia usada no Brasil Cidadão é o código de autorização, que utiliza um *token*.

Json Web Token - JWT

O JWT define como transmitir objetos JSON de forma segura entre aplicações. Tem a característica de ser um padrão aberto.

A manipulação do padrão possui uma assinatura a ser realizada com uma palavra secreta ou uma chave publica/privada.

O JWT é composto por 3 elementos: *Headers*, *Payload e Signature,* explicados a seguir.

Headers

São objetos JSON definidos por 2 atributos: tipo do token(typ) e o algorítimo (alg) de encriptação , como SHA256 ou RSA. Exemplo:

```
{
        "alg": "HS256",
        "typ": "JWT"
}
```

Payload

São os atributos de uma entidade representada por objetos JSON. Exemplo:

```
{
        "sub": "1234567890",
        "name": "John Doe",
        "admin": true
}
```

Signature

Para criar a assinatura, há necessidade de assinar o header codificado, o payload codificado e informar o secret, palavra secreta definida na aplicação. A assinatura é criada para verificar se quem enviou a requisição é quem realmente diz ser.

O resultado é um token (exemplo 1 – token gerado). O token é dividido em 3 partes separadas pelo ponto. As três partes equivalem ao hash do header, payload e a signature.

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9(header).eyJzdWIiOiIxMjM0NTY3ODkwI
iwibmFtZSI6IkpvaG4gRG9lIiwiYWRtaW4iOnRydWV9(payload).TJVA95OrM7E2c
Bab30RMHrHDcEfxjoYZgeFONFh7HgQ(signature)
```

Exemplo 1 – token gerado

Para acessar recursos protegidos, o cliente deve enviar o token gerado através do atributo *Authorization* do *header* da requisição, com a *flag Bearer*, como abaixo:

```
Authorization:Bearer
```

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY
3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiYWRtaW4iOnRydWV9.TJVA95
OrM7E2cBab30RMHrHDcEfxjoYZgeFONFh7HgQ

Código Autorizador

A estratégia é autorizar clientes a acessarem informações dos usuários proprietários através de um código identificador (Cliente ID). O cliente deve ser cadastrado no Portal de Gestão do Brasil Cidadão para obter um Cliente ID. Após , o proprietário da informação, ao ser requisitado, deve habilitar esse cliente para ter acesso às suas informações. O cliente está habilitado para obter do servidor os recursos necessários.

Essa estratégia é muito utilizada no mercado pois é otimizada para as aplicações *server-side*, o qual o código fonte não é exposto e a confidencialidade do Cliente ID é mantida.

A Figura 2 explica a obtenção de recursos do servidor para um cliente cadastrado.

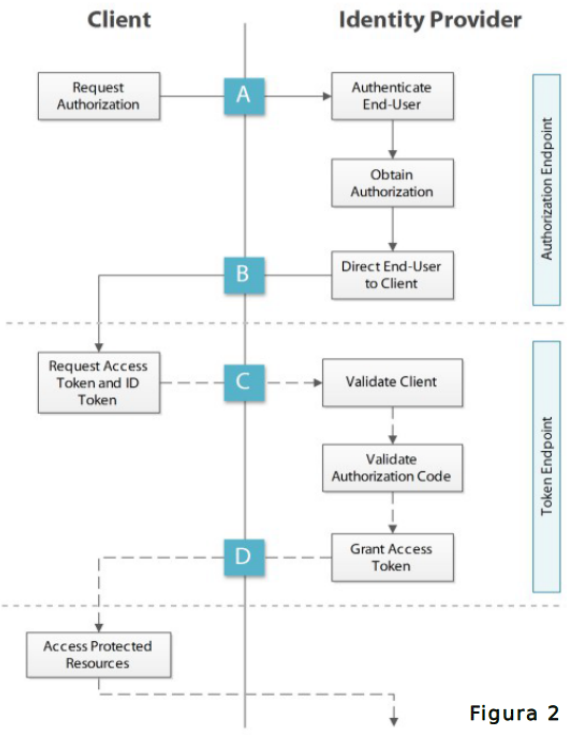

Figura 2 – Obtenção de recursos do Servidor para cliente cadastrado

No passo A, a aplicação cliente solicita autorização. O usuário realiza a autenticação no Brasil Cidadão, obtêm a autorização e é redirecionado para o cliente, conforme o passo B. Já no passo C, o cliente solicita o *Access Token* e o *ID Token*, que são as credenciais para permitir as consultas de recursos por um determinado tempo. As credenciais são geradas no servidor e não podem navegar pelo cliente, para manter a confidencialidade. Após o cliente ser validado e receber o *ID Token* e *Access Token* no passo D, ele pode solicitar ao Brasil Cidadão os recursos necessários. A Figura 3 mostra os parâmetros necessários para as requisições da Figura 2.

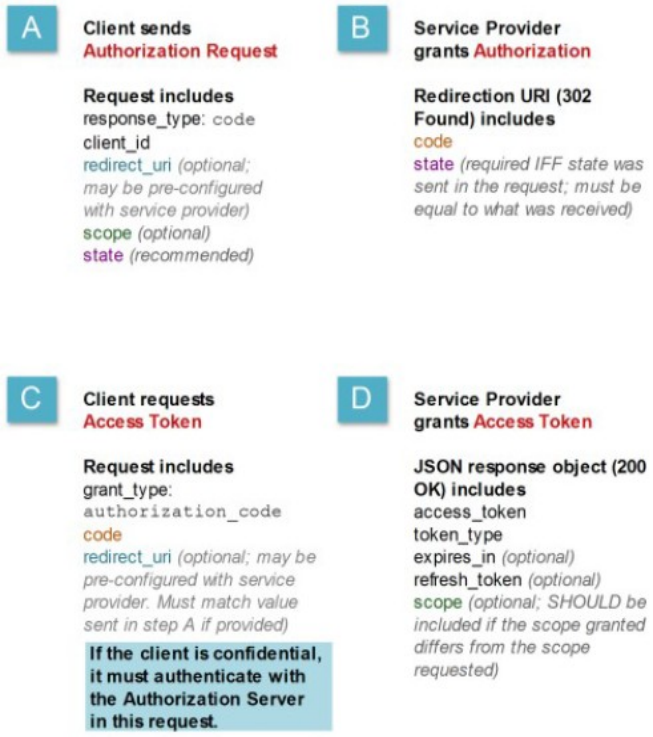

Figura 3

Figura 3 – Explicação dos parâmetros necessários para requisições presentes na figura 2

Autorização para Recursos protegidos

Na plataforma de autenticação, os serviços utilizam informações pessoais relacionadas aos cidadãos, logo, existe a necessidade de vincular recurso informacional ao serviço no processo de habilitação. Quando o cidadão se autentica e acessa algum recurso pela primeira vez, uma solicitação de autorização de uso de dados pessoais é feita. A autorização do uso de dados pessoais permite o funcionamento correto. A figura 4 apresenta a tela em que o cidadão autoriza o uso de recurso protegido (dados pessoais):

Figura 4 – Autorização de Uso de Dados Pessoais

Escopos de Atributos

São conjuntos de informações fornecidos a quem possui autorização.
A figura 5 apresenta tela dos escopos por serviços:

Figura 5 – exemplos de escopos de atributos.

Atributos Disponíveis

Existem dois escopos disponibilizados pelo Brasil Cidadão para apresentar os atributos disponíveis:

- brasil_cidadao (CPF, Nome, e-mail, telefone, foto);
- brasil_cidadao_empresa (CNPJ, Nome Fantasia, CPF do Responsável, Nome do Responsável, Atuação no CNPJ).

Níveis de Autenticação

Tem como principal característica ser um recurso de segurança da informação das identidades, que permite flexibilidade para realização do acesso e atribuição de níveis em conformidade com as normas de segurança. São eles:

- **Nível 1** (Auto Cadastro): Identidade cadastrada com conferência simples;
- **Nível 2** (Dados cadastrais convalidados): Identidade cadastrada com convalidação de dados em bases oficiais;
- **Nível 3** (Dados cadastrais Nível Balcão): Identidade certificada a partir da conferência de documentos de forma presencial em posto de atendimento de governo;
- **Nível 4** (Biometria) : Identidade cadastrada com convalidação de dados biométricos;
- **Nível 5** (Cadastro assinado digitalmente): Identidade cadastrada a partir do certificado digital de pessoa física e assinado digitalmente.

Selos de Confiabilidade Cadastral

Consistem em compor níveis de segurança das contas com a obtenção dos atributos autoritativos do cidadão a partir das bases oficias de governo, por meio das quais permitirão a utilização da credencial de acesso em sistemas internos dos clientes e serviços providos diretamente ao cidadão.

Uso possível para o selo é o uso do nível de confiança cadastral pelos serviços para aplicar controle de acesso às funcionalidades mais críticas. Na figura 6, demonstra os selos de confiabilidade cadastral na área cidadão do Brasil Cidadão:

Figura 6 – selos de confiabilidade cadastral

Lançador de Serviços

O Brasil Cidadão apresenta o conceito de Single Sign On na arquitetura: um ponto único de autenticação, que permite ao usuário fazer o login e acessar diversos serviços ou sistemas integrados. Para permitir isto de forma explicita, o Brasil Cidadão apresenta um funcionalidade chamada Lançador de Serviços.

Esta funcionalidade lista todos os serviços ou sistemas integrados autorizados uma vez pelo cidadão. A figura abaixo demonstra o Lançador de Serviços ativo e as configurações são entregues pelos serviços integrados ao Brasil Cidadão.

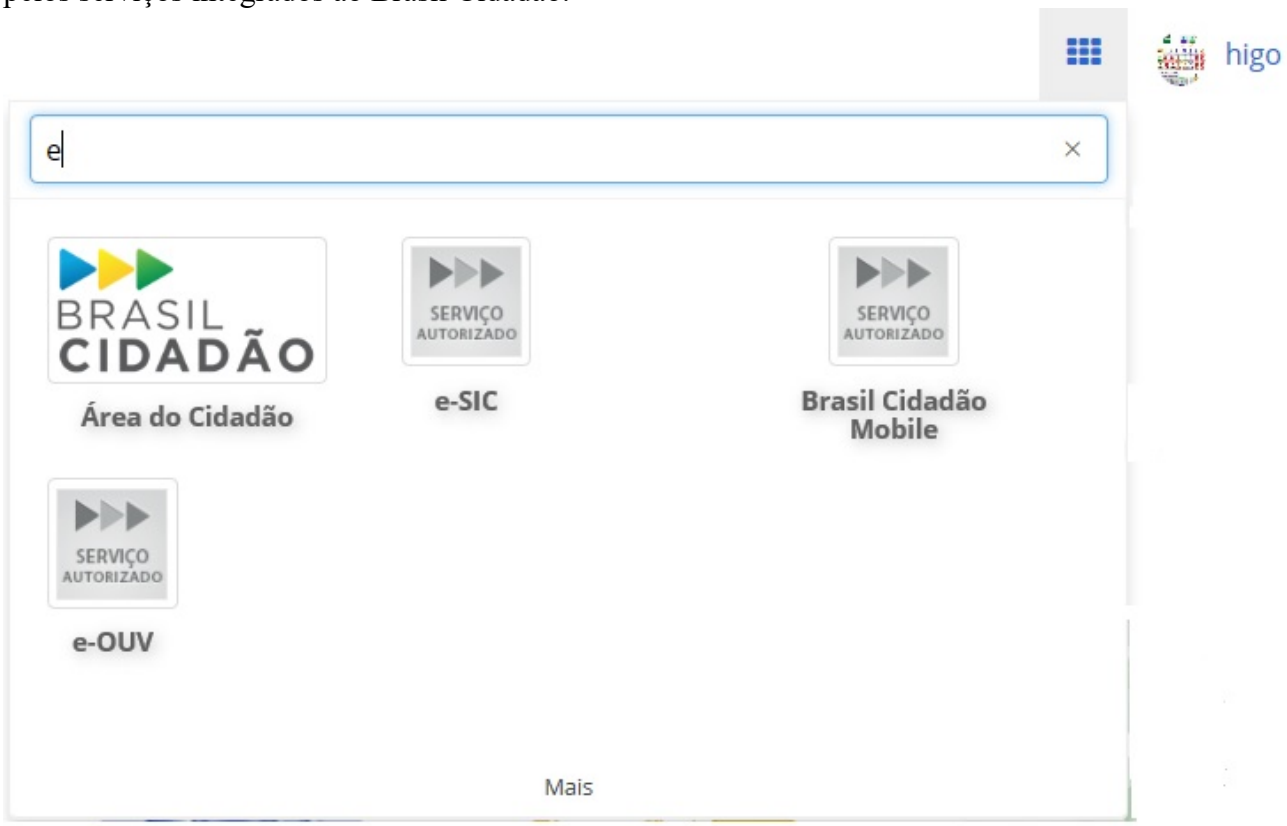

Figura 7 – Lançador de Serviços da Área do Cidadão do Brasil Cidadão.

Iniciando a Integração

Solicitação de Configuração

Para utilização do sistema Brasil Cidadão, há necessidade de liberar os ambientes para aplicação cliente possa utilizar. Essa liberação ocorre por meio do preenchimento do plano de configuração encaminhado junto com este documento (**plano-configuracao-brasil-cidadao-vX.doc**).

O formulário deverá ser encaminhado para os integrantes da Secretaria de Tecnologia da Informação e Comunicação (SETIC) do Ministério do Planejamento para realizar configuração da utilização do Brasil Cidadão.

Parâmetros de integração

Parâmetros de autenticação do serviço consumidor:

- *client_id*: chave de acesso, que identifica o serviço consumidor;
- *client_secret*: senha de acesso do serviço consumidor.

Parâmetros do *queries string*:

- *response_type*: especifica para o provedor o tipo de autorização. Esse tipo de autorização gerará um código para o serviço consumidor. Nesse caso sempre usaremos o *code*;
- *client_id*: o identificador do serviço consumidor fornecido pelo Portal de Gestão.
- *scope*: especifica os recursos que o serviço consumidor quer obter. No caso da autenticação é obrigatório concatenar o *scope* "***openid***" para retornar as informações do usuário logado.
- *redirect_uri*: a *url* do serviço consumidor que o provedor de autenticação redirecionará após o *login* e a autorização;
- *nonce*: sequência de caracteres usado para associar uma sessão do serviço consumidor a um *Token de ID* e para atenuar os ataques de repetição. Pode ser um valor aleatório, mas que não seja de fácil dedução;
- *state*: valor usado para manter o estado entre a solicitação e o retorno de chamada.

Os parâmetros *nonce* e *state* representam variáveis de controle que são utilizadas para autenticidade por parte do Consumidor. Não são obrigatórios o trabalho pelo Consumidor.

Métodos e interfaces de integração (Passo-a-Passo para Integrar)

URLs importantes

URL do Provider: https://testescp-ecidadao.estaleiro.serpro.gov.br

URL de Serviços: https://testeservicos-ecidadao.estaleiro.serpro.gov.br/servicos-ecidadao/ecidadao

Autenticação

Para que a autenticação aconteça, todo o canal de comunicação deve ser realizado com o protocolo HTTPS.

1. Ao requisitar autenticação via Provedor, o mesmo verifica se o usuário está logado. Caso o usuário não esteja logado o provedor redireciona para a página de login.
2. A requisição é feita através de um *GET* para o endereço https://*(url do provider brasil cidadão)*/scp/authorize passando as seguintes informações:
    1. Parâmetros query:
        - response_type: ***code***;
        - *client_id*: <CLIENT_ID> fornecido pelo Brasil Cidadão para a aplicação cadastrada;
        - *scope*: um ou mais escopos inseridos para a aplicação cadastrada. Se for mais de um, esta informação deve vir separada pelo caractere "+".
        - *redirect_uri*: <URI de retorno cadastrada para a aplicação cliente>.
        - *nonce*: <sequência alfa numérica aleatória>.
        - *state*: <sequência alfa numérica aleatória>. Item não obrigatório.

        Exemplo de requisição:

https://*(url do provider brasil cidadão)*/scp/authorize?response_type=code&client_id=ec4318d6-f797-4d65-b4f7-39a33bf4d544&scope=openid+brasil_cidadao&redirect_uri=http://appcliente.com.br/phpcliente/loginecidadao.Php&nonce=3ed8657fd74c&state=358578ce6728b

3. Após autenticado, o provedor redireciona para a página de autorização. O usuário habilitará o consumidor no sistema para os escopos solicitados. Caso o usuário da solicitação autorize o acesso ao recurso protegido, é gerado um "*ticket* de acesso" intitulado *access_token* (*vide especificação OAUTH 2.0*);

4. Após a autorização, a requisição é retornada para a URL especificada no passo 1, enviando os seguintes parâmetros: ***code***=Z85qv1e ***state***=358578ce6728b. Lembrando que para essa requisição o *code* têm um tempo de expiração e só pode ser utilizado uma única vez;

5. Para obter o *token* e o *access token*, o consumidor deve fazer uma requisição *POST* para o endereço https://*(url do provider brasil cidadão)*/scp/token passando as seguintes informações:
    1. No *header* :
        - *Content-Type*: tipo do conteúdo da requisição que está sendo enviada. Nesse caso estamos enviando como um formulário;
        - *Authorization*: Informação codificada na *Base64*, no seguinte formato: CLIENT_ID:CLIENT_SECRET (utilizar a página https://www.base64decode.org/ para gerar codificação). A palavra *Basic* deve

está antes da informação.

Exemplo de header:

Content-Type:application/x-www-form-urlencoded

Authorization: Basic

ZWM0MzE4ZDYtZjc5Ny00ZDY1LWI0ZjctMzlhMzNiZjRkNTQ0OkFJSDRoa
XBfTUJYcVJkWEVQSVJkWkdBX2dRdjdWRWZqYlRFT2NWMHlFQll4aE1i
YUJzS0xwSzRzdUVkSU5FcS1kNzlyYWpaZ3I0SGJu VUM2WlRXV1lJOA==

2. E os parâmetros *query*:
   - *grant_type*: authorization_code*;*
   - *code:* <código retornado pela requisição anterior> *(exemplo: Z85qv1);*
   - *redirect_uri:* <URI de retorno cadastrada no Brasil Cidadão> *(exemplo: http://ecidadao.serpro/phpcliente-val/login-ecidadao.php)*;

Exemplo de requisição

https://*(url do provider brasil cidadão)*/scp/token?grant_type=authorization_code&code=*Z85qv1*&redirect_uri=http://appcliente.com.br/phpcliente/loginecidadao.Php

O serviço retornará, em caso de sucesso, a informação, no formato JSON, conforme exemplo:

```
{
"token":"RsT5OjbzRn430zqMLgV3Iahgsdfjgsjfg JHKJHJuyiiuyU"
"access_token":"RsT5OjbzRn430zqMLgV3Ia"
}
```

Ou , no caso de falha, a informação, conforme exemplo abaixo:

```
{
"error":"invalid_request"
}
```

6. De posse das informações de *token e access token*, a aplicação consumidora já está habilitada para consultar dados de recursos protegidos, que são os escopos de informações. Deve fazer uma requisição *GET* para o endereço https://(url de serviços do brasil cidadão)/usuario/getUserInfo/brasil_cidadao passando as seguintes informações:

1. No *header*:
   - *Authorization*: *Bearer* <ACCESS_TOKEN> da requisição *POST* anterior.

Exemplo de retorno do barramento de serviços no formato *JSON*

{
"cpf":"88918894588",
"nome":"HENRIQUE PRETORIUM ",
"email":"henrique.pretorium@enterprisex.gov.br",
“telefone”:”00000000”,
“foto”:”informacao da foto em formato base 64 com tamanho até 4 MB”
}

Resultados Esperados do Acesso ao Serviços de Autenticação

Os acessos aos serviços do Brasil Cidadão ocorrem por meio de chamadas de *URLs* e a resposta são códigos presentes conforme padrão do protocolo http. Estes códigos são:

- **Código 200:** Dados acessados e retornados em formato *JSON* ao usuário, de acordo com o *JSON* de cada escopo;
- **Código 400:** *Token* recebido por mais de um método;
- **Código 401:** *Token* não encontrado ou inválido , CPF inválido, usuário não existente no sistema, *access token* inválido;
- **Código 403:** Escopo solicitado não autorizado pelo usuário;
- **Código 404:** Escopo obrigatório.

Acesso ao Serviço de Cadastro de Pessoas Jurídicas

O Brasil Cidadão disponibiliza dois serviços para acesso a informações de Pessoa Jurídica. O primeiro apresenta todos os CNPJs cadastrados para um determinado usuário. O segundo, utiliza desse CNPJ para extrair informações cadastradas no Brasil Cidadão para aquela pessoa e empresa.

Para acessar o serviço que disponibiliza os CNPJs vinculados a um determinado usuário, é necessário o seguinte:

1. Na requisição de autenticação, adicionar o escopo “brasil_cidadao_empresa“, conforme exemplo:

https://*(url do provider brasil cidadão)*/scp/authorize?response_type=code&client_id=ec4318d6-f797-4d65-b4f7-39a33bf4d544&scope=openid+brasil_cidadao+brasil_cidadao_empresa&redirect_uri=http://appcliente.com.br/phpcliente/loginecidadao.Php&nonce=3ed8657fd74c&state=358578ce6728b

2. Com o usuário autenticado, a aplicação deverá realizar uma requisição por meio do método *GET* a URL *URL*: https://(url de serviços do brasil cidadão)/servicos-ecidadao/ecidadao/usuario/getConfiabilidade enviando as seguintes informações:
    - No *Header*:
        - *Authorization* : *Bearer* <ACCESS_TOKEN> ;
3. O resultado em formato *JSON* são selos de confiabilidade da autenticação. O delo a ser verificado será o "Representante Legal do CNPJ", conforme o exemplo abaixo:

```
{
"id": 0,
"nivel": 11,
"descricao": "REPRESENTANTE E-CNPJ"
}
```

4. Com o usuário autenticado, a aplicação deverá realizar uma requisição por meio do método *GET* a URL https://(url de serviços do brasil cidadão)/empresa/escopo/<escopo> enviando as seguintes informações:
    - No *Header*:
        - *Authorization* : *Bearer* <ACCESS_TOKEN> ;
    - Parâmetro escopo: o escopo de empresa, por exemplo, *brasil_cidadao_empresa*
5. O resultado em formato *JSON* é a lista de CNPJs do CPF autenticado, conforme o exemplo abaixo:

```
{
        "cnpjs": [
        {"cnpj", <CNPJ>
        "nome" <NOME FANTASIA DA EMPRESA>,
        "atuacao" <ATUAÇÃO DO CPF NO CNPJ INFORMADO tendo o valor "SOCIO" ou "COLABORADOR">},
        {"cnpj",
        "nome",
        "atuacao"}
    ],
    "cpf": "<CPF DO USUÁRIO LOGADO>"
}
```

6. Com o usuário autenticado, a aplicação cliente deverá acessar, por meio do método *GET*, a *URL*: https://(url de serviços do brasil cidadão)/empresa/<cnpj>/escopo/<escopo> enviando as seguintes informações:
    - No *header*:
        - *Authorization*: *Bearer* <ACCESS_TOKEN>

- Parâmetro cnpj: o CNPJ da empresa formatado (sem ponto, barra etc).
- Parâmetro escopo: o escopo de empresa, por exemplo, *brasil_cidadao_empresa*

7. O resultado em formato *JSON* é o detalhamento do CNPJ do CPF autenticado, conforme o exemplo abaixo:

```
{
    "cnpj":"<CNPJ>",
    "nomeFantasia":"<NOME FANTASIA>",
    "atuacao":"<ATUACÃO tendo o valor “SOCIO” ou “COLABORADOR”>",
    "cpfResponsavel":"<CPF DO RESPONSÁVEL>",
    "nomeResponsavel":"<NOME DO RESPONSÁVEL>"
}
```

Resultados Esperados do Acesso ao Serviço de Cadastro de Pessoas Jurídicas

Os acessos aos serviços do Brasil Cidadão ocorrem por meio de chamadas de *URLs* e a resposta são códigos presentes conforme padrão do protocolo http. Estes códigos são:

- **Código 200:** Dados acessados e retornados em formato *JSON* ao usuário, de acordo com cada escopo;
- **Código 400:** *Token* recebido por mais de um método;
- **Código 401:** *Token* não encontrado ou inválido, CNPJ inválido, usuário não existente no sistema, *access token* inválido;
- **Código 403:** Escopo solicitado não autorizado pelo usuário;
- **Código 404:** Escopo obrigatório.

Acesso ao Serviço de Confiabilidade Cadastral (Selos)

Para acessar o serviço de consulta de empresas é necessário:

1. Com usuário autenticado, deverá acessar, por meio do método *GET*, a *URL*: https://(url de serviços do brasil cidadão)/servicos-ecidadao/ecidadao/usuario/getConfiabilidade ;
    - No *header*:
        - *Authorization*: *Bearer* <ACCESS_TOKEN>

2. A resposta em caso de sucesso retorna sempre um *array* de objetos *JSON* no seguinte formato: {id, nivel, descricao};

Resultados Esperados do Acesso ao Serviço de Confiabilidade Cadastral (Selos)

Os selos existentes no Brasil Cidadão são:

```
        {
                "id": 0,
                "nivel": 2,
                "descricao": "Institucional" (Servidor Público)
        },
        {
                "id": 0,
                "nivel": 1,
                "descricao": "Conformidade"
        },
        {
                "id": 0,
                "nivel": 4,
                "descricao": "Biometria"
        },
        {
                "id": 0,
                "nivel": 5,
                "descricao": "Certificado Digital"
        },
        {
                "id": 0,
                "nivel": 3,
                "descricao": "Convalidação" (Módulo Balcão)
        }
        {
                "id": 0,
                "nivel": 10,
                "descricao": "DNI"
        },
        {
                "id": 0,
                "nivel": 11,
                "descricao": "REPRESENTANTE E-CNPJ"
        }
```

Exemplo de implementação

Os exemplos abaixo são exemplos básicos da forma de realizar as requisições para Brasil Cidadão. Cabe ao desenvolvedor realizar a organização e aplicação da segurança necessária na aplicação consumidora.

JAVA

```
package teste;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.math.BigInteger;
import java.net.MalformedURLException;
import java.net.URL;
import java.net.URLEncoder;
import java.security.SecureRandom;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Base64;
import java.util.List;

import javax.net.ssl.HttpsURLConnection;

import org.jose4j.json.internal.json_simple.JSONArray;
import org.jose4j.json.internal.json_simple.JSONObject;
import org.jose4j.json.internal.json_simple.parser.JSONParser;
import org.jose4j.jwk.PublicJsonWebKey;
import org.jose4j.jwt.JwtClaims;
import org.jose4j.jwt.consumer.JwtConsumer;
import org.jose4j.jwt.consumer.JwtConsumerBuilder;


/**
 *
 * O presente código tem por objetivo exemplificar de forma minimalista o consumo dos serviços utilizados pelo Brasil Cidadão.
 *
 */

public class ExemploServicosBrasilCidadao_manual {
```

```
	/**
	 * O processo de autenticação e autorização de recursos ocorre essencialmente em três etapas:
	 *		Etapa 1: Chamada do serviço de autorização do Brasil Cidadão;
	 *	Etapa 2: Recuperação do Access Token e
	 *	Etapa 3: Validação do Access Token por meio da verificação de sua assinatura.
	 * Após concluída essas três etapas, a aplicação cliente terá as informações básicas para conceder acesso de acordo com suas próprias políticas de autorização.
	 * Caso a aplicação cliente necessite de informações adicionais, fica habilitado o acesso à todos os serviços (presentes e futuros) fornecidos pelo Brasil Cidadão por meio do access token.
	 * O presente código exemplifica a chamada aos seguintes serviços:
	 *		Serviço 1: getUserInfo - Serviço que recupera informações do usuário direto da Receita Federal;
	 *		Serviço 2: getConfiabilidade - Serviço que recupera os selos de confiabilidade atribuidos ao usuário;
	 *		Serviço 3: getEmpresasVinculadas - Serviço que recupera a lista de empresas vinculadas ao usuário;
	 *		Serviço 4: getDadosEmpresa - Serviço que detalha a empresa e o papel do usuário nesta empresa.
	 *
	 *
	 *
	 ************************************************************************************************
	 *
	 * Informações de uso
	 * ------------------
	 * Atribua às variáveis abaixo os valores de acordo com o seu sistema.
	 *
	 */


	private static final String URL_PROVIDER = "https://testescp-ecidadao.estaleiro.serpro.gov.br";
   private static final String URL_SERVICOS = "https://testeservicos-ecidadao.estaleiro.serpro.gov.br";
	private static final String REDIRECT_URI = "<coloque-aqui-a-uri>";		//redirectURI informada na chamada do serviço do authorize.
	private static final List<String> SCOPES = Arrays.asList("openid", "brasil_cidadao", "brasil_cidadao_empresa");		//Escopos cadastrados para a aplicação.
	private static final String CLIENT_ID = "<coloque-aqui-o-clientid-cadastrado-para-o-seu-sistema>";		//clientId informado na chamada do serviço do authorize.
```

```
	private static final String SECRET = "<coloque-aqui-o-secret-cadastrado-para-o-seu-sistema>";			//secret de conhecimento apenas do backend da aplicação.

	public static void main(String[] args) throws Exception {

		/**
		 *  Etapa 1: No Browser, chamar a URL do Authorize para recuperar o code e o state (opcional) conforme o exemplo abaixo:
		 *		https://testescp-ecidadao.estaleiro.serpro.gov.br/scp/authorize?response_type=code&client_id=<coloque-aqui-o-client-id>&scope=openid+brasil_cidadao+brasil_cidadao_empresa&redirect_uri=<coloque-aqui-a-uri-de-redirecionamento>&nonce=<coloque-aqui-um-numero-aleatorio>&state=<coloque-aqui-um-numero-aleatorio>
		 *		Descrição dos parametros:
		 *			response_type: Sempre "code";
		 *			client_id:	Identificador do sistema que usa o Brasil Cidadão. Este identificador é único para cada sistema;
		 *			scope:		Lista de escopos requisitados pelo sistema. Escopos são agrupamentos de informações cujo acesso deverá
		 *				ser autorizado pelo cidadão que acessa o sistema. Cada sistema deverá informar que conjunto de informações (escopos) deseja;
		 *		redirect_uri:  Uri para qual será feito o redirect após o login do cidadão (usuário). Para Celulares, usamos uma pseudo URI;
		 *		nonce: número aleatório;
		 *		state: número aleatório (opcional)
		 *
		 *		Observação: Sem o escopo "brasil_cidadao_empresa", não será possível utilizar o serviço de recuperação de informações de empresas.
		 */

		System.out.println("--------------------Etapa 1 - URL do Serviço Authorize------------------");
		System.out.println("Abra um Browser (Chrome ou Firefox), aperte F12. Clique na aba 'Network'.");
		System.out.println("Cole a URL abaixo no Browser (Chrome ou Firefox) e entre com um usuário cadastrado no Brasil Cidadão");
		System.out.println(URL_PROVIDER + "/scp/authorize?response_type=code&client_id=" + CLIENT_ID + "&scope=" + String.join("+", SCOPES) + "&redirect_uri=" + URLEncoder.encode(REDIRECT_URI, "UTF-8") + "&nonce=" + createRandomNumber() + "&state=" + createRandomNumber());

		/**
		 *  Etapa 2: De posse do code retornado pelo passo 1, chame o serviço para recuperar
```

os tokens disponíveis para sua aplicação (Access Token, Id Token e refresh Token (caso necessário)) conforme o exemplo abaixo.
*/

System.out.println("\n--------------------Etapa 2 - Recuperação dos Tokens de Acesso------------------");
System.out.println("Digite abaixo o parâmetro 'code' retornado pelo redirect da etapa 1");
System.out.print("Digite o valor do parâmetro code retornado:");
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
String code = br.readLine();

String tokens = getTokens(code);
System.out.println("JSON retornado:");
System.out.println(tokens);

JSONParser parser = new JSONParser();
JSONObject tokensJson = (JSONObject) parser.parse(tokens);

String accessToken = (String) tokensJson.get("access_token");
String idToken = (String) tokensJson.get("id_token");

/**
* Etapa 3: De posse do access token, podemos extrair algumas informações acerca do usuário. Aproveitamos também para checar a assinatura e tempo de expiração do token.
*	Para isso, este exemplo usa a biblioteca Open Source chamada "jose4j" mas qualquer outra biblioteca que implemente a especificação pode ser usada.
*
*	O Access Token fornece as seguintes informações acerca do usuário:
*		1- id client da aplicação à qual o usuário se autenticou;
*		2- Escopos requeridos pela aplicação autorizados pelo usuário;
*		3- CPF do usuário autenticado
*		4- Nome completo do usuário cadastrado no Brasil Cidadão. Atenção, este é o nome que foi fornecido pelo usuário no momento do seu cadastro
*
*/

JwtClaims jwtClaims;
try {
	jwtClaims = processToClaims(accessToken);
}catch(Exception e) {
	System.out.println("Access Token inválido!");

```
                throw new Exception(e);
            }

            String idClient = jwtClaims.getStringClaimValue("azp");          //Client Id
            List<String> scopes = jwtClaims.getStringListClaimValue("scope");    //Escopos autorizados pelo usuário
            String cpfDoUsuario = jwtClaims.getSubject();                    //CPF do usuário.
            String nomeCompleto = jwtClaims.getStringClaimValue("name");   //Nome Completo do cadastro feito pelo usuário no Brasil Cidadão.

            System.out.println("\n--------------------Etapa 3 - Informações obtidas do Access Token------------------");
            System.out.printf("O usuário %s, CPF %s foi autenticado pelo Brasil Cidadão por meio de %s para usar o sistema %s. Este usuário também autorizou este mesmo sistema à utilizar as informações representadas pelos escopos %s. \n", nomeCompleto, cpfDoUsuario, idClient, String.join(",", scopes) );

            /**
             * Serviço 1: De posse do access token, a aplicação pode chamar o serviço de recuperação de informações do usuário (getUserInfo).
             *
             */

            String infoUserJson = getUserInfo(accessToken, "brasil_cidadao");

            System.out.println("\n--------------------Serviço 1 - Informações do usuário obtidas da Receita Federal------------------");
            System.out.println("JSON retornado:");
            System.out.println(infoUserJson);


            /**
             * Serviço 2: De posse do access token, a aplicação pode chamar o serviço para saber quais selos o usuário logado possui.
             */

            String confiabilidadeJson = getConfiabilidade(accessToken);

            System.out.println("\n--------------------Serviço 2 - Informações acerca da confiabilidade do usuário------------------");
            System.out.println("JSON retornado:");
            System.out.println(confiabilidadeJson);
```

```
                List<Long> seloNivels = new ArrayList<Long>();
                for(Object o: (JSONArray) parser.parse(confiabilidadeJson)){
                   if ( o instanceof JSONObject ) {
                        seloNivels.add((Long) ((JSONObject) o).get("nivel"));
                   }
                }

                if (seloNivels.contains(new Long(11))) { //Selo de REPRESENTANTE E-CNPJ

                        /**
                         * Serviço 3: De posse do access token, a aplicação pode chamar o serviço para saber quais empresas se encontram vinculadas ao usuário logado.
                         *
                         */

                        String empresasJson = getEmpresasVinculadas(accessToken, "brasil_cidadao_empresa");

                        System.out.println("\n--------------------Serviço 3 - Empresas vinculadas ao usuário------------------");
                        System.out.println("JSON retornado:");
                        System.out.println(empresasJson);

                        /**
                         * Serviço 4: De posse do access token, a aplicação pode chamar o serviço para obter dados de uma empresa específica e o papel do usuário logado nesta empresa.
                         */

                        JSONObject empresasVinculadasJson = (JSONObject) parser.parse(empresasJson);
                        JSONArray cnpjs = (JSONArray ) empresasVinculadasJson.get("cnpjs");

                        if (!cnpjs.isEmpty()) {

                                String dadosEmpresaJson = getDadosEmpresa(accessToken, (String) cnpjs.get(0), "brasil_cidadao_empresa");

                                System.out.printf("\n--------------------Serviço 4 - Informações acerca da empresa %s------------------", cnpjs.get(0));
                                System.out.println("JSON retornado:");
                                System.out.println(dadosEmpresaJson);

                        }
```

```
                }
        }
        private static String getTokens(String code) throws Exception {
                String retorno = "";

                String redirectURIEncodedURL = URLEncoder.encode(REDIRECT_URI, "UTF-
8");

                URL url = new URL(URL_PROVIDER + "/scp/token?
grant_type=authorization_code&code=" + code + "&redirect_uri=" + redirectURIEncodedURL);
                HttpsURLConnection conn = (HttpsURLConnection) url.openConnection();
                conn.setRequestMethod("POST");
                conn.setRequestProperty("Accept", "application/json");
                conn.setRequestProperty("authorization", String.format("Basic %s",
Base64.getEncoder().encodeToString(String.format("%s:%s", CLIENT_ID,
SECRET).getBytes())));

                if (conn.getResponseCode() != 200) {
                        throw new RuntimeException("Falhou : HTTP error code : " +
conn.getResponseCode());
                }

                BufferedReader br = new BufferedReader(new
InputStreamReader((conn.getInputStream())));

                String tokens = null;
                while ((tokens = br.readLine()) != null) {
                        retorno += tokens;
                }

                conn.disconnect();

                return retorno;
        }
        private static JwtClaims processToClaims(String token) throws Exception {
                URL url = new URL(URL_PROVIDER + "/scp/jwk");
                HttpsURLConnection conn = (HttpsURLConnection) url.openConnection();
                conn.setRequestMethod("GET");
                conn.setRequestProperty("Accept", "application/json");
                if (conn.getResponseCode() != 200) {
```

```
                        throw new RuntimeException("Falhou : HTTP error code : " +
conn.getResponseCode());
                }

                BufferedReader br = new BufferedReader(new
InputStreamReader((conn.getInputStream())));

                String ln = null, jwk = "";
                while ((ln = br.readLine()) != null) {
                        jwk += ln;
                }

                conn.disconnect();

                JSONParser parser = new JSONParser();
                JSONObject tokensJson = (JSONObject) parser.parse(jwk);

                JSONArray keys =  (JSONArray) tokensJson.get("keys");

                JSONObject keyJSONObject = (JSONObject) keys.get(0);

                String key = keyJSONObject.toJSONString();

                PublicJsonWebKey pjwk = PublicJsonWebKey.Factory.newPublicJwk(key);

                JwtConsumer jwtConsumer = new JwtConsumerBuilder()
                .setRequireExpirationTime() // Exige que o token tenha um tempo de validade
                .setMaxFutureValidityInMinutes(60) // Testa se o tempo de validade do access token
é inferior ou igual ao tempo máximo estipulado (Tempo padrão de 60 minutos)
                .setAllowedClockSkewInSeconds(30) // Esta é uma boa prática.
                .setRequireSubject() // Exige que o token tenha um Subject.
                .setExpectedIssuer(URL_PROVIDER + "/scp/") // Verifica a procedência do token.
                .setVerificationKey(pjwk.getPublicKey()) // Verifica a assinatura com a public key
fornecida.
                .build(); // Cria a instância JwtConsumer.

                return jwtConsumer.processToClaims(token);
        }

        private static String getUserInfo(String accessToken, String scope) {
                String retorno = "";
                try {
                        URL url = new URL(URL_SERVICOS + "/servicos-
```

```
ecidadao/ecidadao/usuario/getUserInfo/" + scope);
                        HttpsURLConnection conn = (HttpsURLConnection) url.openConnection();
                        conn.setRequestMethod("GET");
                        conn.setRequestProperty("Accept", "application/json");
                        conn.setRequestProperty("authorization", accessToken);

                        if (conn.getResponseCode() != 200) {
                                throw new RuntimeException("Falhou : HTTP error code : " +
conn.getResponseCode());
                        }

                        String output;
                        BufferedReader br = new BufferedReader(new
InputStreamReader((conn.getInputStream())));

                        while ((output = br.readLine()) != null) {
                                retorno += output;
                        }

                        conn.disconnect();

                } catch (MalformedURLException e) {

                        e.printStackTrace();

                } catch (IOException e) {

                        e.printStackTrace();

                }
                return retorno;
        }

        private static String getEmpresasVinculadas(String accessToken, String scope) throws
Exception {
                String retorno = "";

                URL url = new URL(URL_SERVICOS + "/servicos-
ecidadao/ecidadao/empresa/escopo/" + scope);
                HttpsURLConnection conn = (HttpsURLConnection) url.openConnection();
                conn.setRequestMethod("GET");
                conn.setRequestProperty("Accept", "application/json");
                conn.setRequestProperty("authorization", accessToken);
```

```
		if (conn.getResponseCode() != 200) {
			throw new RuntimeException("Falhou : HTTP error code : " +
conn.getResponseCode());
		}

		String output;
		BufferedReader br = new BufferedReader(new
InputStreamReader((conn.getInputStream())));

		while ((output = br.readLine()) != null) {
			retorno += output;
		}

		conn.disconnect();

		return retorno;
	}

	private static String getDadosEmpresa(String accessToken, String cnpj, String scope) throws
Exception {
		String retorno = "";

		URL url = new URL(URL_SERVICOS + "/servicos-ecidadao/ecidadao/empresa/" +
cnpj + "/escopo/" + scope);
		HttpsURLConnection conn = (HttpsURLConnection) url.openConnection();
		conn.setRequestMethod("GET");
		conn.setRequestProperty("Accept", "application/json");
		conn.setRequestProperty("authorization", accessToken);

		if (conn.getResponseCode() != 200) {
			throw new RuntimeException("Falhou : HTTP error code : " +
conn.getResponseCode());
		}

		String output;
		BufferedReader br = new BufferedReader(new
InputStreamReader((conn.getInputStream())));

		while ((output = br.readLine()) != null) {
			retorno += output;
		}
```

```
                conn.disconnect();

                return retorno;
        }

        private static String getConfiabilidade(String accessToken) throws Exception {
                String retorno = "";

                URL url = new URL(URL_SERVICOS + "/servicos-
ecidadao/ecidadao/usuario/getConfiabilidade");
                HttpsURLConnection conn = (HttpsURLConnection) url.openConnection();
                conn.setRequestMethod("GET");
                conn.setRequestProperty("Accept", "application/json");
                conn.setRequestProperty("Authorization", accessToken);

                if (conn.getResponseCode() != 200) {
                        throw new RuntimeException("Falhou : HTTP error code : " +
conn.getResponseCode());
                }

                String output;
                BufferedReader br = new BufferedReader(new
InputStreamReader((conn.getInputStream())));

                while ((output = br.readLine()) != null) {
                        retorno += output;
                }

                conn.disconnect();

                return retorno;
        }

        private static String createRandomNumber() {
                return new BigInteger(50, new SecureRandom()).toString(16);

        }

}
```

PHP

1. Arquivo CSS:

```
* {
   box-sizing: border-box;
}

body {
      font-family: Arial, Helvetica, sans-serif;
      margin: 0;
}

.header {
      padding: 20px;
      text-align: center;
      background: rgb(240, 242, 241);
      color: rgb(51, 51, 51);
}

.header h1 {
      font-size: 40px;
}

.navbar {
      overflow: hidden;
      background-color: #333;
      position: sticky;
      position: -webkit-sticky;
      top: 0;
}

.navbar a {
      float: left;
      display: block;
      color: white;
      text-align: center;
      padding: 14px 20px;
      text-decoration: none;
}

.navbar a.right {
      float: right;
}

.navbar a:hover {
      background-color: #ddd;
```

```
        color: black;
}

.navbar a.active {
        background-color: #666;
        color: white;
}

.row {
            display: -ms-flexbox; /* IE10 */
            display: -webkit-box;                 /* chrome */
            -webkit-justify-content: space-around; /* chrome */
            -webkit-flex-flow: row wrap;          /* chrome */
            -webkit-align-items: stretch;         /* chrome */
            display: flex;
            -ms-flex-wrap: wrap; /* IE10 */
            flex-wrap: wrap;
}

.left_side {
    -ms-flex: 30%; /* IE10 */
    flex: 30%;
            width: 30%; /* chrome */
    background-color: #f1f1f1;
    padding: 20px;
}

.right_side {
    -ms-flex: 70%; /* IE10 */
    flex: 70%;
            width: 70%; /* chrome */
    background-color: white;
    padding: 20px;
}

.result {
    background-color: #aaa;
    width: 100%;
    padding: 20px;
}

.resultValido {
            background-color: green;
```

```
    width: 100%;
    padding: 20px;
}

.resultInvalido {
            background-color: red;
    width: 100%;
    padding: 20px;
}

/* Footer */
.footer {
    padding: 20px;
    text-align: center;
    background: #ddd;
}

/* Responsive layout - when the screen is less than 700px wide, make the two columns stack on top of each other instead of next to each other */
@media screen and (max-width: 700px) {
    .row {
        flex-direction: column;
    }
}

/* Responsive layout - when the screen is less than 400px wide, make the navigation links stack on top of each other instead of next to each other */
@media screen and (max-width: 400px) {
    .navbar a {
        float: none;
        width: 100%;
    }
}

pre {
    white-space: pre-wrap;       /* css-3 */
    white-space: -moz-pre-wrap;  /* Mozilla, since 1999 */
    white-space: -pre-wrap;      /* Opera 4-6 */
    white-space: -o-pre-wrap;    /* Opera 7 */
    word-wrap: break-word;       /* Internet Explorer 5.5+ */
  }

/* Center the loader */
```

```
#loader {
  position: absolute;
  left: 50%;
  top: 50%;
  z-index: 1;
  width: 150px;
  height: 150px;
  margin: -75px 0 0 -75px;
  border: 16px solid #f3f3f3;
  border-radius: 50%;
  border-top: 16px solid #3498db;
  width: 120px;
  height: 120px;
  -webkit-animation: spin 2s linear infinite;
  animation: spin 2s linear infinite;
}

@-webkit-keyframes spin {
  0% { -webkit-transform: rotate(0deg); }
  100% { -webkit-transform: rotate(360deg); }
}

@keyframes spin {
  0% { transform: rotate(0deg); }
  100% { transform: rotate(360deg); }
}
```

2. Arquivo PHP

```
<?php

        /**
         *
         * O presente código tem por objetivo exemplificar de forma minimalista o consumo dos serviços utilizados pelo Brasil Cidadão.
         *
         */

        use \Firebase\JWT\JWT;

        $URL_PROVIDER="https://testescp-ecidadao.estaleiro.serpro.gov.br";
   $CLIENT_ID = "<coloque-aqui-o-clientid-cadastrado-para-o-seu-sistema>";
   $SECRET = "<coloque-aqui-o-secret-cadastrado-para-o-seu-sistema>";
```

```
$REDIRECT_URI = "<coloque-aqui-a-uri>";
        $SCOPE = "openid+brasil_cidadao+brasil_cidadao_empresa";
$URL_SERVICOS="https://testeservicos-ecidadao.estaleiro.serpro.gov.br";

        /*
         *  Etapa 1: No Browser, chamar a URL do Authorize para recuperar o code e o state (opcional) conforme o exemplo abaixo:
         *                  https://testescp-ecidadao.estaleiro.serpro.gov.br/scp/authorize?response_type=code&client_id=<coloque-aqui-o-client-id>&scope=openid+brasil_cidadao+brasil_cidadao_empresa&redirect_uri=<coloque-aqui-a-uri-de-redirecionamento>&nonce=<coloque-aqui-um-numero-aleatorio>&state=<coloque-aqui-um-numero-aleatorio>
         *                  Descrição dos parametros:
         *                          response_type: Sempre "code";
         *                          client_id:     Identificador do sistema que usa o Brasil Cidadão. Este identificador é único para cada sistema;
         *                          scope:         Lista de escopos requisitados pelo sistema. Escopos são agrupamentos de informações cujo acesso deverá
         *                                                  ser autorizado pelo cidadão que acessa o sistema. Cada sistema deverá informar que conjunto de informações (escopos) deseja;
         *          redirect_uri:  Uri para qual será feito o redirect após o login do cidadão (usuário). Para Celulares, usamos uma pseudo URI;
         *          nonce: número aleatório;
         *          state: número aleatório (opcional)
         *
         *                  Observação: Sem o escopo "brasil_cidadao_empresa", não será possível utilizar o serviço de recuperação de informações de empresas.
         */

$uri = $URL_PROVIDER . "/scp/authorize?response_type=code"
. "&client_id=". $CLIENT_ID
. "&scope=" . $SCOPE
. "&redirect_uri=" . urlencode($REDIRECT_URI)
. "&nonce=" . getRandomHex()
. "&state=" . getRandomHex();

function getRandomHex($num_bytes=4) {
    return bin2hex(openssl_random_pseudo_bytes($num_bytes));
}

        /*
      Etapa 2: De posse do code retornado pelo passo 1, chame o serviço para recuperar os tokens disponíveis para sua aplicação
```

```
        (Access Token, Id Token e refresh Token (caso necessário)) conforme o exemplo abaixo.
    */
    $CODE = $_REQUEST["code"];
    $STATE = $_REQUEST["state"];

        if (isset($CODE)) {

            $campos = array(
                    'grant_type' => urlencode('authorization_code'),
                    'code' => urlencode($CODE),
                    'redirect_uri' => urlencode($REDIRECT_URI)
                    );
            foreach($campos as $key=>$value) {
                    $fields_string .= $key.'='.$value.'&';
            }
            rtrim($fields_string, '&');
            $ch_token = curl_init();
            curl_setopt($ch_token,CURLOPT_URL, $URL_PROVIDER . "/scp/token" );
            curl_setopt($ch_token,CURLOPT_POST, count($fields));
            curl_setopt($ch_token,CURLOPT_POSTFIELDS, $fields_string);
            curl_setopt($ch_token, CURLOPT_RETURNTRANSFER, TRUE);
            curl_setopt($ch_token,CURLOPT_SSL_VERIFYPEER, false);
            $headers = array(
                    'Content-Type:application/x-www-form-urlencoded',
                    'Authorization: Basic '. base64_encode($CLIENT_ID.":".$SECRET)
            );
            curl_setopt($ch_token, CURLOPT_HTTPHEADER, $headers);
            $json_output_tokens = json_decode(curl_exec($ch_token), true);
            curl_close($ch_token);

            /**
             * Etapa 3: De posse do access token, podemos extrair algumas informações acerca
do usuário. Aproveitamos também para checar a assinatura e tempo de expiração do token.
             *      Para isso, este exemplo usa a biblioteca chamada "firebase/php-jwt" mas
qualquer outra biblioteca que implemente a especificação pode ser usada.
             *
             *      O Access Token fornece as seguintes informações acerca do usuário:
             *                      1- id client da aplicação à qual o usuário se autenticou;
             *                      2- Escopos requeridos pela aplicação autorizados pelo
usuário;
             *                      3- CPF do usuário autenticado
             *                      4- Nome completo do usuário cadastrado no Brasil
Cidadão. Atenção, este é o nome que foi fornecido pelo usuário no momento do seu cadastro
```

```
                *               (ou obtido do Certificado Digital e-CPF caso o cadastro tenha sido feito por este meio). O Serviço getUserInfo obtém as informações do
                *               usuário direto da Receita Federal.
                */
                $url = $URL_PROVIDER . "/scp/jwk" ;
                $ch_jwk = curl_init();
                curl_setopt($ch_jwk,CURLOPT_SSL_VERIFYPEER, false);
                curl_setopt($ch_jwk,CURLOPT_URL, $url);
                curl_setopt($ch_jwk, CURLOPT_RETURNTRANSFER, TRUE);
                $json_output_jwk = json_decode(curl_exec($ch_jwk), true);
                curl_close($ch_jwk);

                $access_token = $json_output_tokens['access_token'];

                try{
                        $json_output_payload_access_token = processToClaims($access_token, $json_output_jwk);
                } catch (Exception $e) {
                        $detalhamentoErro = $e;
                }

                /*
                        Serviço de obtenção cadastro do usuário: De posse do access token, a aplicação pode chamar o serviço de recuperação de informações do usuário (getUserInfo) conforme o exemplo abaixo.
                */
                $url = $URL_SERVICOS . "/servicos-ecidadao/ecidadao/usuario/getUserInfo/brasil_cidadao" ;
                $ch_user_info = curl_init();
                curl_setopt($ch_user_info,CURLOPT_SSL_VERIFYPEER, false);
                curl_setopt($ch_user_info,CURLOPT_URL, $url);
                curl_setopt($ch_user_info, CURLOPT_RETURNTRANSFER, TRUE);
                $headers = array(
                                'Authorization: '. $json_output_tokens['access_token']
                );
                curl_setopt($ch_user_info, CURLOPT_HTTPHEADER, $headers);
                $json_output_user_info = json_decode(curl_exec($ch_user_info), true);
                curl_close($ch_user_info);

                /*
                        Serviço de obtenção de selos de Confiabilidade: De posse do access token, a aplicação pode chamar o serviço para saber quais selos o usuário logado possui.
                */
```

```
		$ch_confiabilidade = curl_init();
		curl_setopt($ch_confiabilidade,CURLOPT_SSL_VERIFYPEER, false);
		curl_setopt($ch_confiabilidade,CURLOPT_URL, $URL_SERVICOS . "/servicos-ecidadao/ecidadao/usuario/getConfiabilidade");
		curl_setopt($ch_confiabilidade, CURLOPT_RETURNTRANSFER, TRUE);
		$headers = array(
				'Accept: application/json',
				'Authorization: '. $json_output_tokens['access_token']
		);
		curl_setopt($ch_confiabilidade, CURLOPT_HTTPHEADER, $headers);
		$json_output_confiabilidade = json_decode(curl_exec($ch_confiabilidade), true);
		curl_close($ch_confiabilidade);

		/*
			Verificar se CPF autenticado possui selo de Confiabildidade e-CNPJ.
		*/
		if ($json_output_confiabilidade['nivel'] == '11') {
			/*
				Serviço de recuperação de empresas vinculadas: De posse do access token, a aplicação pode chamar o serviço para saber quais empresas se encontram vinculadas ao usuário logado.
			*/
			$ch_empresas_vinculadas = curl_init();
			curl_setopt($ch_empresas_vinculadas,CURLOPT_SSL_VERIFYPEER, false);
			curl_setopt($ch_empresas_vinculadas,CURLOPT_URL, $URL_SERVICOS . "/servicos-ecidadao/ecidadao/empresa/escopo/brasil_cidadao_empresa");
			curl_setopt($ch_empresas_vinculadas, CURLOPT_RETURNTRANSFER, TRUE);
			$headers = array(
					'Accept: application/json',
					'Authorization: '. $json_output_tokens['access_token']
			);
			curl_setopt($ch_empresas_vinculadas, CURLOPT_HTTPHEADER, $headers);
			$json_output_empresas_vinculadas = json_decode(curl_exec($ch_empresas_vinculadas), true);
			curl_close($ch_empresas_vinculadas);

			/*
				Serviço de detalhamento da empresa vinculada: De posse do access token, a aplicação pode chamar o serviço para obter dados de uma empresa específica e o papel do usuário logado nesta empresa.
```

```
                        */
                        $cnpj = $json_output_empresas_vinculadas[0];
                        $ch_papel_empresa = curl_init();
                        curl_setopt($ch_papel_empresa,CURLOPT_SSL_VERIFYPEER, false);
                        curl_setopt($ch_papel_empresa,CURLOPT_URL, $URL_SERVICOS .
"/servicos-ecidadao/ecidadao/empresa/" . $cnpj . "/escopo/brasil_cidadao_empresa");
                        curl_setopt($ch_papel_empresa, CURLOPT_RETURNTRANSFER, TRUE);
                        $headers = array(
                                'Accept: application/json',
                                'Authorization: '. $json_output_tokens['access_token']
                        );
                        curl_setopt($ch_papel_empresa, CURLOPT_HTTPHEADER, $headers);
                        $json_output_papel_empresa = json_decode(curl_exec($ch_papel_empresa),
true);
                        curl_close($ch_papel_empresa);
                }
        }
        /**
         * Função que valida o access token (Valida o tempo de expiração e a assinatura)
         *
         */
        function processToClaims($access_token, $jwk)
        {
                $modulus = JWT::urlsafeB64Decode($jwk['keys'][0]['n']);
                $publicExponent = JWT::urlsafeB64Decode($jwk['keys'][0]['e']);
                $components = array(
                        'modulus' => pack('Ca*a*', 2, encodeLength(strlen($modulus)), $modulus),
                        'publicExponent' => pack('Ca*a*', 2,
encodeLength(strlen($publicExponent)), $publicExponent)
                );
                $RSAPublicKey = pack(
                        'Ca*a*a*',
                        48,
                        encodeLength(strlen($components['modulus']) +
strlen($components['publicExponent'])),
                        $components['modulus'],
                        $components['publicExponent']
                );
                $rsaOID = pack('H*', '300d06092a864886f70d0101010500'); // hex version of
MA0GCSqGSIb3DQEBAQUA
                $RSAPublicKey = chr(0) . $RSAPublicKey;
                $RSAPublicKey = chr(3) . encodeLength(strlen($RSAPublicKey)) .
$RSAPublicKey;
```

```
                $RSAPublicKey = pack(
                        'Ca*a*',
                        48,
                        encodeLength(strlen($rsaOID . $RSAPublicKey)),
                        $rsaOID . $RSAPublicKey
                );
                $RSAPublicKey = "-----BEGIN PUBLIC KEY-----\r\n" .
chunk_split(base64_encode($RSAPublicKey), 64) . '-----END PUBLIC KEY-----';

                JWT::$leeway = 3 * 60; //em segundos

                $decoded = JWT::decode($access_token, $RSAPublicKey, array('RS256'));

                return (array) $decoded;
        }

        function encodeLength($length)
        {
                if ($length <= 0x7F) {
                        return chr($length);
                }
                $temp = ltrim(pack('N', $length), chr(0));
                return pack('Ca*', 0x80 | strlen($temp), $temp);
        }
?>

<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <meta http-equiv="X-UA-Compatible"
 content="ie=edge">
   <title>STI Brasil Cidadao</title>
   <link rel="stylesheet" type="text/css" href="css/sti.css">
        <script>
                function waiting() {
                        document.getElementById("loader").style.display = "block";
                }
        </script>
</head>
<body>
```

```
    <div class="header">
        <h1>STI Brasil Cidadão</h1>
        <p><b>S</b>ite de <b>T</b>este <b>I</b>ntegrado ao Brasil Cidadão</p>
    </div>

    <div class="navbar">
                    <?php
                            if (isset($json_output_payload_access_token)) {
                                    echo '<a href="#" class="right">Logout</a>';
                            } else {
                                    echo '<a href="' . $uri .'" onClick="waiting();" class="right">Logar
com o Brasil Cidadão</a>';
                            }
                    ?>
    </div>

            <div id="loader" style="display:none"></div>

    <div class="row">
        <div class="left_side">
            <div>
                <h3>Etapa 1 (obrigatório): Autenticação</h3>
                <p>Ao clicar no botão "Logar com o Brasil Cidadão" a seguinte URL será chamada:</p>
            </div>
        </div>
        <div class="right_side">
            <h3>URL do Serviço Authorize:</h3>
                            <div class="result" style="height:200px;">
                                    <pre><?php echo $uri ?></pre>
                            </div>
        </div>
    </div>

            <?php
                    if (isset($json_output_tokens)) {
            ?>
                    <div class="row">
                            <div class="left_side">
                                    <div>
                                            <h3>Etapa 2 (obrigatório): Recuperar os Tokens</h3>
                                            <p>De posse do code retornado pelo passo 1, chame o serviço
para recuperar os tokens disponíveis para sua aplicação
                                              (Access Token, Id Token e refresh Token (caso
```

```
necessário)):</p>
                        </div>
                    </div>
                    <div class="right_side">
                        <h3>Json:</h3>
                        <div class="result" style="width:900px;">
                            <pre><?php echo json_encode($json_output_tokens,
JSON_PRETTY_PRINT); ?></pre>
                        </div>
                    </div>
                </div>

                <div class="row">
                    <div class="left_side">
                        <div>
                            <h3>Etapa 3 (desejável): Validação do Access Token</h3>
                            <p>De posse do access token, podemos extrair algumas
informações acerca do usuário. Aproveitamos também para checar a assinatura e tempo de
expiração do token:</p>
                        </div>
                    </div>
                    <div class="right_side">
                        <?php
                            if (isset($json_output_payload_access_token)) {
                        ?>
                            <h3>Json:</h3>
                            <div class="result" style="width:900px;">
                                <pre><?php echo
json_encode($json_output_payload_access_token, JSON_PRETTY_PRINT); ?></pre>
                            </div>
                            <div id="result-access_token" class="resultValido"
style="width:900px;">
                                <pre><b>Access Token VALIDO</b></pre>
                            </div>
                        <?php
                            } else {
                        ?>
                            <h3>Access Token:</h3>
                            <div class="result" style="width:900px;">
                                <pre><?php echo $access_token; ?></pre>
                            </div>
                            <div id="result-access_token" class="resultInvalido"
style="width:900px;">
```

```
                                        <pre><b>Access Token INVÁLIDO</b></pre>
                                    </div>
                                    <div class="result" style="width:900px;">
                                        <pre>Detalhamento: <?php echo $detalhamentoErro; ?
></pre>
                                    </div>

                                <?php
                                    }
                                ?>
                            </div>
                        </div>
                <?php
                    }
                    if (isset($json_output_payload_access_token)) {
                ?>
                    <div class="row">
                        <div class="left_side">
                            <div>
                                <h3>Serviço: Recuperar Informações do Usuário</h3>
                                <p>De posse do access token, a aplicação pode chamar o
serviço de recuperação de informações do usuário (getUserInfo):</p>
                            </div>
                        </div>
                        <div class="right_side">
                            <h3>Json:</h3>
                            <div class="result" style="width:900px;">
                                <pre><?php echo json_encode($json_output_user_info,
JSON_PRETTY_PRINT); ?></pre>
                            </div>
                        </div>
                    </div>

                    <div class="row">
                        <div class="left_side">
                            <div>
                                <h3>Serviço: Recuperar Selos do Usuário</h3>
                                <p>De posse do access token, a aplicação pode chamar o
serviço para saber quais selos o usuário logado possui:</p>
                            </div>
                        </div>
                        <div class="right_side">
                            <h3>Json:</h3>
```

```
                        <div class="result" style="width:900px;">
                            <pre><?php echo json_encode($json_output_confiabilidade,
JSON_PRETTY_PRINT); ?></pre>
                        </div>
                    </div>
                </div>
                <?php
                    if ($json_output_confiabilidade['nivel'] == '11') {
                ?>
                    <div class="row">
                        <div class="left_side">
                            <div>
                                <h3>Serviço: Recuperar Vinculos com empresas</h3>
                                <p>De posse do access token, a aplicação pode chamar
o serviço para saber quais empresas se encontram vinculadas ao usuário logado:</p>
                            </div>
                        </div>
                        <div class="right_side">
                            <h3>Json:</h3>
                            <div class="result" style="width:900px;">
                                <pre><?php echo
json_encode($json_output_empresas_vinculadas, JSON_PRETTY_PRINT); ?></pre>
                            </div>
                        </div>
                    </div>

                    <div class="row">
                        <div class="left_side">
                            <div>
                                <h3>Serviço: Recuperar Dados de Empresa</h3>
                                <p>De posse do access token, a aplicação pode chamar
o serviço para obter dados de uma empresa específica e o papel do usuário logado nesta
empresa:</p>
                            </div>
                        </div>
                        <div class="right_side">
                            <h3>Json:</h3>
                            <?php
                                if (empty($json_output_empresas_vinculadas['cnpjs']))
{
                                    echo '<div class="result"
style="width:900px;"><pre>Não há empresas a detalhar.</pre></div>';
                                }
```

```
                                        foreach ($json_output_empresas_vinculadas['cnpjs'] as
$empresa) {
                                                echo '<div class="result"
style="width:900px;"><pre>' . json_encode($json_output_papel_empresa,
JSON_PRETTY_PRINT) . '</pre></div>';
                                        }
                                    ?>
                                </div>
                            </div>
                        <?php
                            }
                        ?>

                <?php
                    }
                ?>
</body>
</html>
```

Sites Úteis

- ***OpenId Connect:*** http://openid.net/developers/specs/
- ***OAuth 2.0:*** https://oauth.net/2/
- ***Tutotial Oauth:*** https://www.digitalocean.com/community/tutorials/anintroduction-to-oauth-2
- ***JWT:*** https://jwt.io/introduction/
- ***Tutorial JWT:*** https://rafaell-lycan.com/2016/autenticacao-jwt-angular-app/
- ***Transformador Base64*:** http://www.motobit.com/util/base64-decoderencoder.asp
- ***Plataforma de Cidadania Digital:*** http://www.planejamento.gov.br/cidadaniadigital