

# Comunicação Portal do titular com o portal do cliente

O funcionamento do Portal do Titular depende da sua comunicação com o Portal do Cliente. Neste documento será apresentada como fazer essa comunicação.

- [Desenho da Solução](#)
- [Acesso ao Portal do titular na forma não autenticada](#)
- [Acesso ao Portal do titular na forma autenticada](#)
- [Autorização MD2 Portal do Titular - comunicação com portal do cliente](#)

# Desenho da Solução

**1. Efetuar Login** = Corresponde ao login do titular no Portal do Cliente

**1.1 Autenticar Titular** = Controle de acesso do Portal do Cliente reconhece o usuário na base.

**2. Acessar Menu Privacidade** = Após entrar no Portal do Cliente, o titular vai acessar o Portal do Titular

**2.1 Direcionar Portal do Titular passando tokenClient** = O Portal do Cliente invoca o Portal do Titular passando o token do cliente

**2.1.1 Redirecionamento com tokenClient e apresentação do menu** = Ao ser invocado, o Portal do Titular chama o Proxy (Intermediário entre o Portal do Titular e o Quality Manager) passando o token do cliente.

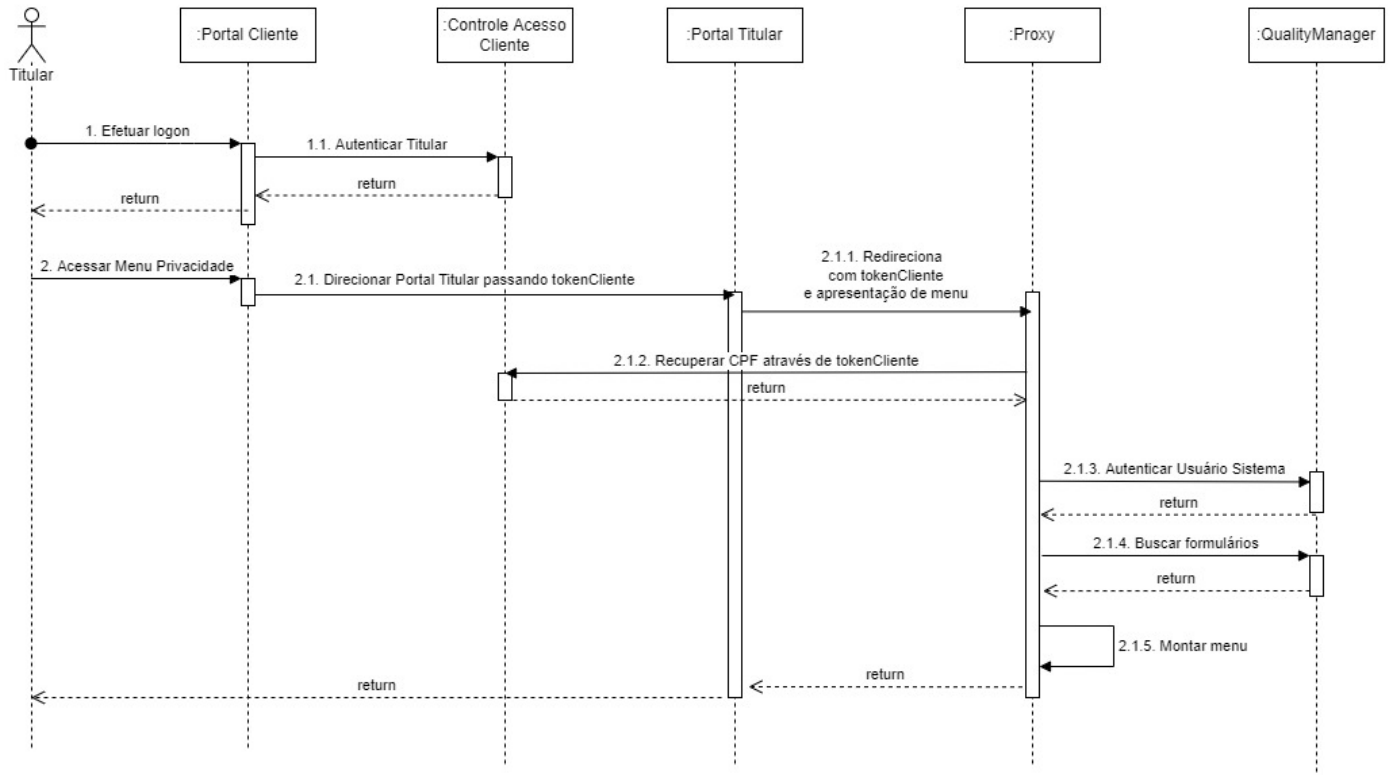
**2.1.2 Recuperar o CPF através do tokenClient** = Ao receber o token do cliente, o proxy recupera e retorna o CPF do mesmo.

**2.1.3 Autenticar Usuário Sistema** = Portal do Titular autentica no Quality Manager com o usuário de sistema.

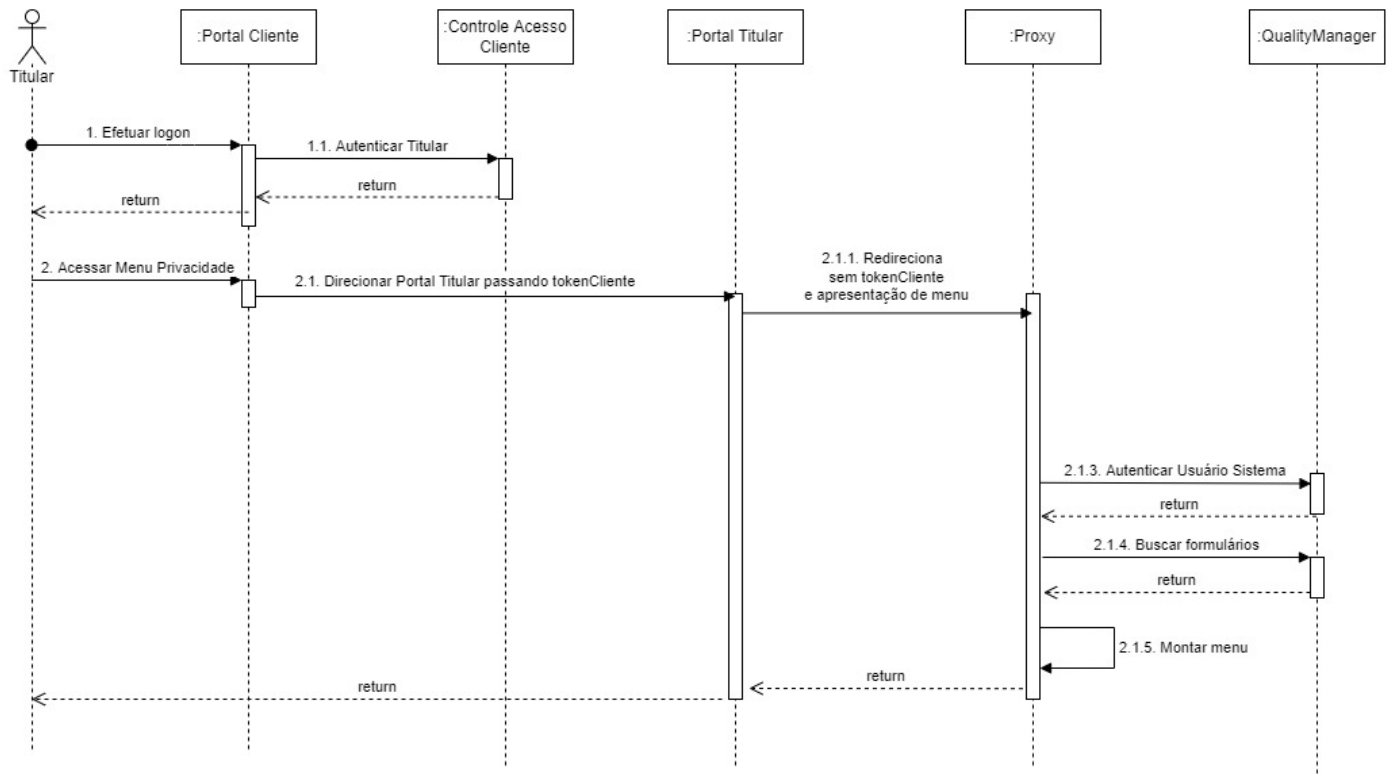
**2.1.4 Buscar Formulários** = Quality Manager retorna os formulários associados ao usuário de sistema logado.

**2.1.5 Mostra menu** = Menu é montado de acordo com os formulários retornados.

## Cenário usuário autenticado



## Cenário usuário não autenticado





# Acesso ao Portal do titular na forma não autenticada

## **Acesso ao Portal do titular na forma não autenticada**

1. Titular efetua login com sucesso no Portal do Cliente.
2. Ao acessar o menu de privacidade, o portal do cliente direciona o usuário ao Portal do Titular.
3. O Portal do Titular, por sua vez, autentica no Quality Manager com o usuário de sistema.
5. Logado no Quality Manager, o usuário de sistema faz a busca dos formulários associados e monta o menu de privacidade, apresentando-o ao titular.

# Acesso ao Portal do titular na forma autenticada

## **Acesso ao Portal do titular na forma autenticada**

1. Titular efetua login com sucesso no Portal do Cliente.
2. Ao acessar o menu de privacidade, o portal do cliente direciona o usuário ao Portal do Titular passando o token do cliente (tokenClient).
3. O Portal do Titular, por sua vez, recupera o CPF através do tokenClient
4. Ao recuperar o CPF, o Portal do Titular autentica no Quality Manager com o usuário de sistema.
5. Logado no Quality Manager, o usuário de sistema faz a busca dos formulários associados e monta o menu de privacidade, apresentando-o ao titular.

# Autorização MD2 Portal do Titular - comunicação com portal do cliente

## Objetivo

O processo de autenticação por token do cliente é método mais seguro para a passagem do CPF/CNPJ ao MD2 Portal do Titular. Foi desenvolvido seguindo os padrões de mercado e boas práticas de consumo de API's.

A documentação a seguir apresenta detalhes técnicos que devem ser seguidos para funcionamento correto da solução.

## Funcionamento

A configuração do MD2 Portal do Titular é efetuada em dois arquivos, **configPortal.js** e **configPortal.properties**. Através do arquivo **configPortal.js**, a aplicação recebe o token gerado pelo cliente e repassa durante a transferência do usuário do portal do cliente para o MD2 Portal do Titular.

Após receber o token, o MD2 Portal do Titular utiliza a configuração no arquivo **configPortal.properties**, previamente feita, para acessar a API informada nos parâmetros, e consulta informando o token e esperando como retorno o número do CPF/CNPJ do titular.

Somente após este processamento, o MD2 Portal do Titular será liberado para a uso.

Os padrões descritos abaixo de envio do token repassado pelo cliente para a API de validação exposta e do retorno do CPF/CNPJ são os aceitos pela aplicação. Outros formatos não são suportados.

# Configuração

## Arquivo configPortal.js

O arquivo configPortal.js armazena funções para que o MD2 Portal do Titular receba o token, enviado pelo portal do cliente, e o encaminhe para a o backend, que faz a validação através da API configurada no arquivo [configPortal.properties](#).

Exemplos de configurações:

### 1 - Token enviado por URL

```
function isLogged() {
    var tokenClient = getParams(window.location.href).tokenClient;
    if ("undefined" != tokenClient && null != tokenClient && "" != tokenClient) {
        return true;
    } else {
        return false;
    }
}

function findTokenClient() {
    // esta função é para quando o cpf/cnpj deve ser adquirido pelo portal
    // através do token encaminhado pelo portal do cliente.
    var tokenClient = getParams(window.location.href).tokenClient;
    if ("undefined" != tokenClient && null != tokenClient && "" != tokenClient) {
        return tokenClient;
    } else {
        return '';
    }
}

function findDocument() {
    // esta função é para quando o cpf/cnpj é encaminhado ao portal diretamente pelo portal
    do cliente.
    return '';
}

var getParams = function (url) {
    var params = {};
```



```

var parser = document.createElement("a");
parser.href = url;
var query = parser.search.substring(1);
var vars = query.split("&");
for (var i = 0; i < vars.length; i++) {
    var pair = vars[i].split("=");
    params[pair[0]] = decodeURIComponent(pair[1]);
}
return params;
};

```

## 2 - Token enviado como parâmetro

```

function isLogged() {
    var documentoClient
    if (window.opener == null) {
        documentoClient = window.document.getElementById('tokenClient');
    } else {
        documentoClient = window.opener.document.getElementById('tokenClient');
    }

    if (null == documentoClient) {
        return false;
    } else {
        return true;
    }
}

function findTokenClient() {
    // esta função é para quando o cpf/cnpj deve ser adquirido
    // pelo portal através do token encaminhado pelo portal do cliente.
    var documentoClient
    if (window.opener == null) {
        documentoClient = window.document.getElementById('tokenClient');
    } else {
        documentoClient = window.opener.document.getElementById('tokenClient');
    }

    if (null == documentoClient) {
        return '';
    } else {

```

```

        return documentoClient.value;
    }
}

function findDocument() {
    // esta função é para quando o cpf/cnpj é encaminhado ao portal diretamente pelo portal
    do cliente.
    return '';
}

```

A configuração do arquivo configPortal.js é de responsabilidade do cliente, porque pode variar conforme o parâmetro será repassado do seu portal do cliente para o portal do titular, portanto os exemplos acima são somente dois casos dos vários que podem existir.

## Arquivo configPortal.properties

O arquivo configPortal.properties é armaneza informações para validação do token recebido pelo portal do cliente, durante a navegação do usuário no MD2 Portal do Titular.

A validação disparada pelo MD2 Portal do Titular é feita através do consumo de uma API e que pode ser feita através dos métodos GET e POST.

### Método GET

```

23 config.portal.secure.authenticationHost=http://localhost:8080/portal-cliente/rest/token/valida/ 1
24 config.portal.secure.authenticationParameterEnv=tokenClient 2
25 config.portal.secure.authenticationParameterRet=cpf 3
26 config.portal.secure.authenticationMethod=GET 4

```

1 – Informe a URL da API para a validação do token

2 – Para que o token seja enviado por parâmetro **diferente** do Authorization Bearer, informe em "authenticationParameterEnv" o nome do parâmetro para que a requisição seja customizada.

Por padrão a aplicação sempre envia o token para a API do cliente no padrão Authorization Bearer

Exemplo de requisição padrão:

```
curl --location --request GET 'http://hostCliente: 8080/servico/api/validaToken' \
```

```
--header 'Authorization: Bearer  
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE2MjY4OTUyOTUsInVzZXJfbmFtZSI6InFtX3NlcnZpY2UiLC.  
andWXA4qW_9iIArs' \  
--data-raw ''
```

Exemplo de requisição customizada:

```
curl --location --request GET 'http://hostCliente: 8080/servico/api/validaToken' \  
--header 'Authorization: Bearer  
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE2MjY4OTUyOTUsInVzZXJfbmFtZSI6InFtX3NlcnZpY2UiLC.  
andWXA4qW_9iIArs' \  
--header 'tokenClient:  
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE2MjY4OTUyOTUsInVzZXJfbmFtZSI6InFtX3NlcnZpY2UiLC.  
andWXA4qW_9iIArs' \  
--data-raw ''
```

3 – Em caso de retorno do CPF/CNPJ por uma variável, informar nesse parâmetro o nome do atributo de retorno. Tipo de retorno aceitos:

1 - Número do cpf/cnpj direto sem formatação

```
11111111111
```

2 - JSON

```
{ "cpf" : "11111111111" }
```

O nome do atributo **cpf** foi apenas com caráter ilustrativo, o sistema irá buscar o nome do atributo com base no que for informado no **parâmetro 3**

4 – Nesse parâmetro informe o verbo a ser utilizado para o consumo do serviço: (GET)

## Método POST

**Até a versão 2.58**, o MD2 Portal do Titular é compatível com o envio do token do cliente para validação na API em 3 formas:

- Através da URL
- Através do parâmetro cadastrado no arquivo de configuração (item 2 da imagem abaixo)

- Através do body da requisição usando o nome definido no arquivo de configuração (item 2 da imagem abaixo)

Na versão 2.58, e posteriores, é adotado a forma descrita nesta documentação.

```
23 config.portal.secure.authenticationHost=http://localhost:8080/portal-cliente/rest/token/valida/ 1
24 config.portal.secure.authenticationParameterEnv=2
25 config.portal.secure.authenticationParameterRet=cpf 3
26 config.portal.secure.authenticationMethod=POST 4
```

1 - Informe a URL da API para a validação do token

2 - Para consumo de API pelo verbo POST, o token é enviado no parâmetro Authorization Bearer. Não informe nenhum valor em "authenticationParameterEnv". Exemplo de requisição:

```
curl --location --request POST 'http://hostCliente: 8080/servico/api/validaToken' \
--header 'Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE2MjY4OTUyOTUsInVzZXJfbmFtZSI6InFtX3NlcnZpY2UiLCJandWXA4qW_9iIARs' \
--data-raw ''
```

3 - Em caso de retorno do CPF/CNPJ por uma variável, informar nesse parâmetro o nome do atributo de retorno. Tipo de retorno aceitos:

1 - Número do cpf/cnpj direto sem formatação

```
11111111111
```

2 - JSON

```
{ "cpf" : "11111111111" }
```

O nome do atributo **cpf** foi apenas com caráter ilustrativo, o sistema irá buscar o nome do atributo com base no que for informado no **parâmetro 3**

4 - Nesse parâmetro informe o verbo a ser utilizado para o consumo do serviço: (POST)

## Referências

- <https://jwt.io/introduction>
- <https://www.rfc-editor.org/rfc/rfc6749#page-18>
- <https://www.oauth.com/oauth2-servers/making-authenticated-requests/>