



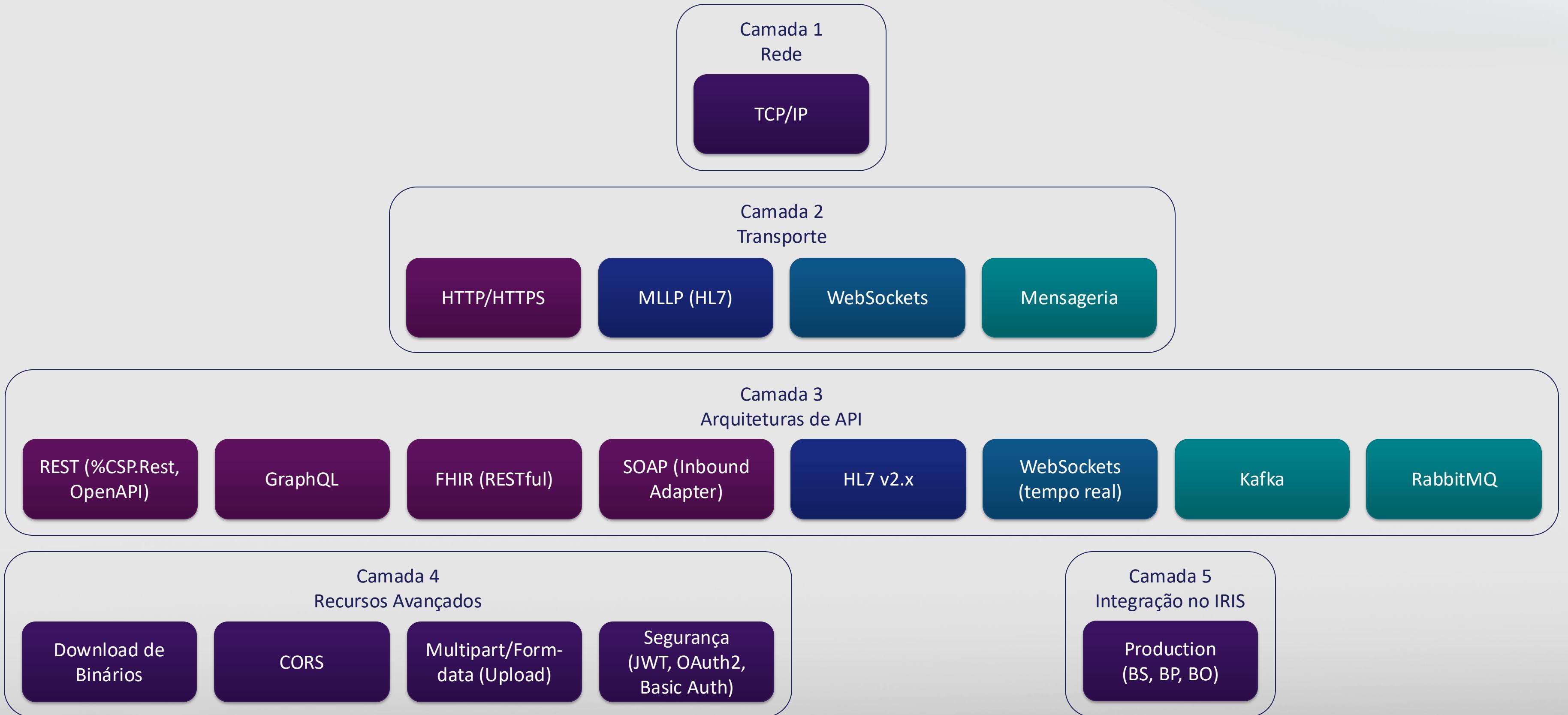
APIs no InterSystems IRIS

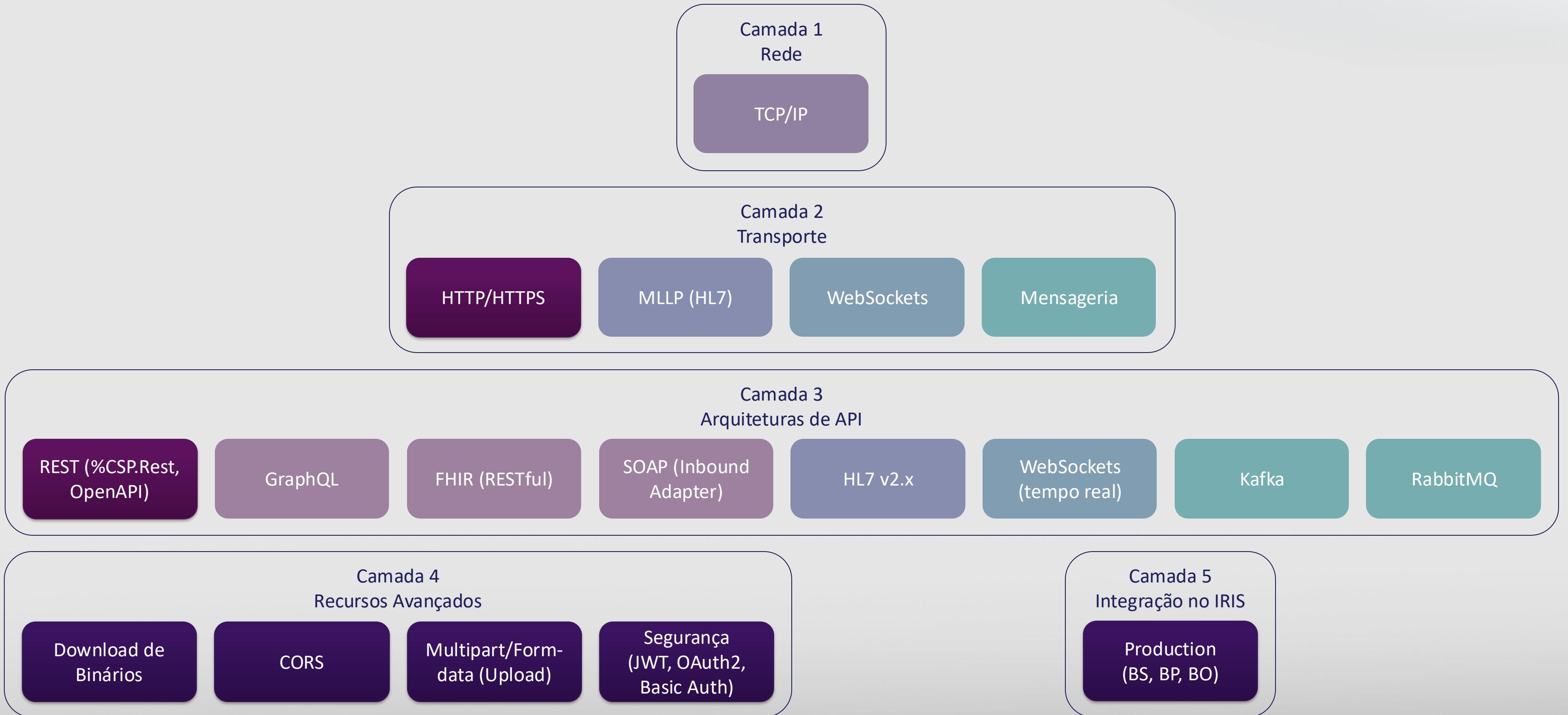


O que veremos?

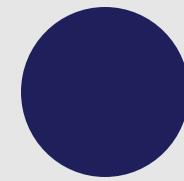
Assuntos que serão abordados durante o curso



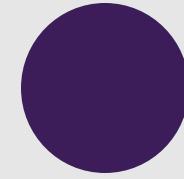




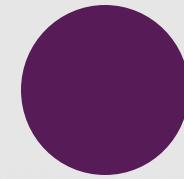
Índice



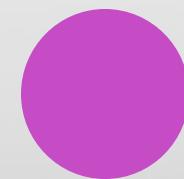
Introdução



Portal de Administração



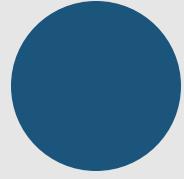
Construindo APIs I



Segurança



Construindo APIs II



APIs e Production



IAM



Introdução

Preparando o ambiente e entendendo APIs



Preparando o ambiente



Introdução a APIs

O que é API?

- Significa “**Interface de Programação de Aplicações**” (ou Application Programming Interface);
- É um mecanismo que permite que dois componentes de software se comuniquem entre si;
- Conecta soluções e serviços sem a necessidade de saber como foram implementados;
- Os tipos mais comuns de arquitetura de API são:
 - **SOAP**;
 - **REST**;
 - **GraphQL**.
- É possível criar todos os tipos de APIs no IRIS.

Exemplos de APIs no IRIS





Portal de Administração

Conhecendo o Portal do IRIS e sua integração com APIs



Portal de Administração do IRIS

O que é o Portal?

- É uma aplicação web (página) padrão do IRIS;
- Permite executar tarefas de administração e gerenciamento de sistemas;
- Torna possível administrar o sistema remotamente, sem o Caché instalado na máquina em uso.



Portal de Administração do IRIS

Como ele se relaciona a APIs?

- É onde fazemos a configuração das aplicações web;
- As configurações de aplicações web incluem:
 - Indicar para qual classe o endpoint será direcionado;
 - Gerenciar o acesso a este endpoint;
 - Gerenciar recursos de segurança.

Editar Aplicação Web

Salvar

Cancelar

Edita a definição de aplicativo web /api/monitor:

General Application Roles Matching Roles Cross-Origin Settings Percent Class Access

Nome /api/monitor
Obrigatório. (por exemplo /csp/appname)

Descrição Monitoring REST Apis

NameSpace %SYS Aplicativo padrão para %SYS: /csp/sys Aplicação Padrão do Namespace

Ativar aplicativo

Habilitar REST
Expedir Classe %Api.Monitor
Obrigatório.

Redirecionar caminho vazio
Enable Traces
Usar autenticação JWT
 WSGI [Experimental].
 CSP/ZEN
 Análise Serviços de Web de Entrada Impedir ataque CSRF de login Enable Traces

Configurações de segurança

Recurso obrigatório Agrupar por ID

Métodos permitidos para autenticação Não Autenticado Senha Kerberos OAuth2

Configurações da sessão

Timeout de sessão 3600 Segundos Classe de eventos .cls

Usar cookie para o controle de sessão sempre Caminho do cookie de sessão /api/monitor/ Escopo do cookie da sessão Estrito Escopo do cookie do usuário Estrito

Editar Aplicação Web, é a página onde são feitas, por exemplo, as configurações do endpoint “/api/monitor”.

bridgergroup.com.br



Portal de Administração do IRIS

O que são Application Roles?

- Relacionam quais funções de usuário aquela aplicação terá acesso;
- Neste curso, vamos sempre configurar a Application Role **%All** para todas APIs, por simplicidade;
- A função **%All** é muito poderosa, e não deve ser utilizada em produção.



Aba Application Roles

É onde são configuradas as funções de usuário de uma aplicação web.

Uma aplicação pode conter uma, nenhuma ou várias funções de usuário simultaneamente.

The screenshot shows the 'Application Roles' tab selected in a top navigation bar. Below the tab, a message states: 'Quando um usuário entra neste aplicativo, as seguintes funções serão automaticamente adicionadas ao seguinte conjunto de função:'. A section titled 'Funções da Aplicação' contains a list with '%DB_IRISSYS' and a 'Remover' button. Below this, instructions say: 'Para adicionar uma função de aplicação, selecione uma ou mais funções disponíveis e pressione 'Associar''. Two lists are shown: 'Disponível' (Available) and 'Selecionado' (Selected). The 'Disponível' list contains numerous function names, many starting with '%'. The 'Selecionado' list is currently empty. Between the two lists are four arrows: up, down, left, and right, used for moving items between them. A large 'Associar' (Associate) button is located at the bottom right of the selection area.

Exemplo de aba de Application Roles.

Visualizando o Portal





Construindo APIs I

Construindo sua primeira API REST no IRIS



API REST

O que é REST?

- Significa “**Transferência de Estado Representacional**” (ou “Representational State Transfer”);
- É um conjunto de restrições de arquitetura;
- Uma API REST é uma API que segue os princípios de design do estilo arquitetônico REST;
- APIs REST se comunicam por meio de solicitações HTTP;
- Uma solicitação deve conter:
 - Identificador exclusivo de recurso (normalmente via URL);
 - **Método HTTP**;
 - **Cabeçalho HTTP**.

API REST

Métodos HTTP

- Informa ao servidor o que ele precisa fazer com o recurso identificado.
- Os métodos HTTP mais comuns são:
 - **GET**, solicita recursos do servidor;
 - **POST**, cria novos recursos no servidor;
 - **PUT**, atualiza recursos existentes no servidor ou cria-os em alguns casos;
 - **PATCH**, atualiza parcialmente um recurso existente;
 - **DELETE**, remove recursos do servidor.

API REST

Classe %CSP.Rest no IRIS

- No framework do IRIS, a classe %CSP.Rest deve ser usada como classe pai para as aplicações web REST;
- As principais características desta classe são:
 - **UrlMap**, define as rotas;
 - **%request**, variável contendo dados da requisição;
 - **%response**, variável contendo dados da resposta;
 - **Write**, comando usado para "escrever" no output da API.

UrlMap

Cada tag **<Route>** corresponde a uma rota na classe.

O parâmetro **Url** define o endpoint que será usado pela rota.

O parâmetro **Method** define o método HTTP que pode ser utilizado na rota.

O parâmetro **Call** define o método que será chamado ao receber requisição na rota.

```
4  XData UrlMap
5  {
6  <Routes>
7      <Route Url="/exemplo-get/:parametroPath" Method="GET" Call="ExemploGet"/>
8      <Route Url="/exemplo-post/" Method="POST" Call="ExemploPost"/>
9  </Routes>
10 }
```

Exemplo de UrlMap.

API REST

Propriedades e métodos da %CSP.Request e %CSP.Response

- %request
 - **Data**, contém os parâmetros da requisição;
 - **GetCgiEnv**, contém variáveis CGI, como os cabeçalhos da requisição;
 - **Content**, contém o corpo da requisição.
- %response
 - **Status**, contém o status HTTP da resposta;
 - **CharSet**, contém o Character Set da resposta;
 - **ContentType**, contém o Content Type da resposta.

Método na %CSP.REST

```
Debug | Copy Invocation
12 ClassMethod Exemplo() As %Library.Status
13 {
14     // A variável %request é um objeto da classe %CSP.Request
15     // Aqui vemos a busca de algumas propriedades.
16     Set userId = %request.Data("userId",1)
17     Set exemploHeader = %request.GetCgiEnv("HTTP_EXEMPLOHEADER")
18     Set body = %request.Content.Read()
19
20     // A variável %response é um objeto da classe %CSP.Response
21     // Aqui vemos a atribuição de algumas propriedades.
22     Set %response.Status = ...#HTTP200OK
23     Set %response.CharSet = "UTF-8"
24     Set %response.ContentType = "application/json"
25
26     Write {"message": "success"}
27
28     Return $$OK
29 }
30 }
```

Implementação de método de exemplo numa classe %CSP.Rest.

A variável **%request** é um objeto da classe %CSP.Request.

A variável **%response** é um objeto da classe %CSP.Response.

Ambas variáveis estão no escopo da classe.

Exemplo de método na %CSP.REST.



1 - Criando endpoint GET

Nosso cliente nos informou que precisa de uma API que retorne a lista completa de produtos cadastrados no sistema.

Para isso, vamos desenvolver uma API no endpoint /api/loja/produtos com o método HTTP GET, e retornar a lista de produtos que está no nosso banco de dados.

1. Abra a classe "exercicios.cspRest.Api" no seu VsCode;
2. Identifique o método ListaProdutos a ser implementado;
3. Configure a rota /produtos [GET] no seu UrlMap;
4. Implemente o método (se quiser ajuda, olhe a classe exemplos.cspRest.Api);
5. Configure a Aplicação Web no seu Portal de Administração;
6. Faça um teste pelo Postman/Insomnia/Curl.



HTTP Curso APIs IRIS / CspRest / Lista Produtos Loja

GET {{url}} /api/loja/produtos

Params Authorization Headers (6) Body Scripts Settings

Query Params

	Key
	Key

Body Cookies Headers (9) Test Results

{ } JSON ▾ Preview Visualize

```
1 [  
2 {  
3   "id": "1",  
4   "nome": "Camiseta",  
5   "preco": 12.34,  
6   "quantidadeEstoque": 10999  
7 },  
8 {  
9   "id": "2",  
10  "nome": "Sapato",  
11  "preco": 56.78,  
12  "quantidadeEstoque": 10998  
13 }
```

1 - Criando endpoint GET

Caso a configuração e implementação forem bem sucedidas, você verá um resultado como o ao lado.

2 - Criando endpoint POST

Agora que nosso cliente consegue visualizar os produtos cadastrados, ele nos pediu para criar uma nova API para realizar o cadastro de novos produtos.

Vamos então desenvolver uma API no endpoint /api/loja/produto com o método HTTP POST, receber um JSON com dados de produto no corpo da requisição e o salvar no nosso banco de dados.

```
{  
  "nome": "Vestido",  
  "preco": 90.12,  
  "quantidadeEstoque": 10997  
}
```

1. Abra a classe "exercicios.cspRest.Api" no seu VsCode;
2. Identifique o método CriaProduto a ser implementado;
3. Configure a rota /produto [POST] no seu UrlMap;
4. Implemente o método (se quiser ajuda, olhe a classe exemplos.cspRest.Api);
5. Faça um teste pelo Postman/Insomnia/Curl.

HTTP Curso APIs IRIS / CspRest / Cria Produto Loja

POST

Params Authorization Headers (8) **Body** Scripts

none form-data x-www-form-urlencoded raw

```
1 {  
2   "nome": "Vestido",  
3   "preco": 90.12,  
4   "quantidadeEstoque": 10997  
5 }
```

Body Cookies Headers (9) Test Results |

{ } JSON Visualize |

```
1 {  
2   "id": "3"  
3 }
```

2 - Criando endpoint POST

Caso a configuração e implementação forem bem sucedidas, você verá um resultado como o ao lado.

Aproveite e chame novamente a sua API de listagem de produtos. Seu produto já está aparecendo lá também?

API REST

Funcionalidades avançadas do UrlMap

- Algumas outras funcionalidades que podem ser configuradas:

- Definição de parâmetros de URL na rota, como no exemplo:

```
<Route Url="/loja/:idLoja/produto/:idProduto" .../>
```

- Tratar igualmente múltiplos métodos HTTP em uma única rota, como no exemplo:

```
<Route Method="POST,PUT" .../>
```

- Utilizar REGEX na rota, como no exemplo:

```
<Route Url="/class/([^\/]+)/([^\/]+)" .../>
```

- Mapear um prefixo para outra classe, como no exemplo:

```
<Map Prefix="/loja" Forward="pacote.loja.Api" />
```

Roteamento avançado

A ordem dos mapeamentos é extremamente importante, pois a requisição irá seguir a primeira rota que ele se encaixar.

Por exemplo, o método "CadastrarUsuario" nunca será chamado nesta classe, pois a requisição /loja/cadastro já teria sido roteada para a classe exemplos.cspRest.loja.Outros

```
4  /// Exemplo do roteamento de mensagens usando prefixos.  
5  /// A ordem dos mapeamentos é extremamente importante,  
6  /// pois a requisição irá seguir a primeira rota que ele  
7  /// se encaixar.  
8  /// Por exemplo, o método "CadastrarUsuario" nunca será chamado  
9  /// nesta classe, pois a requisição /loja/cadastro já teria sido  
10 /// roteada para a classe exemplos.cspRest.loja.Outros  
11 XData UrlMap  
12 {  
13 <Routes>  
14     <Map Prefix="/loja/admin" Forward="exemplos.cspRest.loja.Admin" />  
15     <Map Prefix="/loja/vendas" Forward="exemplos.cspRest.loja.Vendas" />  
16     <Map Prefix="/loja" Forward="exemplos.cspRest.loja.Outros" />  
17     <Route Url="/loja/cadastro" Method="POST" Call="CadastrarUsuario"/>  
18 </Routes>  
19 }  
20  
21 /// Não será chamado devido a ordem das rotas acima.  
22 Debug | Copy Invocation  
23 ClassMethod CadastrarUsuario()  
24 {  
25     Return $$$OK
```

Exemplo de mapeamento no UrlMap.



OpenAPI

O que é OpenAPI?

- É uma especificação aberta que permite descrever de forma padronizada APIs REST;
- Pode ser escrita em YAML ou JSON;
- Pode ser usada para gerar endpoints dentro do IRIS.

OpenAPI

Gerando endpoints no IRIS com OpenAPI

- Para gerar um endpoint no IRIS a partir de uma especificação OpenAPI, é necessário;
 - Criar ou obter uma especificação OpenAPI 2.0, em formato JSON;
 - Ter uma ferramenta de teste REST, como o **Postman**.
- Na ferramenta de teste REST, é preciso criar e enviar uma requisição HTTP com algumas instruções específicas que veremos em breve;
- Se a requisição obtiver sucesso, o IRIS criará as classes chamadas “disp”, “impl” e “spec” automaticamente no Namespace solicitado.

Classe Disp

Esta classe já é compilada diretamente no ambiente e não pode ser editada manualmente.

Ela deve ser utilizada como "Dispatch Class" na aplicação web desejada.

System > Security Management > Web Applications > Edit Web Application - (security settings)

Edit Web Application

Save Cancel

Edit definition for web application /api/produtos:

Application saved.

Click here to create a new web application.

General Application Roles Matching Roles Cross-Origin Settings

Name: /api/produtos
Required. (e.g. /csp/appname)

Description:

Namespace: WORKSHOP_REST Default Application for WORKSHOP_REST: /csp/workshop_rest Namespace Default Application

Enable Application:

Enable: REST Dispatch Class: ProdutosOpenAPI.dispatch
Required.

Redirect Empty Path:

Use JWT Authentication:

WSGI [Experimental].

CSP/ZEN

Analytics Inbound Web Services Prevent login CSRF attack

Security Settings

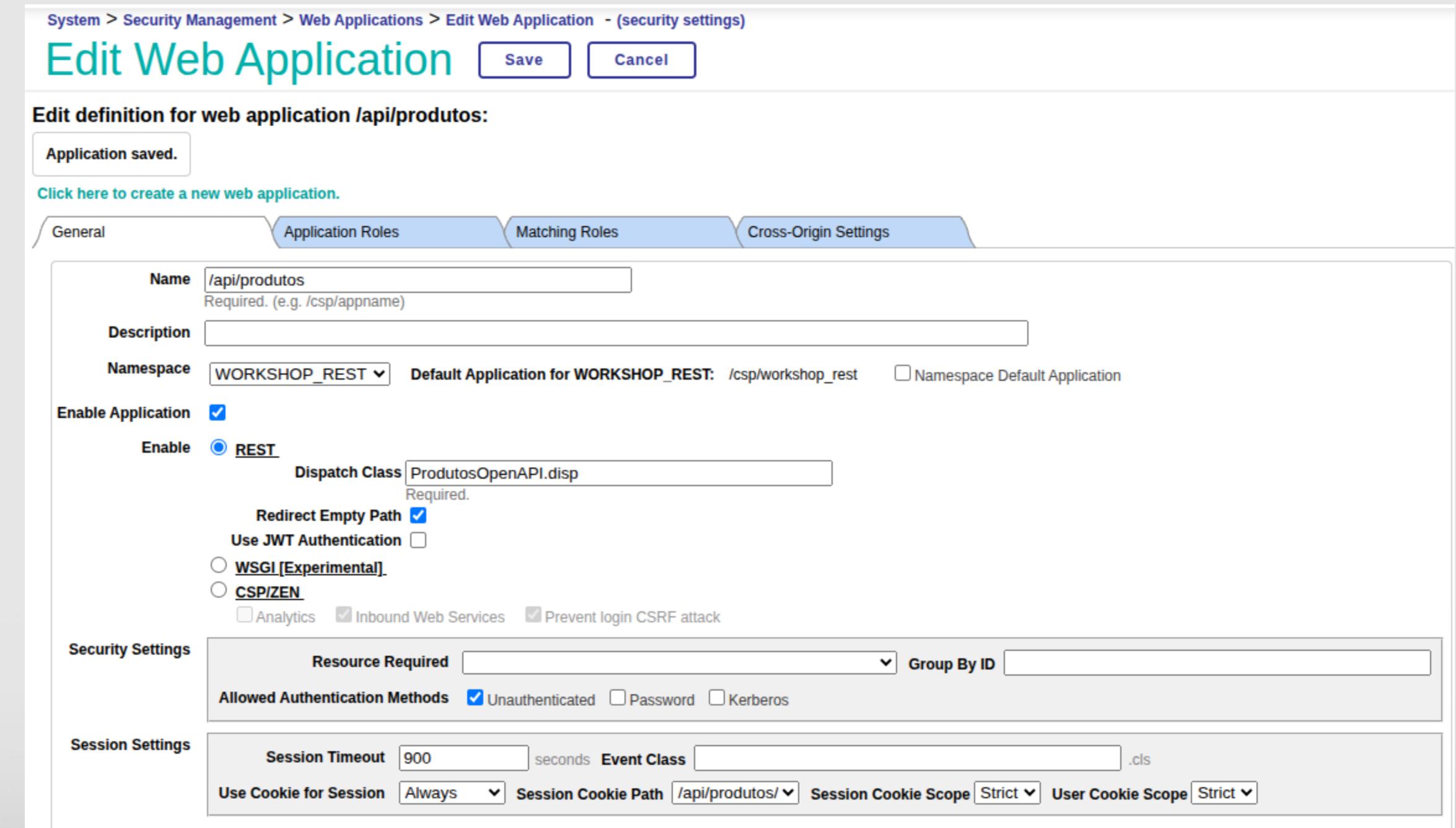
Resource Required: Group By ID:

Allowed Authentication Methods: Unauthenticated Password Kerberos

Session Settings

Session Timeout: 900 seconds Event Class: .cls

Use Cookie for Session: Always Session Cookie Path: /api/produtos/ Session Cookie Scope: Strict User Cookie Scope: Strict



Tela de configuração de aplicações web.

Classe Spec

Nesta classe fica a especificação OpenAPI que foi enviada, no formato JSON.

```
1 Class ProdutosOpenAPI.spec Extends %REST.Spec [ ProcedureBlock ]
2 {
3
4 XData OpenAPI [ MimeType = application/json ]
5 {
6 {
7   "swagger": "2.0",
8   "info": {
9     "title": "API de Produtos (CRUD)",
10    "version": "1.0.0",
11    "description": "Documentação OpenAPI para um CRUD simples de cadastro de produtos."
12  },
13  "basePath": "/v1",
14  "schemes": [
15    "https",
16    "http"
17  ],
18  "consumes": [
19    "application/json"
20  ],
21  "produces": [
22    "application/json"
23  ],
24  "tags": [
25    {
26      "name": "Produtos",
27      "description": "Operações relacionadas ao cadastro de produtos"
28    }
29  ],
```

ProdutosOpenAPI.spec



Classe Impl

Nesta classe são criados os métodos base a ser implementados pelo desenvolvedor.

O nome de cada método vem do campo "operationId" na documentação OpenAPI enviada.

```
1  /// Documentação OpenAPI para um CRUD simples de cadastro de produtos.  
2  /// Inclui endpoints para listar, criar, consultar, atualizar e excluir produtos.<br/>  
3  /// Business logic class defined by OpenAPI in ProdutosOpenAPI.spec<br/>  
4  /// Updated Oct 15, 2025 14:45:17  
5  Class ProdutosOpenAPI.impl Extends %REST.Impl [ ProcedureBlock ]  
6  {  
7  // If ExposeServerExceptions is true, then details of internal errors will be exposed.  
8  Parameter ExposeServerExceptions = 0;  
9    
10 // Retorna uma lista paginada de produtos. Suporta busca por nome e paginação.<br/>  
11 // The method arguments hold values for:<br/>  
12 //     page, Número da página (1-based)<br/>  
13 //     limit, Quantidade de itens por página<br/>  
14 //     search, Texto para buscar pelo nome ou descrição<br/>  
15 // Debug | Copy Invocation  
16 ClassMethod ListaProdutos(page As %Integer, limit As %Integer, search As %String) As %DynamicObject  
17 {  
18     //Place business logic here)  
19     //Do ..%SetStatusCode(<HTTP_status_code>)  
20     //Do ..%SetHeader(<name>,<value>)  
21     //Quit (Place response here) ; response may be a string, stream or dynamic object  
22 }  
23   
24 // Cria um produto e retorna o registro criado.<br/>  
25 // The method arguments hold values for:<br/>  
26 //     product, Objeto do produto a ser criado<br/>  
27 // Debug | Copy Invocation  
27 ClassMethod CriaProduto(product As %DynamicObject) As %DynamicObject  
28 {  
29     //Place business logic here)
```

ProdutosOpenAPI.impl.



3 - Gerando endpoints com OpenAPI

Neste exercício vamos criar um CRUD de Produtos utilizando uma documentação OpenAPI 2.0 no formato JSON.

Primeiramente, utilizaremos o serviço /api/mgmt do IRIS para gerar as classes da API e, em seguida, faremos implementações simples utilizando a classe .Impl.

1. Na sua collection Postman há uma requisição chamada "Criar API Produtos com OpenAPI";
2. Configure o seu host e porta na variável da collection.
3. Para a autenticação, selecione “Basic auth” e escreva o nome de usuário e senha que você utiliza no Portal de Administração;
4. Envie a requisição;
5. No VsCode, exporte do servidor as classes geradas e implemente um simples retorno para cada método usando o Write;
6. Crie uma nova aplicação web de nome "/api/openapi" com a classe .Disp como Dispatch Class. Não se esqueça de atribuir a Application Role %All.



3 - Gerando endpoints com OpenAPI

The screenshot shows the Postman interface with the following details:

- HTTP:** Curso APIs IRIS / OpenAPI / Testar API GET Produto
- Method:** GET
- URL:** {{url}} /api/openapi/products
- Body:** none (selected)
- Headers:** (6) (selected)
- Body Content:** { } JSON (selected)
 - 1 {
 - 2 | "message": "success"
 - 3 }

Se você conseguiu fazer o exercício com sucesso, deve conseguir chamar no Postman a requisição "Testar API GET Produto".

O resultado deve ser como a imagem ao lado.



Segurança

Entendendo e aplicando segurança em APIs no IRIS



Autenticação

O que é autenticação?

- É um processo que verifica a identidade de um cliente ou usuário de uma API REST;
- Garante que o cliente tenha as permissões necessárias para acessar os recursos da API, prevenindo o acesso não autorizado.
- Alguns modos de autenticação comuns são:
 - **Autenticação básica (Basic Auth);**
 - **API Key;**
 - **Autenticação baseada em token;**
 - **OAuth 2.0;**
 - **JWT.**



Autenticação

Qual modo de autenticação utilizar?

- Depende.
- Fatores como segurança, complexidade de implementação, escopo da aplicação e requisitos de uso devem ser levados em consideração;
- Diferentes tipos de autenticação atendem a diferentes necessidades.



Basic Auth

Para configurar a autenticação básica (Basic Auth), basta ativar o checkbox “Senha”.

É necessário utilizar um usuário válido cadastrado no Portal de Administração para se autenticar.

Edita a definição de aplicativo web /api/example:

General	Application Roles	Matching Roles	Cross-Origin Settings	Percent Class Access
<p>Nome: /api/example Obrigatório. (por exemplo /csp/appname)</p> <p>Descrição:</p> <p>NameSpace: USER <input type="button" value="▼"/> Aplicativo padrão para USER: /csp/user <input type="checkbox"/> Aplicação Padrão do Namespace</p> <p>Ativar aplicativo: <input checked="" type="checkbox"/></p> <p>Habilitar: <input checked="" type="radio"/> REST <input type="radio"/> WSGI [Experimental] <input type="radio"/> CSP/ZEN</p> <p>Expedir Classe: Example.API.Router Obrigatório.</p> <p>Redirecionar caminho vazio <input type="checkbox"/></p> <p>Enable Traces <input type="checkbox"/></p> <p>Usar autenticação JWT <input type="checkbox"/></p> <p>Análise <input type="checkbox"/> Serviços de Web de Entrada <input checked="" type="checkbox"/> Impedir ataque CSRF de login <input type="checkbox"/> Enable Traces</p>	<p>Recurso obrigatório: <input type="text"/> Agrupar por ID: <input type="text"/></p> <p>Métodos permitidos para autenticação: <input type="checkbox"/> Não Autenticado <input checked="" type="checkbox"/> Senha <input type="checkbox"/> Kerberos <input type="checkbox"/> OAuth2</p>	<p>Configurações de segurança</p> <p>Configurações da sessão</p> <p>Timeout de sessão: 900 Segundos Classe de eventos: <input type="text"/></p> <p>Usar cookie para o controle de sessão: sempre Caminho do cookie de sessão: /api/example/ Escopo do cookie: <input type="text"/></p>		

Exemplo de configuração de Basic Auth.

oAuth 2.0

Edita a definição de aplicativo web /api/example:

General Application Roles Matching Roles Cross-Origin Settings Percent Class Access

Nome Obrigatório. (por exemplo /csp/appname)

Descrição

NameSpace Aplicativo padrão para USER: /csp/user Aplicação Padrão do Namespace

Ativar aplicativo

Habilitar REST WSGI [Experimental] CSP/ZEN

Expedir Classe Obrigatório.

Redirecionar caminho vazio

Enable Traces

Usar autenticação JWT

Análise Serviços de Web de Entrada Impedir ataque CSRF de login Enable Traces

Configurações de segurança Recurso obrigatório Agrupar por ID
Métodos permitidos para autenticação Não Autenticado Senha Kerberos OAuth2

Configurações da sessão Timeout de sessão 900 Segundos Classe de eventos
Usar cookie para o controle de sessão sempre Caminho do cookie de sessão /api/example/ Escopo do co

Para configurar a autenticação baseada no protocolo oAuth 2.0, ative o checkbox “oAuth2”

Exemplo de configuração de oAuth 2.0.

JWT

Para a autenticação baseada em JWT, é preciso ativar o checkbox “Usar autenticação JWT”.

A API receberá alguns endpoints padrão, como o /login e o /refresh.

É possível ajustar o tempo de duração do tokens de acesso e de atualização.

Importante: as endpoints padrão não devem ser adicionados no UrlMap da classe da API.

Edita a definição de aplicativo web /api/example:

General	Application Roles	Matching Roles	Cross-Origin Settings	Percent Class Access
<p>Nome: /api/example Obrigatório. (por exemplo /csp/appname)</p> <p>Descrição:</p> <p>NameSpace: USER</p> <p>Ativar aplicativo: <input checked="" type="checkbox"/></p> <p>Habilitar: REST</p> <p>Expedir Classe: Example.API.Router Obrigatório.</p> <p>Redirecionar caminho vazio: <input type="checkbox"/></p> <p>Enable Traces: <input type="checkbox"/></p> <p>Usar autenticação JWT <input checked="" type="checkbox"/></p> <p>Tempo limite do token de acesso JWT: 60</p> <p>Tempo limite do token de atualização de JWT: 900</p> <p><input type="radio"/> WSGI [Experimental]</p> <p><input type="radio"/> CSP/ZEN</p> <p><input type="checkbox"/> Análise</p> <p><input checked="" type="checkbox"/> Serviços de Web de Entrada</p> <p><input checked="" type="checkbox"/> Impedir ataque CSRF de login</p> <p><input type="checkbox"/> Enable Traces</p>	Configurações de segurança	<p>Recurso obrigatório: <input style="width: 150px; height: 20px; border: none; background-color: #f0f0f0; border-radius: 5px;" type="button" value="..."/></p> <p>Agrupar por ID: <input type="checkbox"/></p> <p>Métodos permitidos para autenticação: <input type="checkbox"/> Não Autenticado</p> <p><input type="checkbox"/> Senha</p> <p><input type="checkbox"/> Kerberos</p> <p><input type="checkbox"/> OAuth2</p>	Configurações da sessão	<p>Timeout de sessão: 900 Segundos</p> <p>Classe de eventos: <input style="width: 150px; height: 20px; border: none; background-color: #f0f0f0; border-radius: 5px;" type="button" value="..."/></p> <p>Usar cookie para o controle de sessão: sempre</p> <p>Caminho do cookie de sessão: /api/example/</p> <p>Escopo do cookie: <input style="width: 150px; height: 20px; border: none; background-color: #f0f0f0; border-radius: 5px;" type="button" value="..."/></p>

Exemplo de configuração de JWT.

CORS

O que é CORS?

- Significa “**Compartilhamento de Recursos entre Origens**” (ou Cross-Origin Resource Sharing);
- É um mecanismo para integrar aplicações;
- Define uma maneira de os aplicativos web clientes carregados em um domínio interagirem com recursos em outro domínio;
- Permite que o navegador do cliente verifique com os servidores de terceiros se a solicitação foi autorizada antes de qualquer transferência de dados;
- O navegador realiza uma requisição de verificação (preflight) para garantir que o servidor está de acordo com os requisitos de segurança da origem.

CORS

Configuração em uma API

- É necessário configurar os principais cabeçalhos HTTP na resposta, sendo eles
 - **Access-Control-Allow-Origin**, indica quais origens podem acessar os recursos da API;
 - **Access-Control-Allow-Methods**, indica os métodos HTTP permitidos para a requisição;
 - **Access-Control-Allow-Headers**, indica quais cabeçalhos podem ser usados na requisição;
 - **Access-Control-Allow-Credentials**, indica se a requisição pode incluir cookies ou cabeçalhos de autenticação;
 - **Access-Control-Max-Age**, define o tempo (em segundos) que o resultado de uma requisição de verificação pode ser armazenado em cache pelo navegador.



CORS na API

```
4 Parameter HandleCorsRequest = 1;
5
6 // Método para controlar o CORS
7 ClassMethod OnHandleCorsRequest(url As %Library.String) As %Library.Status
8 {
9     Try
10    {
11        // Permitir endereço de origem
12        Do %response.SetHeaderIfEmpty("Access-Control-Allow-Origin", "https://meu-frontend.com")
13
14        // Permitir envio de cookies/credenciais
15        Do %response.SetHeader("Access-Control-Allow-Credentials","true")
16
17        // Permitir headers customizados na request
18        Do %response.SetHeaderIfEmpty("Access-Control-Allow-Headers", "MEU-HEADER-CUSTOMIZADO")
19
20        // Headers que o cliente pode ler na resposta
21        Do %response.SetHeader("Access-Control-Expose-Headers","Content-Length, X-My-Header")
22
23        // Permitir métodos específicos
24        Do %response.SetHeaderIfEmpty("Access-Control-Allow-Methods", "POST, PUT, PATCH")
25
26        // Por quanto tempo o preflight pode ser cacheado (em segundos)
27        Do %response.SetHeader("Access-Control-Max-Age", "3600")
28    }
29    Catch (exception)
30    {
31        Return exception.AsStatus()
32    }
33
34    Return $$OK
35 }
```

Classe exemplos.cors.ApiPorClasse.

Para configurar as políticas de CORS no IRIS, basta adicionar o parâmetro "HandleCorsRequest" e implementar o método "OnHandleCorsRequest" com as configurações desejadas.

Se o método "OnHandleCorsRequest" não for implementado, por padrão serão adotadas as políticas mais permissivas possíveis, por isso prefira sempre implementá-lo



4 - Protegendo APIs REST: Basic Auth

É sempre importante pensar na segurança de nossas APIs.

Por isso, vamos implementar autenticação básica (Basic Auth) em nossas APIs da loja.

1. Encontre a Aplicação Web criada para os exercícios 1 e 2;
2. Nas configurações de segurança, selecione somente "Basic Auth";
3. Tente chamar a API pelo Postman sem passar parâmetros de segurança (você deve receber um erro);
4. Chame novamente a API pelo Postman, agora passando seu usuário e senha do Portal de Administração como Basic Auth.

HTTP Curso APIs IRIS / CspRest / Lista Produtos Loja

Save Share

GET {{url}} /api/loja/produtos Send

Params Auth Headers (7) Body Scripts Settings Cookies

Auth Type

Basic Auth

The authorization header will be automatically generated when you send the request.

Learn more about [Basic Auth](#) authorization.

Username: seuUsuario

Password:  

Body JSON Preview Visualize

200 OK • 7 ms • 462 B • Save Response

```
1 [  
2 {  
3   "id": "1",  
4   "nome": "Camiseta",  
5   "preco": 12.34,  
6   "quantidadeEstoque": 10999  
7 },  
8 {  
9   "id": "2",  
10  "nome": "Sapato",  
11  "preco": 56.78,  
12  "quantidadeEstoque": 10998  
13 },  
14 {  
15  "id": "3",  
16  "nome": "Vestido",  
17  "preco": 90.12,  
18  "quantidadeEstoque": 10997
```

4 - Protegendo APIs REST: Basic Auth

Em caso de sucesso, o resultado será como o ao lado.

5 - Protegendo APIs REST: JWT

Precisamos deixar nossas APIs seguras.

Vamos implementar o JWT em nossas APIs do ERP.

1. Encontre a Aplicação Web criada para os exercícios 3;
2. Nas configurações de segurança, selecione somente "Use JWT Authentication";
3. Tente chamar a API pelo Postman sem passar parâmetros de segurança (você deve receber um erro);
4. Chame a API, mas agora no endpoint "/login" passando seu usuário e senha como Basic Auth;
5. Chame novamente a API pelo Postman, agora passando o token do campo "access_token" como Bearer Token.



5 - Protegendo APIs REST: JWT

The screenshot shows the Postman application interface. At the top, the URL is `https://Curso APIs IRIS / CspRest / JWT Login`. Below it, a POST request is being made to `{{url}}/api/loja/login`. The 'Authorization' tab is selected, showing 'Basic...' as the type. The 'Username' field contains 'apiclient' and the 'Password' field contains a masked password. A note states: 'The authorization header will be automatically generated when you send the request. Learn more about [Basic Auth](#) authorization.' In the bottom section, the response status is '200 OK' with a duration of '48 ms' and a size of '715 B'. The response body is displayed as JSON:

```
1 {  
2     "access_token": "eyJhbGciOiJFUzI1NiIsInR5cCI6IkpXVCJ9.  
eyJpYXQiOjE3NjMzMzc0NzMuNzEyMDU4LCJleHAiOjE3NjMzMzc1MzMzc1MzMsImlzcyI6InNpbHZlci1pcmlzLXNhbRib3gv  
SVJJUyIsInN1YiI6ImFwaWVudCIsInNpZCI6ImxtNFV4bE1TTFhGaloyWDNTd0h0SmFNaCIsImFwcCI6Ii9hcGkv  
bG9qYS8ifQ.  
5QgLUh1W54yPwqXYNbB5dk0JJkfo2Qmqx8uZxbW7M2im6XthEIwSnI6ee0juaqMZF_U6_Hfz56xm0pyjAq0MNg",  
3     "refresh_token": "eyJhbGciOiJFUzI1NiIsInR5cCI6IkpXVCJ9.  
eyJpYXQiOjE3NjMzMzc0NzMuNzEyMDU4LCJleHAiOjE3NjMzMzc1MzMzc1MzMsImlzcyI6InNpbHZlci1pcmlzLXNhbRib3gv  
SVJJUyIsInNpZCI6ImxtNFV4bE1TTFhGaloyWDNTd0h0SmFNaCIsImFwcCI6Ii9hcGkvB9qYS8ifQ.  
gLRTipnBeb9H282GX-f_9aFJ86XjiZwnTkR2IMV5UP8h2nQPX70v2Z4Hh6M2TLrdQYLXLkfV-fSKCNvEjngckA",  
4     "sub": "apiclient",  
5     "iat": 1763137473.712058,  
6     "exp": 1763137533  
7 }
```

Em caso de sucesso, o resultado será como o ao lado.

6 - Utilizando CORS

Vamos implementar configuração de CORS em nossa API da loja.

1. Abra a classe "exercicios.cspRest.Api" no seu VsCode;
2. Adicione o parâmetro HandleCorsRequest
3. Implemente o método OnHandleCorsRequest (use o que está na classe exemplos.cors.ApiPorClasse como base);
4. Faça uma requisição à API de Listar Produtos pelo Postman e observe os headers retornados.

HTTP Curso APIs IRIS / CspRest / Lista Produtos Loja

Save Share

GET {{url}} /api/loja/produtos Send

Params Authorization Headers (7) Body Scripts Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL

This request does not have a body

Headers 200 OK • 17 ms • 689 B • Save Response

Key	Value
Date	Mon, 27 Oct 2025 18:33:05 GMT
Server	Apache
ACCESS-CONTROL-ALLOW-CREDENTIALS	true
ACCESS-CONTROL-ALLOW-HEADERS	MEU-HEADER-CUSTOMIZADO
ACCESS-CONTROL-ALLOW-METHODS	POST, PUT, PATCH
ACCESS-CONTROL-ALLOW-ORIGIN	https://meu-frontend.com
ACCESS-CONTROL-MAX-AGE	3600
CACHE-CONTROL	no-cache
EXPIRES	Thu, 29 Oct 1998 17:04:19 GMT
PRAGMA	no-cache
CONTENT-LENGTH	205
Keep-Alive	timeout=120
Connection	Keep-Alive
	application/json

6 - Utilizando CORS

Em caso de sucesso, o resultado será como o ao lado. Basta selecionar a aba "Headers" na resposta do Postman para observar os Headers retornados, incluindo os de CORS.



Construindo APIs II

Construindo APIs com conceitos mais complexos



Manuseando arquivos com APIs

Manuseando arquivos no IRIS

- A classe %File é usada para manipulação de arquivos no SO. Alguns dos seus métodos são:
 - **%New**, cria uma instância que referencia um path no SO;
 - **Open**, abre o arquivo para operações no IRIS;
 - **Close**, fecha o arquivo para liberá-lo no SO.
- A classe %Stream é usada para manipulação de arquivos na memória. Alguns dos seus métodos são:
 - **Write**, escreve conteúdo na Stream;
 - **Read**, lê o conteúdo que está na Stream;
 - **CopyFrom**, copia o conteúdo de outra Stream.

Manuseando arquivos com APIs

Upload de arquivos

- Utiliza-se o método HTTP POST;
- O Content Type da requisição será o multipart/form-data;
- Neste Content Type podem ser enviados outros dados como chave valor.

Realizando upload

Para subir um arquivo, utilizaremos uma requisição do tipo multipart/form-data.

Pegamos o arquivo usando o método GetMimeType.

Fazemos validações de existência de pasta e arquivo com o mesmo nome.

Por fim, criamos o novo arquivo e escrevemos nele o conteúdo do arquivo enviado na API.

```
73 // Para ler o arquivo da requisição, usar o método GetMimeType  
74 Set source = %request.GetMimeType("file")  
75  
76 // Pegando o nome do arquivo enviado  
77 Set filename = source.FileName  
78  
79 // Caminho da pasta onde ficarão os arquivos  
80 Set basePath = "/files"  
81  
82 // Validando se o caminho existe  
83 If ('##class(%Library.File).DirectoryExists(basePath)  
84 {  
85 | Throw ##class(%Exception.General).%New("Base path not found.", 404)  
86 }  
87  
88 // Setando nome do arquivo. O método NormalizeFilename  
89 // vai tratar o nome do arquivo e retornar o path completo  
90 // para usarmos ao abrir o arquivo  
91 Set filepath = ##class(%Library.File).NormalizeFilename(basePath _ filename)  
92  
93 // Validando se arquivo existe  
94 If (#class(%Library.File).Exists(filepath)  
95 {  
96 | Throw ##class(%Exception.General).%New("There is already a file with this name.", 409)  
97 }  
98  
99 // Instanciando uma referência ao arquivo na memória  
100 Set file = ##class(%Library.File).%New(filepath)  
101  
102 // Abrindo o arquivo nos modos:  
103 // N - New  
104 // W - Write  
105 // S - Stream  
106 Do file.Open("NWS")  
107  
108 // Copiando a Stream de entrada para o arquivo  
109 Do file.CopyFrom(source)  
110 Do file.%Save()  
111 Do file.Close()
```

Classe "exemplos.arquivos.Api".

Manuseando arquivos com APIs

Download de arquivos

- É utilizado o método HTTP GET;
- Alguns dos Content Types usados são:
 - application/pdf
 - image/jpeg
 - text/csv

```
Debug | Copy invocation
12 ClassMethod Download(filename As %Library.String) As %Library.Status
13 {
14     Try
15     {
16         // Caminho da pasta onde ficarão os arquivos
17         Set basePath = "/files"
18
19         // Validando se o caminho existe
20         If ('##class(%Library.File).DirectoryExists(basePath)')
21         {
22             Throw ##class(%Exception.General).%New("Base path not found.", 404)
23         }
24
25         // Setando nome do arquivo. O método NormalizeFilename
26         // vai tratar o nome do arquivo e retornar o path completo
27         // para usarmos ao abrir o arquivo
28         Set filepath = ##class(%Library.File).NormalizeFilename(basePath _ filename)
29
30         // Validando se arquivo existe
31         If ('##class(%Library.File).Exists(filepath)')
32         {
33             Throw ##class(%Exception.General).%New("File not found.", 404)
34         }
35
36         // Instanciando uma referência ao arquivo na memória
37         Set file = ##class(%Library.File).%New(filepath)
38
39         // Abrindo o arquivo nos modos:
40         // R - Read
41         // S - Stream
42         Do file.Open("RS")
43
44         // Escolhendo o ContentType apropriado
45         Set %response.ContentType = "application/pdf"
46
47         // Escrevendo o arquivo até o final no output da API
48         While ('file.AtEnd')
49         {
50             Write file.Read()
51         }
52     }
53 }
```

Classe "exemplos.arquivos.Api".

Realizando download

Para fazer o download de um arquivo precisamos primeiramente instanciar esse arquivo em uma Stream.

Feito isso, é necessário atribuir o valor "application/pdf" no ContentType da resposta da API.

Por fim, fazemos a leitura do arquivo para o output da API usando um loop para ler o arquivo por completo.



7 - Enviando arquivos via API

Vamos construir uma API de upload de arquivos.

A parte de salvar o arquivo no IRIS já foi feita por outro desenvolvedor. Cabe a você fazer o lado do recebimento do arquivo na API.

1. Abra a classe `exercicios.arquivos.Api`;
2. Identifique o método `Upload` a ser implementado;
3. Leia os comentário para entender o que falta ser feito no método;
4. Complete o método com o que falta (Use a classe "`exemplos.arquivos.Api`" como referência caso tenha dificuldades);
5. Chame a API pelo seu Postman e faça o Upload de um arquivo;
6. Confira a pasta `/iris/arquivos` no seu VsCode para ver se o arquivo chegou lá.



7 - Enviando arquivos via API

The screenshot shows the Postman application interface. At the top, it displays the URL: `HTTP Curso APIs IRIS / Arquivos / Upload arquivo`. Below this, the method is set to `POST` and the endpoint is `{{url}}/api/arquivos/upload`. The `Body` tab is selected, showing the configuration for a file upload. Under the `Body` section, the `form-data` option is selected. A table is present with one row, where the key is `file` and the value is a file named `sample-1.pdf`. The status bar at the bottom indicates a successful response: `200 OK`, `13 ms`, `263 B`.

A requisição com sucesso do Postman deve estar como a ao lado.

8 - Recebendo arquivos via API

Vamos construir uma API de download de arquivos.

A parte de ler o arquivo no IRIS já foi feita por outro desenvolvedor. Cabe a você fazer o lado do envio do arquivo na API.

1. Abra a classe `exercicios.arquivos.Api`;
2. Identifique o método `Download` a ser implementado;
3. Leia os comentário para entender o que falta ser feito no método;
4. Complete o método com o que falta (Use a classe "`exemplos.arquivos.Api`" como referência caso tenha dificuldades);
5. Chame a API pelo seu Postman e faça o Download de um arquivo que você fez upload previamente.

HTTP Curso APIs IRIS / Arquivos / Download arquivo

Save Share

GET {{url}} /api/arquivos/download/sample-1.pdf Send

Params Authorization Headers (6) Body Scripts Settings Cookies

Query Params

Key	Value	Description
Key	Value	Description

Body 200 OK 8 ms 55.72 KB Save Response

Hex Preview Visualize

Your Company
123 Your Street
Your City, ST 12345
(123) 456 - 7890

Product Brochure

September 04, 20XX

Product Overview

Lore ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.

Lorem ipsum
Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming id quod mazim placerat facer possim assum. Typi non habent claritatem insitam; est usus legentis in iis qui facit eorum claritatem.

Lorem ipsum
Duis autem vel eum iriure dolor in hendrerit in

8 - Recebendo arquivos via API

A requisição com sucesso do Postman deve estar como a ao lado.



APIs e Production

Conhecendo a interação de Productions do IRIS com APIs



Productions do IRIS

O que é Production?

- Uma **Production** (ou Produção) é um framework de integração para conectar sistemas facilmente e desenvolver aplicações para interoperabilidade;
- Pode integrar vários sistemas de software diferentes;
- Inclui elementos que se comunicam com sistemas externos, bem como elementos que executam processamento interno à Production.

Productions do IRIS

Componentes da Production

- Os componentes em uma produção são conhecidos como **Business Hosts**;
- Existem três tipos de Business Hosts:
 - **Business Services**, recebem/criam requisições de entidades externas à Production e as transmitem para os hosts internos.
 - **Business Processes**, recebem requisições de outros hosts (Business Services ou de outros Business Processes) e processam as requisições e/ou as encaminham. São responsáveis pela lógica de negócios;
 - **Business Operations**, recebem requisições de hosts internos à Production e as processam e/ou as retransmitem para entidades externas.

Productions do IRIS

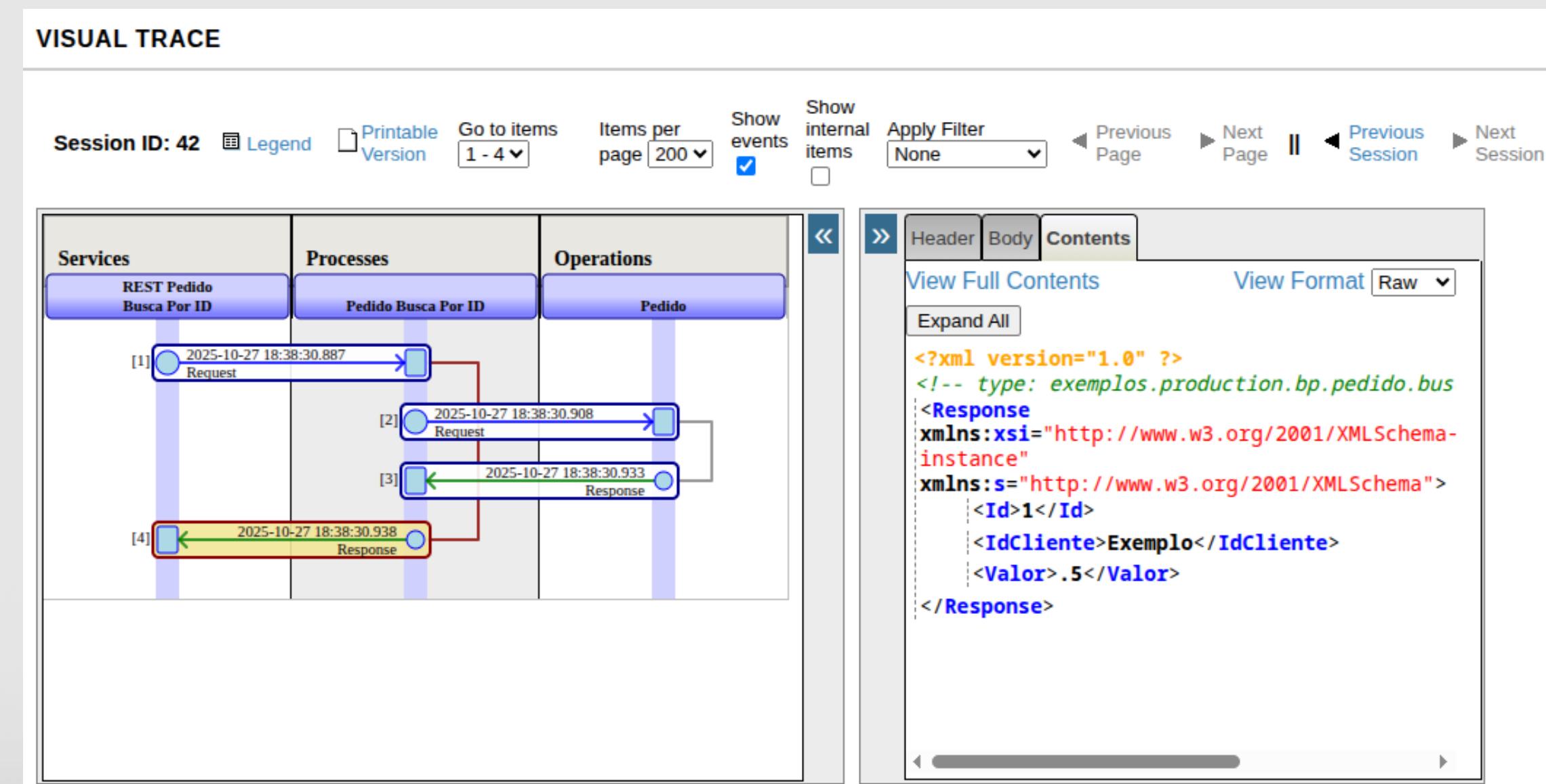
Diagrama de Rastreamento Visual

- Os Business Hosts se comunicam entre si através de **mensagens**;
- O Diagrama de Rastreamento Visual nos permite visualizar as mensagens que foram trafegadas durante uma integração;
- As mensagens podem ser de requisição ou resposta, e elas são compostas por seus metadados e o conteúdo que foi trafegado.

Diagrama na prática

No Diagrama, vemos que em cada coluna está um componente da Production, e em cada linha há uma chamada feita entre um componente e outro.

Ao clicar numa chamada, podemos ver os detalhes da mensagem que foi trafegada.



Exemplo de Diagrama de Rastreamento Visual

Productions do IRIS

Como chamar a Production a partir da API

- Uso da classe Ens.Director para chamar um componente da Production;
- Classes de Request dos componentes;
- O BS tem que implementar o método OnProcessInput;
- O API chamará o BS com o método ProcessInput.

API chamando BS

```
11 ClassMethod BuscaPedidoPorId(id) As %Library.Status
12 {
13     Set %response.ContentType = "application/json"
14     Set %response.CharSet = "utf-8"
15
16     Try
17     {
18         // Usando o Ens.Director para pegar uma
19         // referência ao BS que está na Production
20         Do ##class(Ens.Director).CreateBusinessService("REST Pedido Busca Por ID", .businessService)
21
22         // Instanciando classe de Request do BS
23         Set bsRequest = ##class(exercicios.production.bs.pedido.buscaPorId.Request).%New()
24
25         // Preenchendo a classe de Request com dados necessários
26         Set bsRequest.Id = id
27
28         // Iniciando o BS através do ProcessInput
29         Do businessService.ProcessInput(bsRequest, .bsResponse)
30
31         // Usando as variáveis retornadas na Response do BS
32         Set %response.Status = bsResponse.HttpStatus
33         Write bsResponse.Body
34     }
35     Catch (exception)
36     {
37         Return exception.AsStatus()
38     }
39
40     Return $$OK
41 }
```

Classe exemplos.production.Api

Primeiramente usamos o Ens.Director para criar uma referência ao BS pelo seu nome na Production.

Em seguida, é criada e preenchida a instância da classe de Request do BS.

Chamamos o método ProcessInput para iniciar a execução do BS.

Por fim, usamos o objeto de resposta do BS para retornar dados ao cliente da API.



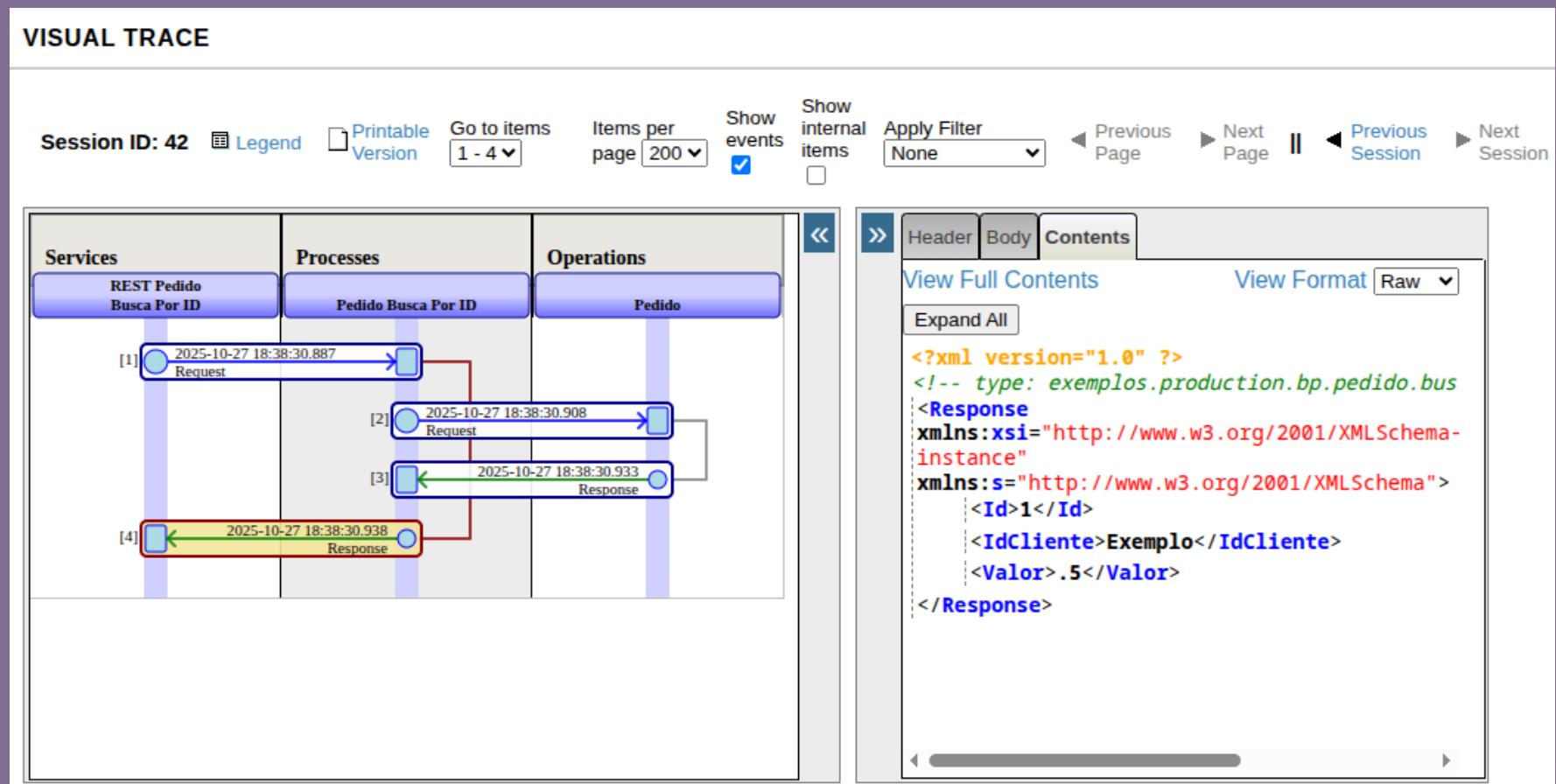
9 - Criando uma API para Production

Nesta API iremos fazer a busca de Pedidos no nosso Banco de Dados.

Chamaremos componentes na Production para ter uma visão clara dos dados que estão sendo trafegados.

1. Abra a classe `exercicios.production.Api`;
2. Identifique o método a ser implementado;
3. Leia os comentário para entender o que falta ser feito no método;
4. Complete o método com o que falta (Use a classe "`exemplos.production.Api`" como referência caso tenha dificuldades);
5. Chame a API pelo seu Postman;
6. Abra a Production no Portal de Administração e encontre o Diagrama de Rastreamento Visual da requisição feita pelo Postman.

9 - Criando uma API para Production



O Diagrama de Rastreamento Visual deve estar similar à imagem ao lado.



IAM

InterSystems Application Manager

bridgergroup.com.br



O que é o IAM?

- O **InterSystems API Manager** (IAM) é um componente embarcado nos produtos InterSystems IRIS;
- Permite monitorar, controlar e gerenciar tráfego de APIs HTTP expostas pelo IRIS ou por outros produtos;
- Disponibiliza recursos de segurança, como configuração de autenticação nas APIs expostas permitindo vários métodos como Basic, OAuth, JWT, SAML.

Configuração de Gateway Services

Através desta opção configuramos os serviços informando a URL de acesso, protocolo utilizado, IP, Porta, Timeouts.

A um serviço podemos associar Rotas e Restrições (autenticação, tráfego) através dos Plugins.

Uma Restrição aplicada ao serviço afetará todas as suas Rotas.

The screenshot shows the InterSystems API Management web interface. The top navigation bar includes 'Workspaces', 'Dev Portals', 'Vitals', and 'Teams'. On the left, a sidebar menu lists 'Dashboard', 'API Gateway' (which is selected), 'Gateway Services', 'Routes', 'Consumers', 'Plugins', 'Upstreams', 'Certificates', 'SNIs', 'Keys', 'Vitals', and 'Secrets'. The main content area displays the 'Gateway Services > PedidosIRIS > PedidosIRIS' page. It shows the URL `http://192.168.1.8:52774/api/pedido`, a status of 'Enabled', and tags 'IRIS' and 'Pedido'. A 'Configuration' section on the right lists various service details:

Configuration	Value
ID	e8edb797-7040-4a55-bb41-4135301d0b40
Name	PedidosIRIS
Enabled	Enabled
Last updated	2025-09-26 01:34:08 +0000
Created	2025-09-11 21:20:46 +0000
Protocol	http
Host	192.168.1.8

Exemplo de configuração de Gateway Service.

Rotas de Gateway Services

The screenshot shows the 'Routes' section of the 'PedidosIRIS' service. It displays two routes:

Name	Protocols	Hosts	Methods	Paths	Tags
Status	http	-	GET	/status	STATUS
StatusAtualiza	http	-	PUT	/status	STATUS

Exemplo de rotas do Gateway Service.

As Rotas estão sempre associadas a um serviço específico.

Cada Rota está associada a um método HTTP específico (GET, POST, PUT, etc).

Podemos configurar Restrições específicas para cada Rota, associando Plugins específicos a elas.

Plugins

Existem vários Plugins categorizados por Autenticação, Segurança, Controle de Tráfego, Serverless, Analítico & Monitoramento, Transformação e Log.

Plugins adicionam restrições e controles aos Serviços, Rotas e Consumidores.

The screenshot shows the InterSystems API Management interface. On the left, there is a sidebar with a dark blue background containing the following items:

- BT
- Dashboard
- API Gateway
- Gateway Services (selected)
- Routes
- Consumers
- Plugins (selected)
- Upstreams
- Certificates
- SNIs
- Keys
- Vitals

The main content area has a white background and displays the following information:

InterSystems API Management

Workspaces Dev Portals Vitals Teams

Gateway Services > PedidosIRIS >

PedidosIRIS

<http://192.168.1.8:52774/api/pedido>

Status Enabled Tags IRIS Pedido

Configuration Analytics Routes Plugins Document

Plugins

Plugins allow you to extend Kong's capabilities with features like rate limiting, authentication, and logging. [Learn more](#)

Name	Status	Ordering	Tags
Basic Authentication	Enabled	Static	STATUS
Rate Limiting	Enabled	Static	STATUS

Exemplo de plugins de Gateway Service.

The screenshot shows the InterSystems API Management interface. The left sidebar has a dark blue background with various navigation items: BT, Dashboard, API Gateway (selected), Gateway Services, Routes (selected), Consumers, Plugins, Upstreams, Certificates, SNI's, Keys, Vitals, and Secrets. The main content area has a white background. At the top, it says 'Routes > StatusAtualiza >'. Below that, 'StatusAtualiza' is displayed with 'Tags STATUS' and 'Methods PUT'. A 'Configuration' tab is selected, showing the following details:

ID	c22ce166-ea1a-408f-b312-08bdd51c6209
Name	StatusAtualiza
Gateway Service	PedidosIRIS
Protocols	http
Tags	STATUS
Host(s)	
Method(s)	PUT
Path(s)	/status

Exemplo de configuração de Rota.

Configuração de Rotas

As Rotas estão sempre associadas aos Serviços.

É possível ter um Plugin diferenciado associado a cada rota de um mesmo Serviço.

As Rotas podem possuir um Path que é concatenado a URL do serviço.



Web Gateway

A porta de entrada das APIs

bridgergroup.com.br



Web Gateway

O que é o Web Gateway?

- O **Web Gateway** é uma aplicação utilitária embarcada nos produtos InterSystems IRIS;
- Media a conexão entre o Web Server e a(s) instância(s) do IRIS;
- Suporta os protocolos HTTP, HTTPS e Web Socket;
- Disponibiliza recursos de log, load balancing e failover.



Acesso ao Web Gateway

Para acessar o Web Gateway, basta utilizarmos o endpoint abaixo após o IP e Porta do nosso IRIS:

/csp/bin/Systems/Module.cxw?CSPSYS=17&CSPSYSn=11922856&CSPTKN=

Nesta tela veremos informações da versão do servidor e outras aplicações relevantes, como o Web Server.

The screenshot shows a web browser window with the URL `10.152.0.87:8080/csp/bin/Systems/Module.cxw?CSPSYS=17&CSPSYSn=11447337&CSPTKN=`. The page title is "Web Gateway Management". On the left, there's a sidebar with links like "About Web Gateway", "Management", "System Status", "Test Server Connection", "View Event Log", "View HTTP Trace", "Configuration", "Default Parameters", "Server Access", and "Application Access". The main content area displays server information:

About Web Gateway	
Version:	2023.2.0.227.0
Gateway Build:	2302.1835
OpenSSL Version:	OpenSSL 1.1.1f 31 Mar 2020
Web Server Name:	10.152.0.87
Web Server Type:	Apache Web Server: Apache/2.4.41 (Ubuntu) CSP-Apache_Module/2023.2.0.227.0-2302.1835
Active Interface:	Apache Module
Configuration:	/opt/webgateway/conf/CSP.ini
Event Log:	/opt/webgateway/logs/CSP.log

At the bottom right, it says "Copyright © 1997 - 2025 InterSystems Corporation".

Exemplo do Web Gateway.

Configuração de servidores

The screenshot shows the 'Web Gateway Management' interface. On the left, a sidebar menu includes links for 'About Web Gateway', 'Management' (with 'System Status', 'Test Server Connection', 'View Event Log', and 'View HTTP Trace'), and 'Configuration' (with 'Default Parameters', 'Server Access', and 'Application Access'). The main content area displays the following information:

- Web Gateway Version 2025.2.0.227.0
- Test Server Connection**:
Server connection test was successful: LOCAL (localhost:1972)
\$ZVersion: IRIS for UNIX (Ubuntu Server 22.04 LTS for x86-64) 2025.1.1 (Build 380U) Fri Jul 7 2025 23:49:38 EDT
- Copyright © 1997 - 2025 [InterSystems Corporation](#)

Exemplo do teste de conexão entre o Web Gateway e um servidor.

No Web Gateway você pode configurar e testar a conexão entre o Web Gateway e diversos servidores.

A conexão com os servidores deve ser feita a partir da porta de SuperServer do IRIS (padrão 1972).



Event Log

O Log de Eventos fica habilitado por padrão, e mostra as tentativas de chamadas à aplicações e serviços através do Web Gateway.

Podemos alterar o nível de detalhamento deste Log de acordo com diferentes parâmetros descritos na documentação do Web Gateway:

https://docs.intersystems.com/iris20231/csp/docbook/DocBook.UI.Page.cls?KEY=CGI_oper_config#CGI_config_parms_system_event



The screenshot shows the 'Web Gateway Management' interface. On the left, a sidebar menu includes links for 'About Web Gateway', 'Management', 'System Status', 'Test Server Connection', 'View Event Log' (which is currently selected), and 'View HTTP Trace'. Under 'Management', there are links for 'Default Parameters', 'Server Access', and 'Application Access'. Below the sidebar, a link leads to 'InterSystems IRIS Management' and 'Back to Management Portal'. The main content area is titled 'View Event Log (Date and Time Order)' and contains a 'Clear Log' button. The log entries are displayed in green text. The first entry is a 'HTTP Request' for a GET request to '/api/cep/proxycep/24240010'. The second entry is an 'HTTP Response' with status code 200 OK, containing headers like User-Agent, Accept, Accept-Encoding, and Content-Type, along with a cookie and content-length information. The third entry is another 'HTTP Request' for a GET request to '/api/cep/proxycep/24240010'. The fourth entry is an 'HTTP Response' with status code 200 OK, containing headers like Date, Expires, Pragma, and Content-Length, along with a JSON response body. The fifth entry is another 'HTTP Request' for a GET request to '/api/cep/proxycep/24240010'.

Exemplo do Event Log.

HTTP Trace

The screenshot shows a web-based application titled "Web Gateway" with a sub-section "Web Gateway HTTP Trace Facility". On the left, there's a toolbar with "Trace ON" (highlighted in red), "Trace OFF", "Refresh", and "Clear". Below the toolbar, two log entries are listed: "Wed Nov 19 12:54:24 29" and "Wed Nov 19 12:54:36 2a". The main area displays a trace of a single request. At the top, the request line "GET /api/cep/proxycep/24240010 HTTP/1.1" is shown. The "Host" header is highlighted with a red box. The full request header is detailed below, followed by the response status line "HTTP/1.1 200 OK". The response headers include Content-Type, Cache-Control, Date, Expires, Pragma, and Content-Length. The response body is a JSON object containing address details like cep, logradouro, complemento, unidade, bairro, localidade, Niterói, uf, estado, Rio de Janeiro, regiao, Sudeste, ibge, gla, ddd, and siafi.

Date: Wed Nov 19 12:54:24; Request ID: 29; Session ID: xp0jlc69S6; Remote-Addr: 192.168.190.137

[Show Response](#)

GET /api/cep/proxycep/24240010 HTTP/1.1
Host: 192.168.190.137
Connection: keep-alive
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/138.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate
Accept-Language: pt-BR,pt;q=0.9,en-US;q=0.8,en;q=0.7
Cookie: state-D846921D-5788-11F0-848D-2239D2B16665=5jOrv7mM4MMdJCVpOCgtb1j0WTGCB4RZF2IHT4Og+4%3D%3ASYSADM%3A2%2C0; CSPWSERVERID=hzzbGvRT; IRISSessionToken=5vZ9vBzPWPBaGyv; CSPBrowserId=m1zFUYgeV2fS7vLfYCPckA

[Show Request](#)

HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
Cache-Control: no-cache
Date: Wed, 19 Nov 2025 12:54:26 GMT
Expires: Thu, 29 Oct 1998 17:04:19 GMT
Pragma: no-cache
Content-Length: 279

{\x0a "cep": "24240-010",\x0a "logradouro": "Rua Abel",\x0a "complemento": "",\x0a "unidade": "",\x0a "bairro": "Santa Rosa",\x0a "localidade": "Niterói",\x0a "uf": "RJ",\x0a "estado": "Rio de Janeiro",\x0a "regiao": "Sudeste",\x0a "ibge": "3303302",\x0a "gla": "",\x0a "ddd": "21",\x0a "siafi": "5865"\x0a}

Exemplo do HTTP Trace.

O HTTP Trace fica desligado por padrão, mas podemos habilitá-lo para momentos de Troubleshooting.

Ele vai registrar as chamadas HTTP feitas aos serviços disponibilizados pelo Web Gateway.

Obrigado!

