Centro Paula Souza

AUTENTICADOR CPS

Serviço de Autenticação para usuários Microsoft e Aplicações CPS

Manual do Desenvolvedor

Divisão de Informática

Tarcísio de Freitas
Governador

Felício Ramuth
Vice-Governador

Vahan Agopyan
Secretário de Ciência, Tecnologia e Inovação

CENTRO PAULA SOUZA

Laura Laganá
Diretora-Superintendente

Emilena Lorenzon Bianco
Vice-Diretora-Superintendente

Armando Natal Maurício
Chefe de Gabinete da Superintendência

Almério Melquíades de Araújo
Coordenador do Ensino Médio e Técnico

Rafael Ferreira Alves
Coordenador do Ensino Superior de Graduação

Douglas Hamilton Oliveira
Diretor da Divisão de Informática

Vicente Mellone Junior
Coordenador de Recursos Humanos

Marisa Souza
Coordenadora de Formação Inicial e Educação Continuada

Dirce Helena Salles
Coordenadora da Assessoria de Comunicação

Helena Gemignani Peterossi
Coordenadora da Pós-Graduação, Extensão e Pesquisa

Bruna Fernanda Ferreira
Coordenadora de Infraestrutura

Magda de Oliveira Vieira
Coordenadora de Gestão Administrativa e Financeira

INDICE

RESUMO

Gerenciar o controle de usuários é essencial para garantir a segurança dos sistemas, especialmente em ambientes que evoluíram do modelo desktop para sistemas web ou aplicativos móveis. Este documento descreve de maneira sucinta os principais conceitos envolvidos na etapa inicial de autorização, no contexto da estrutura de um serviço de autenticação dos usuários das diversas aplicações do Centro Paula Souza (CPS).

Em uma proposta de ambiente centralizado de autenticação, estão envolvidos o recurso de identificação de usuários, chamado Provedor de Identidade (Identity Provider = IdP), e uma aplicação que se comunica com o IdP por meio de protocolos de autenticação, como SAML, OpenID ou OAuth2. Nesse contexto, a integração e padronização do formato da rotina de login de usuários são cruciais, representando o primeiro passo na validação da identidade do usuário e na subsequente obtenção de suas permissões de acesso.

Palavras-chave: aplicação; autenticador; identity provider; login; padronização.

INFORMAÇÕES DO DOCUMENTO

Título do Manual: **Autenticador CPS**

Área responsável: **Divisão de Informática - Arquitetura**

Data da criação: **17/07/2023**

Data da atualização: **18/07/2024**

Público-alvo: Desenvolvedores

1 INTRODUÇÃO

No contexto de integração de sistemas, uma das operações destacadas pela Divisão de Informática (DI) do Centro Paula Souza, foi o trabalho para fornecer um login único para o usuário entre os diferentes sistemas existentes. Foi para solucionar essa demanda que surgiu o Autenticador CPS.

Em um primeiro momento, o foco esteve em oferecer condições para que as aplicações pudessem reconhecer um usuário autenticado por uma base única, e optou-se por utilizar o Identity Provider da Microsoft, que já estava presente no ambiente do CPS. Na sequência, o processo de autenticação foi integrado ao sistema Sysmail para oferecer também às aplicações o CPF do usuário, indicando uma identificação única, independente dos domínios (tenants Microsoft @cps, @fatec, @etec...) que o usuário tenha utilizado para realizar sua autenticação.

Após essa primeira etapa, surgiu uma nova demanda por melhorias de segurança na própria integração interna dos dados e no processo de comunicação entre as aplicações do CPS. Para essa demanda, o Autenticador incorporou rotinas que tratam da autenticação de aplicações. Cada aplicação recebe um registro com nome e senha, a exemplo do que acontece com um usuário (pessoa) e, uma vez validadas as informações, o Autenticador devolve um *token* para a aplicação. De posse desse *token,* um determinado sistema pode enviar requisições para outro sistema. A aplicação que recebe a requisição com o *token* solicita ao Autenticador a verificação do mesmo e obtém como retorno a indicação se aquela credencial é válida ou não.

Todo esse processo foi estabelecido para complementar questões de segurança, contribuindo para um processo de autenticação segura.

Um processo de autenticação programado de forma segura vai impedir que uma credencial vazada cause maiores danos. No entanto, um processo de autenticação baseado apenas na identificação de senha e usuário já deixou de ser seguro há muito tempo. Existem formas de roubar a credencial do usuário, sendo as mais ‘simples’ a instalação de malwares que interceptam a digitação da pessoa, por exemplo.

O uso dos sistemas corporativos em equipamentos não protegidos, como celulares ou computadores pessoais que podem não estar protegidos por um antivírus (endpoint) adequado, pode facilitar ataques ao atacante.

Segundo a OWASP, o processo de autenticação com **Multifactor Authentication Protection** é o cenário que proporciona melhor segurança no processo.

> *MFA is by far the best defense against the majority of password-related attacks, including brute-force, credential stuffing and password spraying, with analysis by Microsoft suggesting that it would have stopped 99.9% of account compromises.*
> *https://cheatsheetseries.owasp.org/cheatsheets/Multifactor_Authentication_Cheat_Sheet.html*

Dessa forma, para evitar maiores danos na operação dos sistemas, será necessário estabelecer processos mais seguros, mesmo que isso aumente a complexidade da operação para o usuário.

> ***Caso real:*** *Um usuário recebe um e-mail com um anexo PDF e o abre em sua máquina. Sem um antivírus atualizado, ele não percebe que um vírus do tipo Trojan foi instalado em seu computador. Com o tempo, ele acessa todos os sistemas normalmente, sem saber que suas credenciais de acesso estão sendo enviadas ao atacante. O atacante, tenta acessar todos os sistemas do usuário, já que o Trojan envia o link do sistema, o usuário e a senha. Nos sistemas mais protegidos, como o aplicativo bancário, o atacante não consegue acesso porque o MFA estar habilitado. No entanto, em um sistema, o atacante consegue acessar e invade-o por meio da injeção de scripts maliciosos no servidor. Uma sequência de eventos catastróficos ocorre devido à programação insegura, má configuração e falta de monitoramento.*
>
> ***Resultado:*** *uma simples credencial interrompe a operação de um importante sistema da instituição. Onde aconteceu? Aqui no CPS.*

Não é tão simples quanto parece, pois a presença de vírus pode destruir todas as proteções possíveis. Uma senha forte, por exemplo, ajuda em situações em que o atacante tenta roubar a credencial por tentativas repetidas, mas não adianta se o usuário foi vítima de um malware que rouba tudo o que ele digita. Ativar o múltiplo fator de autenticação é uma solução eficaz, pois, mesmo com o nome de usuário e a senha, o atacante não conseguirá entrar no sistema.

Mesmo assim, a campanha para trocar e manter uma senha forte é importante para evitar outros problemas, assim como utilizar senhas diferentes para cada serviço ou sistema.

Este documento tem por objetivo descrever o funcionamento e os recursos do Autenticador CPS.

2 SERVIÇO DE AUTENTICAÇÃO

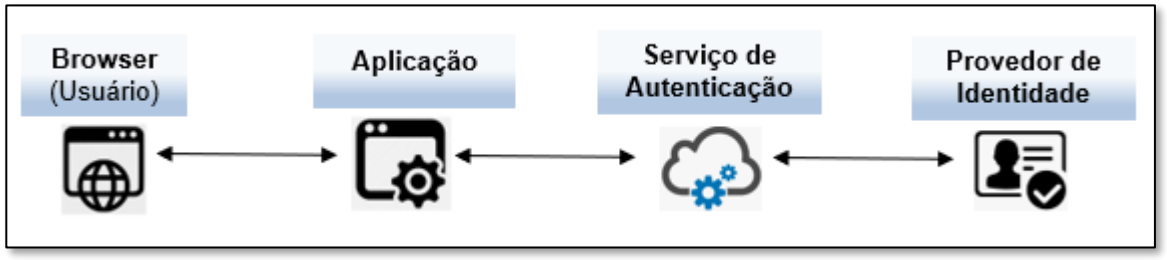
O processo de comunicação entre a aplicação e o IdP depende de um relacionamento de confiança estabelecido, na prática, por meio do registro da aplicação e do fornecimento da identificação do aplicativo, da identificação da organização, da senha e das chaves de acesso, para que o IdP possa responder às chamadas realizadas por meio do protocolo.

Em um ambiente corporativo, com um número considerável de aplicações, gerenciar seus registros e mantê-los atualizados em caso de novas aplicações ou encerramento do ciclo de vida é uma tarefa exaustiva. Além disso, o registro individualizado de cada aplicação exigiria que cada uma delas implementasse, conforme sua linguagem de programação, o tratamento do processo de comunicação do protocolo de autenticação com o IdP, o que pode se tornar improdutivo do ponto de vista do tempo e esforço dessa implementação.

Neste cenário, faz sentido que exista, entre o IdP e a aplicação, uma camada responsável pelo tratamento do protocolo e da comunicação com o IdP, tornando mais simples o uso dos recursos de *login*, verificação de *tokens* e *logout*. Essa é a tarefa do serviço de autenticação, tratado neste documento como Autenticador CPS.

A *Figura 1* ilustra o funcionamento de um ambiente em que a aplicação recebe uma solicitação do usuário para uma atividade que envolve o IdP. Neste momento, a aplicação encaminha a requisição para o Serviço de Autenticação, que, por sua vez, realiza a comunicação com o IdP e devolve à aplicação o resultado da requisição.

Figura 1 - Fluxo com Serviço de Autenticação



2.1 AUTENTICADOR MICROSOFT IDENTITY PROVIDER

O Autenticador CPS foi desenvolvido para desempenhar o papel do Serviço de Autenticação apresentado na *Figura 1*. Estruturalmente, é uma aplicação nos moldes de uma API (*Application Programming Interface*) baseada nos princípios do padrão REST (*Representational State Transfer*). Na prática, isso significa que o Autenticador disponibilizará endereços web (rotas) que seguirão métodos e padrões definidos ao receber requisições nesses endereços. Dessa forma, um endereço como, por exemplo, `https://esb.cps.sp.gov.br/sso-back/login` pode estar preparado para receber uma requisição do método GET e devolver a tela de login do usuário.

As rotas disponibilizadas pelo Autenticador até o presente momento são apresentadas no *Quadro 1*:

Quadro 1 - **Rotas do Autenticador**

Rota	Método	Ação
/login	GET	Executa chamada para que o IdP inicie o processo de login do usuário
/verifying-token	GET	Compara o token enviado como parâmetro com o gerado no momento do login. Retorno será inválido para os casos em que o token não mais existe ou está diferente do estado original
/verifying-token-deeply	GET	Verfica a assinatura digital do token enviado como parâmetro retornando válido quando a assinatura for reconhecida
/logout	DELETE	Remove disponibilidade do token existente para o usuário

Nos itens a seguir, serão abordados com mais detalhes os recursos disponíveis nos endereços de chamada.

2.1.1 Utilizando a rota */ login*

O processo de login começa com o envio de uma requisição para a rota '`/login`' do endereço web do Autenticador, incluindo a versão desejada. Além disso, é necessário informar dois parâmetros: o endereço para o qual o Autenticador deve encaminhar a chamada em caso de validação do *login* pelo IdProvider e o endereço a ser acionado se o login falhar. *O Quadro 2* detalha os itens da rota de login.

Quadro 2 - Requisição de login

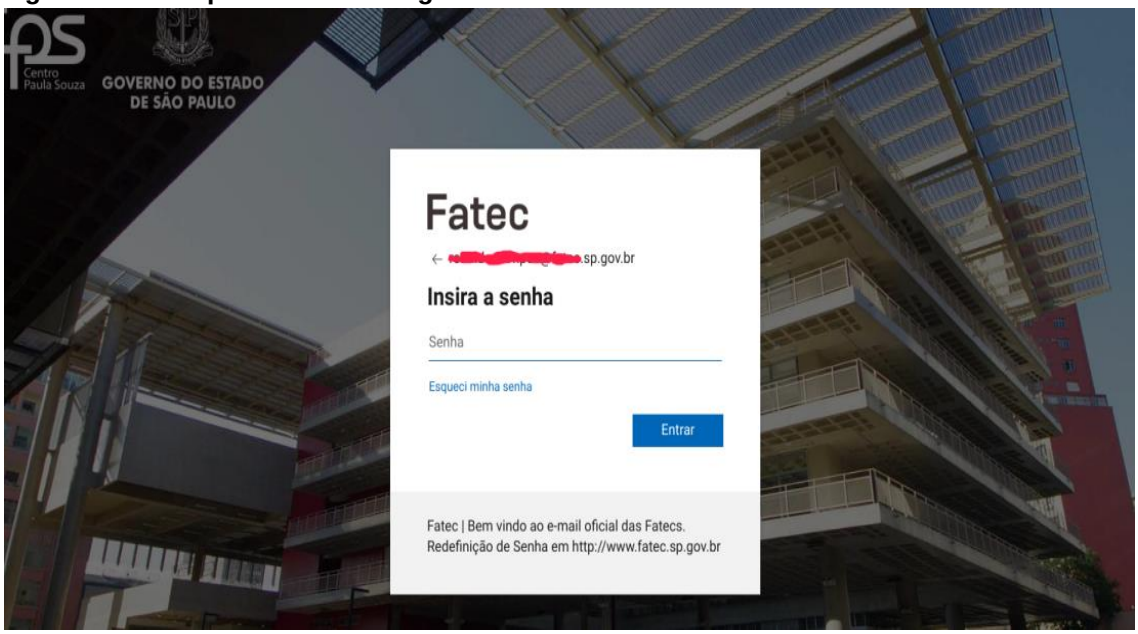
Item	**Descrição**
Method	**GET**
Base URL	https://esb.cps.sp.gov.br/sso-back/
Version	**v1**
Complete URL	https://esb.cps.sp.gov.br/sso-back/v1/login
Query params	1 - **callback**: endereço (url) na qual a aplicação tratará o *JWT token* retornado em caso de sucesso 2 - **callback_error**: endereço (url) da aplicação para receber a mensagem de erro em caso de falha do login

Considerando uma aplicação que deseja solicitar o login para seus usuários, alocada na URL '*http://meusite.com.br'*, e com os caminhos '*/logged'* para tratar o login bem-sucedido e '*/logerror'* para os casos de falha, a URL com os parâmetros seria:

```
https://esb.cps.sp.gov.br/sso-back/v1/login?callback=http://meusite.com.br/logged&callback_error=http://meusite.com.br/logerror
```

Essa chamada direcionará o usuário para a tela de login do *Id Provider da Microsoft,* conforme ilustrado na *Figura 2.*

Figura 2 - Exemplo de tela de Login



O tratamento do retorno do login é feito manipulando os dados recebidos na URL fornecida. Se o login for válido, a aplicação receberá um *token* no formato *JWT*, que poderá ser decodificado para acessar o nome de usuário e outras informações necessárias. (Mais detalhes sobre o *token JWT* estão disponíveis na seção "*Estrutura JWT"* deste documento).

Dica: Algumas linguagens de programação oferecem recursos nativos para o tratamento de tokens JWT. Além disso, o site '*https://jwt.io'* apresenta uma lista de bibliotecas prontas e adequadas para diferentes linguagens de programação, que podem auxiliar nessa tarefa.

Na prática, é possível extrair da seção "*payload"* do JWT o e-mail ou CPF para verificar o perfil do usuário que se autenticou na aplicação. Isso dará início à etapa de autorização dos acessos. Um exemplo da seção com os dados já decodificados pode ser visto na *Figura 3*.

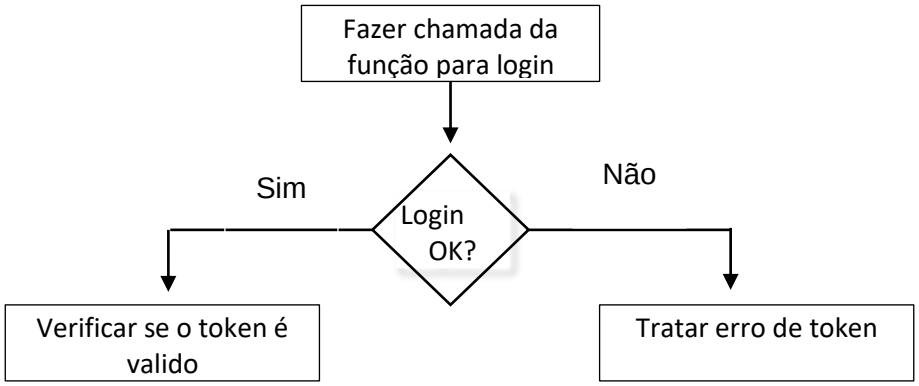
Figura 3 - Exemplo de seção payload no JWT de retorno do Login

Para os casos em que o login for validado, a indicação é redirecionar os usuários para o uso do login que já funcionava no sistema anteriormente, até que a situação deste usuário em particular, seja resolvida com o uso do login pelo Id Provider.

IMPORTANTE:

Sempre que a aplicação receber um token, é imprescindível que ele seja validado (utilizando as rotas 'verifying-login' ou 'verifying-login-deeply') antes de autorizar o usuário a realizar ações. o procedimento deve ser feito também todas as vezes em que o contexto da aplicação exigir autorização ou até mesmo de maneira periódica, para garantir que o token continua válido.

O fluxograma abaixo ilustra o processo de uso da rota login.



2.1.2 Utilizando a rota */ verifying login*

O propósito deste recurso é realizar a verificação da validade do *token*. Para isso, basta enviar a requisição incluindo o JWT recebido no parâmetro *token*. O *Quadro 3* complementa os detalhes dessa rota.

Quadro 3 - Verificação do Token

Item	**Descrição**
Method	**GET**
Base URL	https://esb.cps.sp.gov.br/sso-back/
Version	**v1**
Complete URL	https://esb.cps.sp.gov.br/sso-back/v1/verifying-token
Query params	**1** - **token**: JWT token recebido no momento do login

2.1.3 Utilizando a rota */ verifying – login- deeply*

De forma análoga ao recurso anterior, a proposta desta rota é verificar a validade do token de maneira mais abrangente, validando não apenas a existência e validade, mas também a parte da assinatura digital do *token.* O parâmetro a ser fornecido é o *token* JWT recebido no momento do login. O *Quadro 4* apresenta mais alguns detalhes do uso desta rota.

Quadro 4 - Verificação abrangente do Token

Item	**Descrição**
Method	**GET**
Base URL	https://esb.cps.sp.gov.br/sso-back/
Version	**v1**
Complete URL	https://esb.cps.sp.gov.br/sso-back/v1/verifying-token-deeply
Query params	1 - **token**: JWT *token* recebido no momento do login

Exemplo de requisição para verificação mais detalhada do *token*:

```
https://esb.cps.sp.gov.br/sso-back/v1/verifying-token-
deeply?token=JWTToken
```

2.1.4 Utilizando a rota */ logout*

Conforme o próprio nome indica, o objetivo desta rota é realizar o desligamento, ou *'logout',*" do usuário. Vale destacar que esta chamada efetuará o '*logout'* do usuário não apenas no contexto da aplicação, mas também no contexto do login vinculado ao *Id Provider* da Microsoft. Detalhes sobre o uso da rota estão destacados no *Quadro 5.*

Quadro 5 - Logout

Item	**Descrição**
Method	**DELETE**
Base URL	https://esb.cps.sp.gov.br/sso-back/
Version	**v1**
Complete URL	https://esb.cps.sp.gov.br/sso-back/v1/logout/<token>?callback_logout
Query params	1 - **<token>**: token JWT recebido no momento da autenticação 2 - **callback_logout**: endereço (url) na qual a aplicação tratará o retorno de logout. Por exemplo, redirecionar usuário.

Exemplo de requisição para *logout*:

```
https://esb.cps.sp.gov.br/sso-back/v1/logout?callback_logout=
```

2.1.5 Exemplos de código

Para exemplificar como a aplicação pode executar suas ações, são apresentados alguns códigos simples em diversas linguagens de programação. A intenção não é cobrir todos os procedimentos necessários, pois cada aplicação pode ter sua própria lógica de encaminhamento nas etapas pós-login. Na *Figura 4*, há um exemplo de página HTML para a etapa de envio da requisição de *login* ao Autenticador. O exemplo de código apresenta, na mesma página, dois recursos para realizar a chamada: uma âncora e um botão.

Figura 4 - Exemplo de página html para solicitar login com o Autenticador

```
1  <html>
2
3  <head>
4      <title>Teste de Login Autenticador CPS</title>
5  </head>
6
7  <body>
8
9      <p> Exemplo de chamada com âncora </p>
10     <a href="https://esb.cps.sp.gov.br/auth/login?url=https:/app.br/loginok&url2=loginerrado">
11         Login SSO
12     </a>.
13
14     <p> Exemplo de chamada com Botão </p>
15     <a href="https://esb.cps.sp.gov.br/auth/login?url=https:/app.br/loginok&url2=loginerrado">
16         <button>
17             Login SSO
18         </button>
19     </a>
20
21 </body>
22
23 </html>
```

2.1.6 Exemplo em PHP

Em outro exemplo, usando a linguagem PHP, apresentado na *Figura 5*, é mostrada uma forma de tratar o retorno do *token* recebido no endereço da aplicação que processa o *login* que foi validado.

Figura 5 - Exemplo de tratamento / decodificação do token em Linguagem PHP

```
@session_start();

if (isset($_GET['token'])) {
 function tokenDecode($token, $key){
  $decode = json_decode(base64_decode(str_replace('_', '/', str_replace('-',
'+', explode('.', $token)[1]))));
  return $decode->$key;
 }
 $_SESSION['CpfMicrosoft']   = tokenDecode($_GET['token'], 'cpf');
 $ _SESSION['EmailMicrosoft'] = tokenDecode($_GET['token'], 'email');

 // localiza o perfil da pessoa no sistema a partir do CPF e EMAIL
 //
 }
```

Maiores informações podem ser obtidas com a equipe do sistema SIG.

2.1.7 Exemplo em GENEXUS

A forma mais simples de operar o login no Genexus é obter o *token* da URL e, em seguida, processá-lo com o objeto *'JWTCreator'*. Uma operação *'FromJson'* divide os dados em credenciais para localizá-lo na base de perfis do sistema.

Figura 6 - Exemplo de tratamento / decodificação do token em Linguagem GENEXUS

```
Event Start
 &jwttoken   = &token.Replace("token=", "")
 &payload    = &JWTCreator.GetPayload(&jwttoken)
 &header     = &JWTCreator.GetHeader(&jwttoken)
 &jwtSDT.FromJson(&payload.Trim())

 // procura o usuário para determinar o perfil
 &jwtSDT.cpf
 &jwtSDT.email

EndEvent
```

Maiores informações podem ser obtidas com a equipe do sistema SIGA.

2.1.8 Exemplo em C#

Em C#, a operação deve obter o *token* enviado pela URL, decodificá-lo com *JwtSecurityTokenHandler* e, em seguida, extrair o CPF, nome, e-mail e tipo do usuário.

Figura 7 - Exemplo de tratamento / decodificação do token em Linguagem C#

```
public AutenticacaoCPS(string token)
{
 // URL do Autenticador
 var client = _clientFactory.CreateClient();
 client.BaseAddress = new Uri("https://esb.cps.sp.gov.br/sso-back/v1/");

 // verifica se o token é valido
 var parametros = $"verifying-token?token={token}";
 AutenticadorCPS dados = new AutenticadorCPS();

 var response = await client.GetAsync(parametros);
 if (response.IsSuccessStatusCode)
 {
  var stringResponse = await response.Content.ReadAsStringAsync();
  var handler = new JwtSecurityTokenHandler();
  var jwtSecurityToken = handler.ReadJwtToken(token);

  // obtem os dados da pessoa que realizou a autenticação
  dados.cpf = jwtSecurityToken.Claims.First(c => c.Type == "cpf").Value;
  dados.name = jwtSecurityToken.Claims.First(c => c.Type == "name").Value;
  dados.email = jwtSecurityToken.Claims.First(c => c.Type == "email").Value;
  dados.user_type = jwtSecurityToken.Claims.First(c => c.Type ==
"user_type").Value;

 }
 else {
   throw new HttpRequestException(response.ReasonPhrase);
 }
return View(dados);
}
```

Maiores informações com a equipe do sistema CPRJI.

2.2 AUTENTICADOR PARA APLICAÇÕES CPS

Outra necessidade dentro no contexto da autenticação ocorre aplicações e não usuários (pessoas) precisam estabelecer um processo de comunicação entre si. Em um ambiente heterogéneo em termos de tecnologias e soluções é importante que o CPS estabeleça protocolos de integração de dados e comunicação entre aplicações.

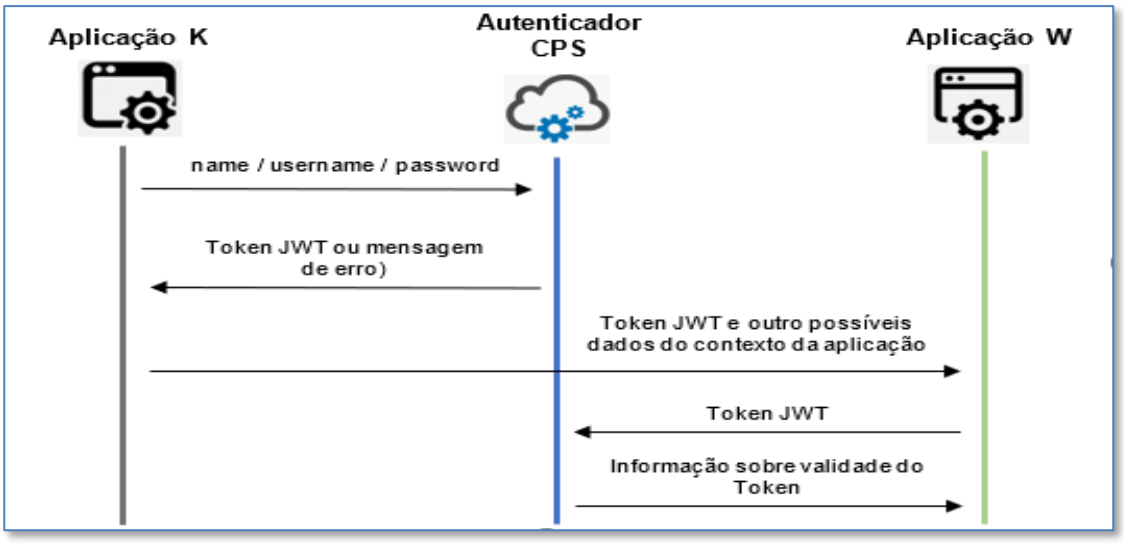
Para esse tipo de situação, foi criado um cadastro central de credenciais de aplicações, permitindo que elas autentiquem. Dessa forma, quando uma aplicação, chamada aqui hipoteticamente de **K**, precisa se comunicar com outra, ela realiza um processo de *login* enviando suas credenciais e recebendo um *token* JWT. De posse do token, a aplicação **K** deve encaminhá-lo à aplicação com a qual deseja se comunicar.

Considere que a aplicação **K** pretendo se comunicar com aplicação **W**. Ao receber a solicitação de comunicação, a aplicação **W** valida o *token* recebido encaminhando-o para o Autenticador, que verificará do conteúdo o *token* para identificar se foi emitido por ele e se as informações contidas estão corretas. O retorno do Autenticador à aplicação **W** pode ser mensagem de erro caso haja algum problema com o *token*.

Uma vez recebida a informação do *token*, a aplicação **W** poderá gerenciar a autorização da aplicação **K** no contexto de seu sistema aceitando-a se o token for válido ou recusar seu acesso (*token* inválido).

Na *Figura 8* apresenta uma representação do fluxo de comunicação entre o Autenticador CPS e as duas aplicações envolvidas na operação descrita.

Figura 8– Fluxo de comunicação entre Autenticador CPS e aplicações



Algumas definições utilizadas na descrição do processo da *Figura 8*:

- **Aplicação:** Sistemas computacionais no contexto do CPS tais como: Siga, SigURH, NSA
- **Autenticador CPS:** Serviço corporativo que oferece recursos para encaminhamento de autenticação de usuários pelo Identity Provider da Microsoft e um token próprio utilizado no ambiente interno do CPS para autenticação das aplicações.

2.2.1 Cadastro da aplicação no Autenticador

A primeira etapa para que a aplicação possa realizar as etapas de autenticação é fazer parte do cadastro de aplicações do Autenticador. Para realizar o cadastro, o responsável pela aplicação deve entrar em contato com a Divisão de informática, notificando-a de sua necessidade.

Vale destacar que se trata de um cadastro para o processo de autenticação. As autorizações de acesso ao uso de recursos de outras aplicações e serviços são controladas no contexto de cada sistema. Por exemplo, o simples cadastro no Autenticador não garante que a aplicação tenha acesso a todos os dados oferecidos pelo serviço de Tabelas Corporativas, pois cada integração tem suas próprias características. O processo de autorização de tais acessos é, portanto, tratado separadamente.

Voltando à questão do cadastro, uma vez que a Divisão de Informática tenha recebido a solicitação e realizado a inclusão da aplicação no Autenticador CPS, o responsável pela aplicação receberá as credenciais para realizar o *login*. A credencial é composta por um objeto contendo três itens no formato “chave-valor”, como no exemplo da *Figura 9.*

Figura 9 - Exemplo de formato dos dados de cadastro da aplicação no Autenticador CPS

```
{
  "name": "Appname",
  "username": "Appuser",
  "password": "qbp#8918@35lwk147"
}
```

O campo “name” representa o nome da aplicação ou sistema, funcionando como uma descrição.

O campo “username” representa o nome com que a aplicação será registrada ao acessar os serviços. Essa informação é única para cada sistema e não deve ser confundida como um nome de usuário típico de um desenvolvedor, pois não são permitidos vários usuários para uma mesma aplicação. As identificações de cada aplicação são únicas.

O campo “password” deve conter a senha de acesso.

A necessidade de se ter uma espécie de nome de usuário e senha para a aplicação é que, ao serem cadastradas, cada aplicação receberá um Application-ID para identificá-la. Com o uso complementar de um nome e senha, é possível implementar políticas de troca dessas senhas sem a necessidade de alterar o ID da Aplicação, já que ele somente será trafegado quando a autenticação for realizada e um *token* no formato JWT for retornado ao requisitante do login.

2.2.2 Utilizando a rota / login

Para realizar login e receber um *token* de acesso, deve-se enviar uma requisição para o endereço do Autenticador que processará esse pedido. O *Quadro 6* detalha os itens da rota de login para aplicações. Os dados enviados no formato *Json no Body* devem corresponder aos que foram cadastrados para aplicação, conforme descrito no item anterior deste documento.

Quadro 6 - Requisição de login de aplicações

Item	Descrição
Method	**POST**
Base URL	https://esb.cps.sp.gov.br/sso-back/
Endpoint	**app-v1/login**
Complete URL	https://esb.cps.sp.gov.br/sso-back/app-v1/login
Request Body	{ "name": "Appname", "username": "Appuser", “password”: “qbp#8918@35lwk147” }

Informações e a possibilidade de testes da requisição para este e outros endpoints podem ser obtidas no endereço eletrônico: https://datacorp.cps.sp.gov.br/endpoints/

2.2.3 Utilizando a rota /check

Essa rota deve ser utilizada pela aplicação para validar o *token* recebido. O *Quadro 7* apresenta as informações para utilização.

Quadro 7 - Verificação de token de aplicação

Item	Descrição
Method	**GET**
Base URL	https://esb.cps.sp.gov.br/sso-back/
Endpoint	**app-v1/check**
Complete URL	https://esb.cps.sp.gov.br/sso-back/app-v1/check
Header	`Authorization: Bearer eyJhbGciO1NiIsInR5...(JWT format)`

3 CONCEITOS E TECNOLOGIAS

Autenticação e Autorização são etapas diferentes no tratamento do usuário no contexto da aplicação. A **autenticação** ocorre quando o usuário prova sua identidade, geralmente quando sua identificação e senha são validadas.

A **autorização** se refere às ações que a aplicação permitirá que o usuário realize em seu contexto, uma vez que ele foi autenticado. Assim, a autorização representa o conjunto de permissões que o usuário tem ou não para usar os recursos do sistema.

3.1 AUTENTICAÇÃO LOCAL VERSUS PROVEDOR DE IDENTIDADE

A forma mais tradicional das aplicações identificarem seus usuários é realizando essa operação por meio de um banco de dados local com as contas de usuários. Em um ambiente empresarial, também é comum que o usuário precise acessar diferentes aplicações, e nesse caso, a autenticação local pode trazer alguns pontos negativos tais como:

- Usuários normalmente acham tedioso repetir inserção de seus dados de login;
- Nome de usuário e senha podem não ser os mesmos para todas as aplicações, o que aumenta a complexidade para o usuário;

- Vários bancos de dados com possibilidade de autenticar usuários de forma separada podem se tornar um risco de segurança para a organização, além de tornar a gestão dos recursos mais complexa e custosa

A solução padrão para este tipo de solução é delegar a tarefa de autenticação para um serviço dedicado exclusivamente a esse objetivo, chamado de **Provedor de Identidade *(Identity Provider = IdP)***.

Embora empresas do mercado web como *Google, Facebook* e *Twitter* ofereçam formas de IdP para seus usuários, no caso de uma organização específica, como o CPS, a solução ideal é o uso de um IdP interno. A centralização deve trazer benefícios no que diz respeito à facilidade de gestão de login das aplicações e um potencial ganho no tempo de desenvolvimento do login para novas aplicações.

O ambiente que melhora a experiência de usabilidade do usuário durante o processo de login é comumente chamado de Single Sign On.

3.2 Single Sign On - SSO

Um sistema SSO (*Single Sign On*) pode ser concebido como um método de autenticação que permite aos usuários se autenticarem em diferentes aplicações utilizando apenas uma credencial (informações de usuário e senha).

O SSO funciona a partir de um relacionamento de confiança entre um serviço provedor de login (neste caso, o Autenticador CPS) e um IdP. Em termos técnicos esse "relacionamento de confiança" se estabelece por meio de um "certificado digital" enviado do IdP para o Serviço de Autenticação. no formato de um *token* (mais informações sobre *tokens* podem ser obtidas no item *JWT - JSON Web Token* deste documento).

Um *token* SSO é um conjunto de dados passado de um sistema para outro durante o processo de *login*. Nesse processo, o *token* precisa ser digitalmente assinado para permitir sua verificação.

Para o Autenticador CPS, o processo de comunicação com o IdP foi construído utilizando o protocolo OpenID Connect, que, no ambiente do *Microsoft Azure*, serviços de nuvem do CPS, recebe o nome de *Microsoft OICD - OpenID Connect*. A seguir, são descritas as principais características desse protocolo.

3.3 OpenID Connect

OpenID Connect não é o primeiro protocolo desenvolvido para oferecer recursos de IdP. Antes dele, já existiam, por exemplo, o *SAML, OpenID 1.0* e *OpenID 2.0*. No entanto, o *OpenId Connect* é recomendado por sua simplicidade e usabilidade, alcançadas a partir do aprendizado com seus antecessores.

Essas características são atribuídas a ele devido as seguintes características:

Simplicidade: seu funcionamento é simples o suficiente para permitir a integração com aplicações que requerem o básico em termos de autorização, mas também possui características e opções de segurança que atendem a exigências maiores.

Consumo baseado em *token* de identidade: os clientes do IdP baseado no *OpenID Connect* recebem a identificação do usuário codificada no padrão *JSON Web Token (JWT),* chamado de *ID Token*. O JWT oferece portabilidade para um número considerável de assinaturas e algoritmos de criptografia.

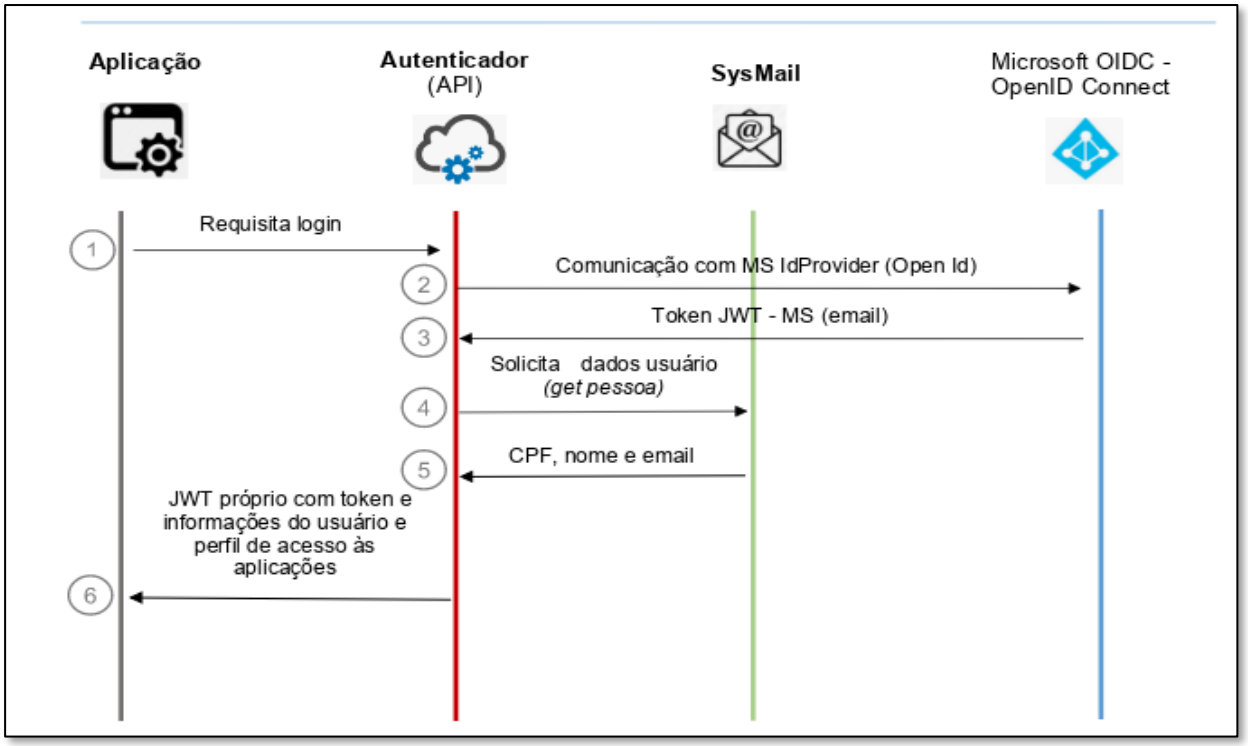
Baseado no protocolo *OAuth2.0*: o *ID Token* é gerado com base no formato padrão do fluxo do protocolo *OAuth2*, oferecendo suporte tanto para aplicações web quanto para aplicativos móveis. O acesso à autenticação e autorização (obter token de acesso) é possível usando um único protocolo.

3.4 FLUXO DE LOGIN PARA AUTENTICADOR CPS

Esta seção descreverá o fluxo de login sob dois aspectos: um fluxo resumido e um fluxo detalhado. O fluxo resumido descreve o processo sem os detalhes específicos da comunicação com o *Microsoft OpenID*, enquanto o fluxo detalhado contém um número maior de especificações.

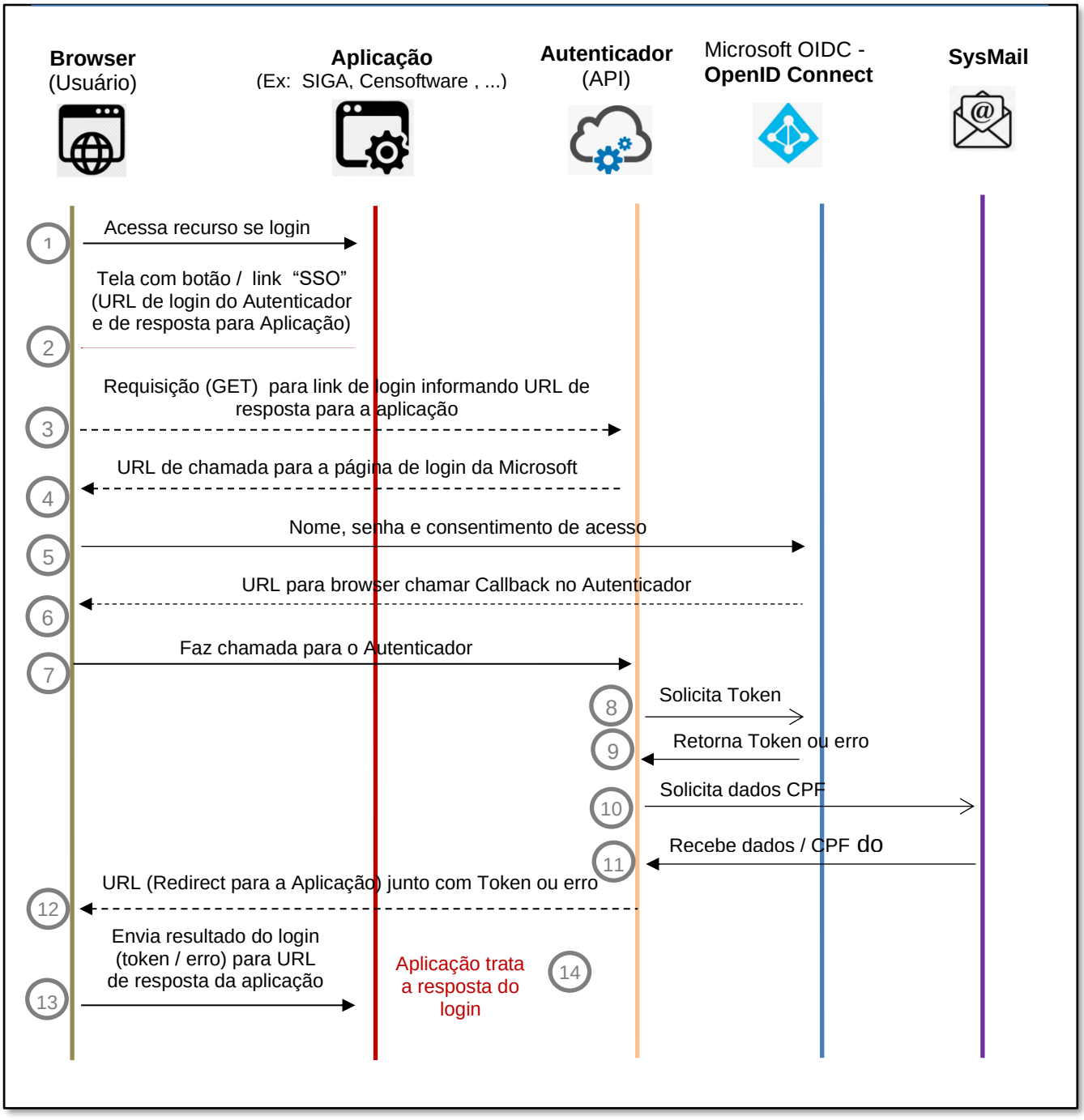
As etapas do processo de login resumido são ilustradas na *Figura 10*. Embora o processo pareça longo, a aplicação precisa se concentrar apenas das etapas 1 e 6.

Figura 10 - Fluxo resumido do uso do Autenticador CPS / Microsoft



A *Figura 11* apresenta o fluxo, incluindo o usuário como início do processo e fornece mais detalhes sobre a etapa do *OpenID Connect*. Embora o fluxo pareça longo, a aplicação cuidará apenas dos passos 2 e 14.

Figura 11 - Fluxo detalhado do uso do Autenticador CPS / Microsoft



O processo de *login* consiste em quatorze etapas especificadas da seguinte forma:

1 - O usuário indica por meio do navegador ou outra interface da aplicação, seu desejo de realizar login, representado, por exemplo, pela abertura da página inicial ou clique em um botão de solicitação de *login*.

2 - A aplicação responde ao usuário oferencendo o recurso de *login*, que pode ser umlink, botão ou qualquer outro elemento da interface que, ao ser executado, envie requisição para a API do Autenticador CPS.

3 - A partir da ação do usuário a aplicação realiza uma chamada para a rota do Autenticador responsável pela solicitação de login usando o método GET, informando dois parâmetros:

***'URL'*:** indica o endereço no qual a aplicação tratará o retorno do login, caso ele seja autorizado pelo IdP;

***'URL2'*:** representa o endereço da aplicação que cuidará do retorno caso o processo de autorização seja inválido, como quando o usuário fornecedados de login incorretos.

Considerando que a aplicação SIGA definiu sua rota para tratar o login válido em https://siga.cps.sp.gov.br/autenticou e o endereço para retorno de *login* inválido em https://siga.cps.sp.gov.br/erro-no-login e que o endereço do Autenticador responde em https://esb.cps.sp.gov.br/sso-back/v1/login, a chamada completa (incluindo endereço do Autenticador e os parâmetros da aplicação para tratar os retornos) ficaria assim:

a. Rota do Autenticador que recebe as solicitações de login.
b. Ponto de interrogração "?" indicando o início da passagem do primeiro parâmetro
c. Nome do parâmetro ('*callback*') e sinal de igual '=' indicando o conteúdo que o parâmetro vai assumir.
d. Valor (conteúdo) para o primeiro parâmetro.
e. Indicador '&' de início do segundo parâmetro.
f. Nome do segundo parâmentro ('*callback_erro'r*) e sinal de igual '=' indicando o conteúdo que o parâmetro vai assumir
g. Valor (conteúdo) para o segundo parâmetro.

4 - O Autenticador devolve ao navegador uma indicação de chamada para a URL da página de login do IdP, que, neste caso, é uma página disponibilizada pelo serviço do *Microsoft Azure*.

5 - O usuário informa sua identificação (email e senha) e deve ainda consentir o uso dos dados no ambiente de login, permitindo que o IdP valide os dados de login;

6 - O IdP devolve à aplicação o endereço no qual o Autenticador deve tratar o retorno da tentativa de login. Esse endereço é conhecido pelo IdP porque foi cadastrado quando o Autenticador foi registrado como uma aplicação que pode se comunicar com o IdP;

7 - O navegador faz a chamada para a rota do Autenticador que trata o retorno da inserção dos dados pelos usuários;

8 - Com o retorno oferecido pelo IdP, o Autenticador lhe encaminha uma solicitação de *token*;

9 - O IdP retonar a solicitação do *token* com duas possibilidades. Se os dados informados pelo usuário são válidos, o retorno será um token digitalmente assinado. Caso contrário, o IdP retorna à indicação de login inválido.

10 - O Autenticador seleciona no *token* da *Microsoft* a informação referente ao e-mail do usuário e, com esses dados, solicita ao sistema Sysmail que lhe retorne qual é o CPF do usuário.

11 - O *Sysmail* retorna os dados solicitados (CPF do usuário) ou uma mensagem de erro.

12 - Ao receber os dados do *Sysmail,* o Autenticador gera um *token* JWT próprio do CPS com os dados do usuário e o encaminha para que o navegador o devolva para o endereço (URL) indicado pela aplicação no início do fluxo de *login*. Dessa forma, o *token* recebido pela aplicação na etapa 14 já terá o CPF contido nele. Isso é importante porque o CPF tem sido usado como identificador de usuários no contexto das aplicações. Assim, mesmo que o usuário tenha diferentes e-mails para diferentes domínios, como @cps, @fatec e @etec, ele terá sempre o mesmo CPF para

o identificar, e a aplicação pode encontrar o usuário independentemente do e-mail usado para se identificar.

13 - O navegador faz a chamada para o endereço da aplicação que trata o *login* ou o endereço que recebe o erro. As rotas da aplicação que recebem o retorno são as informadas na etapa 3, mais especificamente nos parâmetros “d” e “g”.

14 - A aplicação recebe os dados do resultado do *login* e os trata conforme seus critérios de autorização para o usuário ou fornece as respectivas informações ao usuário de que o login não foi validado.

A rota do Autenticador CPS que cuida das solicitações de login tem o seguinte endereço: https://esb.cp.sp.gov.br/sso-back/v1/login

3.5 ESTRUTURA *JWT*

O conceito de *token* está quase sempre ligado à identificação. A palavra pode estar associada tanto a processos de identificação quanto a alguns tipos de criptoativos (criptomoedas). Em termos de identificação, pode se referir a um dispositivo físico, como os utilizados por algumas instituições financeiras para gerar um número de identificação, ou, no caso do JWT, a um conjunto de símbolos que representam uma assinatura digital ou uma chave.

Um *token* JWT é o resultado de um conjunto de especificações que fazem parte de uma estrutura maior, o JOSE (JSON *Objetct Signing and Encryption*). No do mecanismo de autenticação aqui tratado, o Token JWT é, na prática, uma *string*, uma sequência de caracteres que representa um objeto.

O *token* JWT é composto por três partes: *Header, Payload* e *Signature.*

Header: localizado na parte inicial do token é um objeto do tipo JSON composto por dois elementos: ‘alg’ e ‘typ’. O primeiro indica qual o algoritmo de criptografia utilizado (por exemplo, HMAC, SH256, HS384 ou RSA), e o segundo o tipo de token (JWT, por exemplo).

Payload: segundo elemento do *token* JWT, é nesta parte que estão contidas as *claims*. Uma *claim* pode ser compreendida com um atributo ou propriedade. No JWT, as

claims podem ser classificadas em três tipos: *registered, private* e *public* (registrada, privada e púbica). *Claims* do tipo *"registered"* são criadas de forma pré-determinada pelo JWT. Exemplos desse tipo incluem:

- 'jti': é a identidade (ID) única do *token*;
- 'exp': abreviação de *expiration*, indica o momento de expiração do *token*, a partir do qual ele não é mais válido. *Claims* dos tipos privadas e públicas são criadas pela organização usuária e normalmente indicam informações sobre a própria organização e os usuários autenticados.

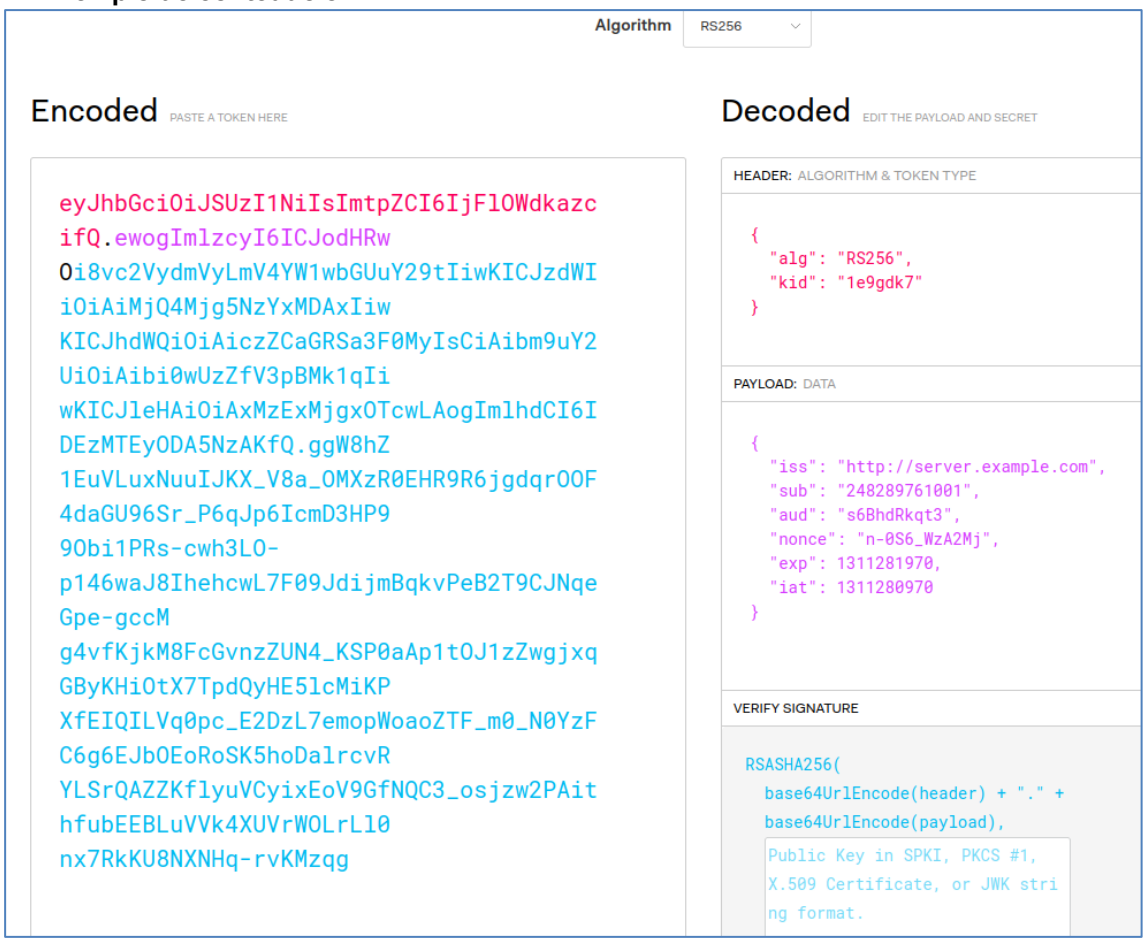
***Signature*:** resultado de uma codificação (*encode*) do *Header* e do *Payload,* acrescentando-se uma palavra-chave. Este campo, gerado a partir do tipo de referência para o *token* enviado no *Header* (no exemplo, do tipo JWT), permite verificar se a cadeia de *strings* que forma o *token* não foi alterada. A palavra-chave que controla este processo fica armazenada do lado do servidor IdP.

A *Figura 12* apresenta um conteúdo no formato *JWT* (lado esquerdo = *Encoded*) submetido ao processo de decodificação (lado direito = *Decoded*). No lado direito, é possível identificar as 3 partes do conteúdo do *token*: *Header, Payload* e *Signature*.

A imagem foi gerada a partir do modelo de arquivo disponível em: https://connect2id.com/learn/openid-connect que foi submetido à própria interface *web* de testes do JWT disponível em: https://jwt.io/ .

Destaca-se que a página web do JWT (https://jwt.io/) é uma excelente ferramenta de testes para verificar e aprender sobre o JWT.

Figura 12 - Exemplo de conteúdo JWT



Na prática, percebe-se que a grande utilidade do JWT não reside em ocultar informações, pois elas podem ser decodificadas por meio do padrão *Base64 decode*. A real vantagem está na confiabilidade dos mecanismos de assinatura digital. Como o token é autocontido, ou seja, carrega em si mesmo o conteúdo e a assinatura, o servidor IdP pode validar sua legitimidade. Vale destacar que o *token* não inclui o campo de senha do usuário.

REFERÊNCIAS

Auth0, 2022. **JWT**. Disponível em: https://jwt.io/

Connect2id, 2022. **OpenID Connect explained**. Disponível em: https://connect2id.com/learn/openid-connect

JONES et al. 2012. JSON Web Token (JWT) - draft-jones-json-web-token. Disponível em: https://openid.net/specs/draft-jones-json-web-token-07.html

NOLETO, C. 2022 **JWT (JSON Web Tokens)**: o que é e como usar na prática?. Disponível em: https://blog.betrybe.com/tecnologia/jwt-json-web-tokens/

ONELOGIN, 2022 **How does Single Sign-On Work?** Disponível em: https://blog.betrybe.com/tecnologia/jwt-json-web-tokens/

Plataforma de Identidade da Microsoft e protocolo OpenId Connect https://docs.microsoft.com/pt-br/azure/active-directory/develop/v2-protocols-oidc

Administração Central
Rua dos Andradas, 140
Santa Ifigênia – 01208-000
São Paulo – SP
Tel.: +55 11 3324-3300

Centro Paula Souza

SÃO PAULO
GOVERNO DO ESTADO