Documentação  
de apoio para utilização do  
Integrador CIGAM

- Características Gerais -

Sumário

[1. Características do Integrador 3](#_Toc497125257)

[1.1. Objetivo 3](#_Toc497125258)

[1.2. Abrangência 3](#_Toc497125259)

[1.3. Software e hardware necessário 3](#_Toc497125260)

[1.4. Estrutura de funcionamento 4](#_Toc497125261)

[1.5. Processamento dos dados 5](#_Toc497125262)

[1.6. Estrutura dos registros 6](#_Toc497125263)

[1.7. Validação de parâmetros 6](#_Toc497125264)

[1.8. Tabela de integração e log 7](#_Toc497125265)

[1.9. Registros correlatos 7](#_Toc497125266)

[1.10. Segurança 8](#_Toc497125267)

[1.11. Requisição e resposta dos Webservices 8](#_Toc497125268)

[1.11.1. XML 9](#_Toc497125269)

[1.11.2. JSON 10](#_Toc497125270)

[1.11.3. Tipos especiais 11](#_Toc497125271)

[1.11.4. Configurações especiais do banco de dados 11](#_Toc497125272)

[1.12. Log 12](#_Toc497125273)

[1.13. Aplicação de Integração 13](#_Toc497125274)

[2. Utilizando o retorno dos Serviços 14](#_Toc497125275)

[2.1. Xml 14](#_Toc497125276)

[2.2. JSon 15](#_Toc497125277)

[3. Log do Integrador 18](#_Toc497125278)

[3.1. Requisição 18](#_Toc497125279)

[3.2. Métodos 18](#_Toc497125280)

[3.2.1. Listar 18](#_Toc497125281)

[3.2.2. ListarIdCorrelato 19](#_Toc497125282)

[3.2.3. ListarIdIntegracao 19](#_Toc497125283)

[3.2.4. ListarIdLog 20](#_Toc497125284)

[3.2.5. ListarLogIntegrador 21](#_Toc497125285)

1. Características do Integrador
   1. Objetivo

O Integrador é a maneira oficial para que o CIGAM possa integrar-se com aplicativos de terceiros de maneira automática. O Integrador deve contemplar aqueles casos em que o processo de integração seja normal, ou seja, sem a necessidade de intervenções por customizações.

A ideia principal é que seja disponibilizado um pacote (Kit) com a sua respectiva documentação de modo que não sejam necessários conhecimentos avançados ou técnicos para a implantação, treinamento e suporte.

* 1. Abrangência

O Integrador irá disponibilizar os dados gerados no CIGAM para que aplicativos de terceiros possam ter acesso e usar essas informações conforme sua necessidade. Também será fornecida uma camada para que esses aplicativos de terceiros possam enviar dados para o processamento pelo CIGAM.

Quando o CIGAM receber dados, estes deverão estar íntegros a nível de tipagem para que sejam aceitos. A importação irá validar a regra de negócio e recusar os registros com problemas.

O CIGAM fornecerá os seus dados para outros aplicativos para que possam utilizá-los como desejarem. Em nenhum momento o CIGAM irá controlar o status de integração com outros aplicativos.

* 1. Software e hardware necessário

Uma estrutura muito simples é suficiente para disponibilizar a camada de serviços para integrações. Poderá ser utilizado o Windows Server 2003 ou superior com IIS 6 ou superior. Contudo, a homologação do ambiente foi feita no Windows Server 2008 R2 com IIS 7 e portando esse é o ambiente indicado.

Em alguns casos, dependendo principalmente da versão do Windows Server utilizado, a configuração do IIS 6 acabou tornando-se mais trabalhosa devido a incompatibilidade dos demais serviços utilizados no mesmo servidor. Essa situação é contornada automaticamente com o uso do IIS 7, fazendo que não seja indicado o uso de versões do Windows Server que não suportam o IIS 7.

Para execução dos serviços será necessário também o Microsoft .Net Framework na versão 3.5 e o Asp.Net do servidor deverá suportar o modo compatibilidade 32 bits. O Provider (driver de acesso ao banco de dados) nativo do banco de dados também precisa estar instalado no servidor.

Ambientes com servidores Linux com Mono também torna-se uma possibilidade alternativa. Considerando as inúmeras distribuições do sistema operacional bem como as versões dos pacotes e incompatibilidades entre eles, nenhum desses ambientes foi homologado.

A nível de hardware a necessidade também é relativamente simples, considerando que a camada de serviços não irá manter sessão entre servidor e cliente, ou seja, a conexão é iniciada na requisição e encerrada após a resposta. Dessa forma, as configurações de hardware para o servidor precisarão ser avaliadas em função do número de requisições, número de rotinas utilizadas na integração e performance na execução da importação.

* 1. Estrutura de funcionamento

A camada de Web Services trabalha em conjunto com componentes do ERP CIGAM que possibilitam sistemas de terceiros realizarem consultas de modo síncrono e movimentações e cadastros de modo assíncrono, uma vez que as requisições precisam passar por regras de negócio antes de serem inseridas no sistema.

Funciona de forma passiva, os serviços entregam informações sempre que solicitado. Nos serviços de cadastros e movimentos registra o resultado da operação em uma tabela de log que pode ser consultada através de um programa no ERP e também através de serviços disponíveis na própria camada.

O Integrador deve contemplar os casos em que o processo de integração seja normal, ou seja, sem a necessidade de intervenções por customizações. Para estes casos, a necessidade de novos serviços deve ser avaliada pelas partes envolvidas.

A estrutura propõe as seguintes características:

1. Camada de Webservices para que os aplicativos de terceiros possam obter os dados dos cadastros. Cada serviço terá seu nome intuitivo ao dado retornado e conterá a respectiva documentação (seja embutida ou não). Nessa camada não há qualquer processamento dos dados por parte do CIGAM, ou seja, apenas listagens.
2. Camada de Webservices para que os aplicativos de terceiros possam enviar dados para uma rotina ou cadastro específico do CIGAM. Nessa camada os dados são pré-processados pelo CIGAM e retornado o status do aceite a nível de tabela de integração. Nesse caso, se o dado for inconsistente, um código de erro com uma mensagem será retornado ao chamador. Do contrário (caso os dados estiverem consistentes) o ID de integração será retornado para futuras conferências.
3. Camada de objetos de banco de dados para uso alternativo a camada de Webservices. Essa camada possuirá as funcionalidades das duas camadas de Webservices listadas anteriormente. Cabe lembrar que o uso dessa camada é aconselhado somente em casos extremos (para maiores informações verificar com o Suporte Técnico do CIGAM).
   1. Processamento dos dados

Quando as aplicações de terceiros estiverem obtendo dados do CIGAM via consumo do respectivo serviço ou da respectiva camada de objetos de banco de dados, os parâmetros de data e hora do controle de gravação dos registros serão solicitados. Dessa forma se torna possível filtrar os registros de acordo com a data e hora em que estes sofreram alterações. Quando não informados, todos os dados do respectivo cadastro serão retornados. Nenhum controle por flag de leitura ou de integração será controlado pelo CIGAM e caso seja necessário, tal controle será feito pelos aplicativos dos terceiros em suas tabelas.

Também existirão parâmetros para a chave-única do cadastro para que se possa obter dados de apenas um registro (quando informado). O controle de acesso se feito através do parâmetro para a informação do PIN (que será detalhado mais adiante).

Quando as aplicações de terceiros estiverem enviando dados para o CIGAM, tais dados serão inseridos na tabela de integração caso tenham sido aceitos. A importação/processamento somente será feita quando o programa específico da Aplicação de Importação for executado. Caso o dado não seja aceito, nenhum registro será inserido na tabela de integração e apenas a mensagem de erro (com código do erro e descrição) será enviada ao chamador.

Para controle de existência de registros as camadas de consultas irão disponibilizar uma lista com todos os registros cadastrados na tabela. Essa lista conterá somente os campos chaves e com ela será possível que a aplicação do terceiro compare com sua base cadastrada para saber quais registros foram excluídos.

Caso a tabela tenha um grande número de registros ou se existem muitas exclusões, poderá ser criada uma trigger a qual consome o respectivo serviço a cada exclusão (cabe lembrar que a utilização de triggers interfere na performance da aplicação agravando-se ainda mais quando utilizada por requisições de webservices).

* 1. Estrutura dos registros

A tabela de integração irá receber os registros a serem importados pelo CIGAM. Nessa tabela, todos os tipos de cadastros serão incluídos e um campo ‘Tipo’ irá identificar o tipo de registro/rotina que irá fazer a importação.

Exemplo:

**MATCAD** Cadastro de materiais

**EMPCAD** Cadastro de empresas

Essa tabela terá um grande número de campos para contemplar qualquer tipo de cadastro do CIGAM. Somente os campos de ID e tipo serão indexados.

A camada de serviços será a porta de entrada para as integrações. Existirá um serviço para cada tipo de cadastro/registro. Nesse serviço, além dos parâmetros necessários, existirá também o parâmetro PIN que identificará o usuário/aplicativo que está fazendo a requisição.

Existirá um campo para controle do status do registro na tabela de integração. Esse controle será feito no registro máster quando houver relacionamento máster detalhe. Esse controle é necessário para que somente seja importado o relacionamento inteiro e evite que o programa de importação execute um ciclo exatamente no momento em que os dados estejam sendo inseridos na tabela de integração. Logicamente, o status do registro máster será liberado somente após todos os registros de detalhe terem sido incluídos. Também existe a estratégia que o programa que faz a importação sempre inicie pelo registro máster, e nesse caso, inserindo-o por último na tabela de integração, também será garantia de integridade.

Existirá um webservice para listagem da tabela de integração que permitirá a filtrar a consulta por ID da tabela, data da inclusão do registro ou por tipo do registro e ou ambos. Essa consulta auxilia as aplicações de terceiros a controlarem se tal registro integrado já foi importado.

* 1. Validação de parâmetros

No momento da requisição do serviço, caso haja algum parâmetro definido como obrigatório e não informado, será retornada uma mensagem de erro com o nome do parâmetro. Caso tenha mais de uma ocorrência, os demais nomes de parâmetros também serão listados.

Para parâmetros que possuem dependência de outro parâmetro, será exibido um asterisco (“\*”) após o nome.

* 1. Tabela de integração e log

Nomes-lógicos: GEINTEGRADOR (integração) e GEINTEGRADORLOG (log)

Todas as alterações de dados requisitadas para o CIGAM ficarão na tabela de integração até seu processamento pela Aplicação de Integração. Sempre que uma das camadas alterarem os dados dessa tabela será adicionada uma linha da tabela de log.

Como a tabela de integração e log possuem campos com nomes genéricos a qualquer cadastro ou rotina do CIGAM, a consulta bem como a inclusão de dados deverá ser feita pelas respectivas camadas as quais apresentarão os nomes dos campos adequadamente.

As tabelas de integração e log ficarão em um banco separado. Esse banco será acesso pelo Database CIGAM\_INTEGRA em aplicações do CIGAM e pela Aplicação de Integração (componente CGIntegrador.ecf)

Se o banco de dados for diferente ao utilizado pelo CIGAM (inclusive se mudar apenas o database/usuário), deve-se informar uma *connection string* chamada INTEGRACAO no arquivo de configuração. Também deve ser informado o banco de dados nessa nova conexão no parâmetro providerName.

Exemplo:

<add name="CIGAM" connectionString="Server=10.200.0.42;Database=INFRA;User ID=TESTE;Password=ABYZ.;"/>

<add name="INTEGRACAO" connectionString="Server=10.200.0.42;Database=INFRA;User ID=TESTE;Password=ABYZ.;" providerName="02"/>

O código de banco de dados do CIGAM deve ser informado na configuração BANCO\_DADOS da aplicação.

Exemplo:

<add key="BANCO\_DADOS" value="03" />

* 1. Registros correlatos

A tabela de integração poderá ter registros correlatos pendentes de importações, como por exemplo, pedidos versus itens versus grade do item. Neste caso quando o pedido for importado será feita uma verificação se existem itens para esse pedido. Da mesma forma, quando o item do pedido for importado será verificado se existe grade do item. Nesse ponto, somente os registros com status liberado serão avaliados e importados/recursados.

Os registros correlatos que não estão liberados, não serão tratados e, portanto jamais serão reavaliados. Para evitar registros perdidos na tabela de integração, será executada uma rotina de limpeza que irá excluir os registros com ID correlato cujo registro máster não esteja mais pendente de importação. Essa exclusão registrará um log específico.

* 1. Segurança

Não será utilizado o cadastro de usuário nem de direitos do CIGAM, até porque as aplicações de terceiros não podem controlar adequadamente esses dados.

O PIN irá identificar quem está fazendo a requisição e deve estar inserido no arquivo de configurações do ambiente do Web Service (seção <appSettings>).

Conforme exemplo abaixo, as requisições com o PIN “001” podem utilizar os serviços ‘CADMAT’ e ‘CADEMP’. Os demais “002”, “003” e “004” podem utilizar todos os serviços, pois não há restrição atribuída a eles.

Exemplo:

<appSettings>

<add key="pin" value="001;002;003;004"/>  
<add key="001" value="CADMAT;CADEMP"/>

* 1. Requisição e resposta dos Webservices

Serão suportadas as tecnologias SOAP e Rest/JSon sendo que nessa segunda opção os parâmetros devem ser enviados por POST e o nome do método junto na URL. As respostas dos serviços também estarão disponíveis nessas duas tecnologias.

Quando o retorno contiver mais de um registro, a exportação será feita com duas coleções (objetos), uma para as colunas e outra para os dados. Neste caso as colunas ganham cabeçalhos nomeados para referenciar os dados (c1, c2...).

A opção EXPORT\_HEADERS pode ser inserida no arquivo de configurações do ambiente do Web Service (seção <appSettings>) para indicar se a exportação deve ocorrer sempre usando os cabeçalhos ou não, independentemente da quantidade de registros.

Exemplo:

<appSettings>

<add key="EXPORT\_HEADERS" value="N"/>

É possível incluir no arquivo Settings.

Quando o nome da coluna da tabela no banco de dados for diferente do nome utilizado no CIGAM/CGData, na coleção dos nomes de colunas existirá uma tag com essa identificação.

Não serão exportados campos ainda não nomeados (exemplo: Campo31, Campo32...). Exceção: Campos de usuário que são os iniciados com nome “Usr”.

No formato JSON será necessário atributar ambas as propriedades, pois não há como colocar um valor para a tag principal como no XML. Neste caso, o apelido da coluna deverá ficar na tag “db\_alias”.

Como primeiro nó da árvore, em ambos os formatos será utilizado o valor da tradução do Nome Lógico da tabela exportada. As colunas serão agrupadas na coleção “col” e os campos de cada registro na coleção “reg”. Os dados serão formatados com decodificação UTF-8.

Exemplos das respostas:

* + 1. XML

<esmateri>

<col>

<cd\_material>c1</cd\_material>

<descricao>c2</descricao>

<valor\_imposto db\_column=”campo3”>c3</valor\_imposto>

</col>

<reg>

<c1>001</c1>

<c2>Material Teste 1</c2>

<c3>10</c3>

</reg>

<reg>

<c1>002</c1>

<c2>Material Teste 2</c2>

<c3>20</c3>

</reg>

</esmateri>

* + 1. JSON

{

"col": [

{

"cd\_empresa": "c1",

"descricao": "c2",

"valor\_imposto": {

"db\_column": "campo3",

"db\_alias": "c3"

}

}

],

"esmateri": [

{

"c1": "001",

"c2": "Material Teste 1",

"c3": "10"

},

{

"c1": "002",

"c2": "Material Teste 2",

"c3": "20"

}

]

}

Existe um validador para o padrão JSON em: <http://jsonlint.com/>

* + 1. Tipos especiais

Alguns dados terão formatos especiais na requisições e respostas.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tipo do dado | Formato | Máscara | Exemplo | Onde |
| Data | XML | DateTime (conforme ambiente em que o Serviço está rodando) | 01-jan-2013 | Requisição |
| Data | XML | dd/mm/yyyy | 01/01/2013 | Resposta |
| Data | JSON | dd/mm/yyyy  ou  dd-mmm-yyyy | 01/12/2013  ou  01-dec-2013 | Requisição e Resposta |
| Booleano | XML | Bool (conforme tipagem) | True | Requisição |
| Booleano | XML | True  ou  False | True  ou  False | Resposta |
| Booleano | JSON | TRUE,True,true  ou  FALSE,False,false | True  ou  false | Requisição |
| Booleano | JSON | True  ou  False | True  ou  False | Resposta |
| Time | XML e JSON | 161050 | 161050 | Requisição e Resposta |
| Numéricos | XML e JSON | Sem separador de milhar e com ponto (“.”) como separador decimal | 150  120.00  1250.33222 | Requisição e Resposta |

Quando utilizar o formato XML, os valores enviados para parâmetros de tipos especiais, no serviço SOAP o próprio serviço garante o dado de entrada correto. Quando exportado irá seguir conforme a tabela anterior.

Quando utilizar o formato JSON, para os campos do tipo data o dado terá que vir no formato conforme a tabela anterior e para campos do tipo booleano terá que vir um “true” (não importando a capitalização) para que seja verdadeiro, pois para qualquer outra coisa será considerado como falso.

Para campos de dados do tipo hora, terão o dia com 24 horas e sem dois pontos “:” (HHMMSS)

* + 1. Configurações especiais do banco de dados

Para que os serviços acessem corretamente os dados do tipo Date e DateTime, é obrigatório que a sessão com o banco de dados permita a informação de filtros por datas no formato americano “dd-mmm-yyyy” Exemplo: 01-jan-2013, 20-feb-2013 e 12-oct-2013.

Para bancos de dados SQLServer essa parametrização já vem por padrão.

Para bancos de dados Oracle esse parâmetro pode variar de acordo com o idioma do Sistema Operacional no momento da instalação. Neste caso, verifique se o parâmetro de sessão “nls\_language” está definido como “AMERICAN”. Esse valor pode ser ajustado no Registro do Windows para que todas as sessões sejam inicializadas dessa forma. Observar também se o Web Server que hospeda os serviços está com o modo 32bits/x86 habilitado, pois neste caso também é necessário ajustar os “nls\_language” do Client32 do Oracle.

* 1. Log

A tabela de log será praticamente uma cópia da tabela de integração com a adição dos campos para a data e hora da ocorrência/execução. O processo de criação e registro dos Logs será feito da seguinte forma:

1. O Webservice recebe a requisição e verifica se todos os parâmetros são válidos. Caso não sejam, retorna um erro ao usuário e não registra qualquer tipo de inserção na tabela de integração ou de log.
2. Se os parâmetros passados para o webservice forem válidos, insere na tabela de integração, retorna uma mensagem e sucesso ao usuário sem registrar log.
3. Quando o programa da aplicação de integração executar e importar o registro da tabela de integração, um log será criado com os dados importados (inclusive o usuário logado no CIGAM que fez a importação). Após isso, o registro da tabela de integração será apagado.
4. No log terá um campo para saber se o registro foi importado ou recusado. Caso recursado, conterá também o código e a mensagem de erro. Isso se faz necessário para que os registros com erro não fiquem na tabela de integração.
5. Após liberar os registros pendentes será registrado log. Se tentar liberar um registro inexistente será retornado um erro e não será registrado log. Não será testado se está sendo liberado um registro já liberado pois teria também que ser feito um controle para a liberação dos correlatos.

Existirá uma tela on-line para que o usuário/administrador/DBA possa visualizar a tabela de Log.

A tabela de log não será indexada para ganho de performance. Se a tabela do log for grande, poderão ser criados os índices de acordo com as necessidades.

Não será gerado log sobre as consultas de dados efetuadas.

O log poderá ser limpo ou mantido apenas “n” dias conforme a necessidade do administrador do banco de dados (a limpeza será manual por segurança).

* 1. Aplicação de Integração

A camada de serviços e ou de banco de dados irão garantir “apenas” que o dado chegue até à tabela de integração. A importação dos dados é feita por um programa específico da aplicação de integração o qual deve estar preparado para tratar toda a integridade (regra de negócio, correlatos, etc.).

A Aplicação de Integração é o local onde o programa que importa os dados para o CIGAM deve ficar. Essa aplicação terá acesso aos componentes tradicionais do produto (CGData, Kernel, Functions, etc).

Os programas da Aplicação de Integração deverão estar preparados para terem sua chamada automatizada com o Magic xpi.

Será disponibilizado também um componente para Magic xpi extraído a partir da aplicação de integração.

1. Utilizando o retorno dos Serviços
   1. Xml

Os serviços que utilizam o protocolo SOAP, tem como retorno uma String no formato Xml, para melhor visualização destes dados, está disponível na classe “Xml”, do namespace Cigam.Utils, um método estático chamado “XmlStringToDataTable”. Este retorna um objeto do tipo System.Data.DataTable. Segue abaixo um exemplo de implementação.

private DataTable TrataRetornoEmpresa()

{

//Instancia do serviço

CadastroEmpresas.IntegracaoEmpresaXml integracaoEmpresa = new

CadastroEmpresas.IntegracaoEmpresaXml();

//Chamada do método ListarIndividual e atribuição do retorno a uma string

string retornoString = integracaoEmpresa.ListarIndividual("pin", "000001");

//Conversão da string em DataTable

DataTable dataTableRetorno =

Cigam.Utils.Xml.XmlStringToDataTable(retornoString);

return dataTableRetorno;

}

Também pode ser utilizada a classe XmlDocument, do namespace System.Xml para fazer a manipulação dos dados. Segue abaixo um exemplo de implementação.

private XmlDocument TrataRetornoEmpresa()

{

//Instancia do serviço

CadastroEmpresas.IntegracaoEmpresaXml integracaoEmpresa = new

CadastroEmpresas.IntegracaoEmpresaXml();

//Chamada do método ListarIndividual e atribuição do retorno a uma string

string retornoString = integracaoEmpresa.ListarIndividual("pin", "000001");

//Instancia da classe XmlDocument

XmlDocument retornoXml = new XmlDocument();

//Método para carregar o documento

retornoXml.LoadXml(retornoString);

return retornoXml;

}

* 1. JSon

Os serviços que utilizam o protocolo REST, tem como retorno uma String no formato JSon, para trabalhar com estes dados, está disponível a classe “JSonDocument” no namespace Cigam.Utils. Esta classe fornece a interpretação da string retornada além de propriedades e métodos para manipulação dos dados. Seguem abaixo exemplos de como instanciar e alimentar a classe.

**Exemplo 1**

Private DataTable TrataRetornoEmpresa()

{

//Lista de parâmetros que serão enviados na requisição

List<ServiceParameter> parametros = new List<ServiceParameter>(new[]

{

new ServiceParameter("pin","valorPin"),

new ServiceParameter("empresa","000001")

});

//Chama o método para fazer a requisição e atribui ao StreamReader

StreamReader reader = Cigam.Utils.Web.Handlers.GetPostRequestReturn(  
 "http://servidor/CadastroEmpresas.ip.ashx/ListarIndividual",  
 parametros, ResponseTypeString.RestJSon);

//Inicializa um novo JSonDocument a partir do conteúdo do stream

Cigam.Utils.JSonDocument jsonDoc = new Cigam.Utils.JSonDocument(reader.ReadToEnd());

//Cria um novo DataTable a partir do jsonDoc

DataTable dataTableRetorno = jsonDoc.CreateDataTable();

return dataTableRetorno;

}

**Exemplo 2**

private DataTable TrataRetornoEmpresa()

{

//Lista de parâmetros que serão enviados na requisição

List<ServiceParameter> parametros = new List<ServiceParameter>(new[]

{

new ServiceParameter("pin","valorPin"),

new ServiceParameter("empresa","000001")

});

//Chama o método para fazer a requisição e atribui ao StreamReader

StreamReader reader = Cigam.Utils.Web.Handlers.GetPostRequestReturn( "http://servidor/CadastroEmpresas.ip.ashx/ListarIndividual", parametros, ResponseTypeString.RestJSon);

//Instancia um JSonDocument vazio

Cigam.Utils.JSonDocument jsonDoc = new Cigam.Utils.JSonDocument();

//Carrega o documento a partir do próprio Stream

jsonDoc.LoadFromStream(reader.BaseStream);

//Cria um novo DataTable a partir do jsonDoc

DataTable dataTableRetorno = jsonDoc.CreateDataTable();

return dataTableRetorno;

}

**Exemplo 3**

private DataTable TrataRetornoEmpresa()

{

//Instancia um novo JSonDocument

Cigam.Utils.JSonDocument jsonDoc = new Cigam.Utils.JSonDocument();

//Carrega seu conteúdo a partir de um arquivo

jsonDoc.LoadFromFile(@"C:\temp\listarIndividual.txt");

//Cria o DataTable a partir do documento

DataTable dataTableRetorno = jsonDoc.CreateDataTable();

return dataTableRetorno;

}

Esta classe também fornece uma propriedade contendo uma lista genérica que guarda objetos do tipo Cigam.Utils.JSonObject, e esta lista fornece informações sobre os registros da consulta, como o nome da coluna e o valor. Através desta classe também é possível obter o nome original das colunas, caso a consulta possua um cabeçalho de apelidos, ou saber se a consulta retornou exception, success ou query. Veja os exemplos abaixo:

**Exemplo 1**

private bool VerificaExistenciaCodigo(string conteudoJSon, string codigo)

{

//Instancia um novo JSonDocument e inicializa com a string desejada

Cigam.Utils.JSonDocument doc = new Cigam.Utils.JSonDocument(conteudoJSon);

//Varre todos os objetos retornados e procura se algum deles contem o código desejado.

foreach(Cigam.Utils.JSonObject obj in doc.Objects)

{

if(obj.ColumnName.Equals("empresa") && obj.ColumnValue.Equals(codigo))

{

MessageBox.Show("Empresa ja cadastrada.");

return true;

}

}

return false;

}

**Exemplo 2**

private bool VerificaExistencia(string conteudoJSon)

{

//Instancia um novo JSonDocument e inicializa com a string desejada

Cigam.Utils.JSonDocument doc = new Cigam.Utils.JSonDocument(conteudoJSon);

//Verifica se retornou exception

if(doc.IsException)

{

//Procura pelo objeto que contenha contenha a mensagem de erro e a exibe

Cigam.Utils.JSonObject obj = doc.Objects.First(o =>  
 o.ColumnName.ToLower().Equals("message"));

MessageBox.Show(obj.ColumnValue);

return false;

}

//Verifica se é mensagem de sucesso

else if (doc.IsSuccess)

{

//Procura pelo objeto que tenha o Id do registro inserido e exibe o valor

Cigam.Utils.JSonObject obj = doc.Objects.First(o =>   
 o.ColumnName.ToLower().Equals("idregistro"));

MessageBox.Show(obj.ColumnValue);

return true;

}

else if(doc.IsQuery)

{

return true;

}

return false;

}

1. Log do Integrador
   1. Requisição

Soap: <http://servidor/Log.integrador.asmx>

Rest/JSon: <http://servidor/Log.integrador.ashx>/metodo

* 1. Métodos
     1. Listar

Parâmetros

|  |  |  |  |
| --- | --- | --- | --- |
| Nome (o símbolo “\*” indica que é obrigatório) | Tipo | Tamanho | Observações |
| pin\* | string | 20 |  |
| tipoRegistro | string | 1 |  |
| acao | string | 1 |  |
| dataRequisicaoInicial | date |  |  |
| dataRequisicaoFinal | date |  |  |
| horaRequisicaoInicial | time | 6 |  |
| horaRequisicaoFinal | time | 6 |  |

Retorno

Se pin inválido:

O usuário não possui direitos suficientes para executar esta rotina.

Se nenhum registro encontrado:

A consulta não retornou nenhum registro.

Se sucesso:

|  |  |  |  |
| --- | --- | --- | --- |
| Nome | Tipo | Tamanho | Observações |
| idRegistro | int |  |  |
| idIntegracao | int |  |  |
| idCorrelato | int |  |  |
| tipoRegistro | string | 6 | De acordo com o tipo criado por cada serviço que gerou o log |
| acao | string | 1 |  |
| statusRegistro | string | 1 | L=Liberado, P=Pendente |
| dataRequisicao | date |  |  |
| horaRequisicao | date |  |  |
| pin | string | 20 |  |
| dataIntegracao | date |  |  |
| horaIntegracao | time | 6 |  |
| usuarioIntegracao | string | 3 |  |
| tipoMensagem | string | 2 |  |
| codigoMensagem | int |  |  |
| descricaoMensagem | string | 256 |  |
| detalheMensagem | string | 500 |  |

* + 1. ListarIdCorrelato

Parâmetros

|  |  |  |  |
| --- | --- | --- | --- |
| Nome (o símbolo “\*” indica que é obrigatório) | Tipo | Tamanho | Observações |
| pin\* | string | 20 |  |
| idCorrelato | integer |  |  |

Retorno

Se pin inválido:

O usuário não possui direitos suficientes para executar esta rotina.

Se nenhum registro encontrado:

A consulta não retornou nenhum registro.

Se sucesso:

|  |  |  |  |
| --- | --- | --- | --- |
| Nome | Tipo | Tamanho | Observações |
| idRegistro | int |  |  |
| idIntegracao | int |  |  |
| idCorrelato | int |  |  |
| tipoRegistro | string | 6 | De acordo com o tipo criado por cada serviço que gerou o log |
| acao | string | 1 |  |
| statusRegistro | string | 1 | L=Liberado, P=Pendente |
| dataRequisicao | date |  |  |
| horaRequisicao | date |  |  |
| pin | string | 20 |  |
| dataIntegracao | date |  |  |
| horaIntegracao | time | 6 |  |
| usuarioIntegracao | string | 3 |  |
| tipoMensagem | string | 2 |  |
| codigoMensagem | int |  |  |
| descricaoMensagem | string | 256 |  |
| detalheMensagem | string | 500 |  |

* + 1. ListarIdIntegracao

Parâmetros

|  |  |  |  |
| --- | --- | --- | --- |
| Nome (o símbolo “\*” indica que é obrigatório) | Tipo | Tamanho | Observações |
| pin\* | string | 20 |  |
| idIntegracao | integer |  |  |

Retorno

Se pin inválido:

O usuário não possui direitos suficientes para executar esta rotina.

Se nenhum registro encontrado:

A consulta não retornou nenhum registro.

Se sucesso:

|  |  |  |  |
| --- | --- | --- | --- |
| Nome | Tipo | Tamanho | Observações |
| idRegistro | int |  |  |
| idIntegracao | int |  |  |
| idCorrelato | int |  |  |
| tipoRegistro | string | 6 | De acordo com o tipo criado por cada serviço que gerou o log |
| acao | string | 1 |  |
| statusRegistro | string | 1 | L=Liberado, P=Pendente |
| dataRequisicao | date |  |  |
| horaRequisicao | date |  |  |
| pin | string | 20 |  |
| dataIntegracao | date |  |  |
| horaIntegracao | time | 6 |  |
| usuarioIntegracao | string | 3 |  |
| tipoMensagem | string | 2 |  |
| codigoMensagem | int |  |  |
| descricaoMensagem | string | 256 |  |
| detalheMensagem | string | 500 |  |

* + 1. ListarIdLog

Parâmetros

|  |  |  |  |
| --- | --- | --- | --- |
| Nome (o símbolo “\*” indica que é obrigatório) | Tipo | Tamanho | Observações |
| pin\* | string | 20 |  |
| idLog | integer |  |  |

Retorno

Se pin inválido:

O usuário não possui direitos suficientes para executar esta rotina.

Se nenhum registro encontrado:

A consulta não retornou nenhum registro.

Se sucesso:

|  |  |  |  |
| --- | --- | --- | --- |
| Nome | Tipo | Tamanho | Observações |
| idRegistro | int |  |  |
| idIntegracao | int |  |  |
| idCorrelato | int |  |  |
| tipoRegistro | string | 6 | De acordo com o tipo criado por cada serviço que gerou o log |
| acao | string | 1 |  |
| statusRegistro | string | 1 | L=Liberado, P=Pendente |
| dataRequisicao | date |  |  |
| horaRequisicao | date |  |  |
| pin | string | 20 |  |
| dataIntegracao | date |  |  |
| horaIntegracao | time | 6 |  |
| usuarioIntegracao | string | 3 |  |
| tipoMensagem | string | 2 |  |
| codigoMensagem | int |  |  |
| descricaoMensagem | string | 256 |  |
| detalheMensagem | string | 500 |  |

* + 1. ListarLogIntegrador

Parâmetros

|  |  |  |  |
| --- | --- | --- | --- |
| Nome (o símbolo “\*” indica que é obrigatório) | Tipo | Tamanho | Observações |
| pin\* | string | 20 |  |
| idIntegracao | integer |  |  |
| exibirCorrelatos | bool |  |  |
| TipoRegistro | string | 6 |  |

Retorno

Se pin inválido:

O usuário não possui direitos suficientes para executar esta rotina.

Se nenhum registro encontrado:

A consulta não retornou nenhum registro.

Se sucesso:

|  |  |  |  |
| --- | --- | --- | --- |
| Nome | Tipo | Tamanho | Observações |
| idRegistro | int |  |  |
| idIntegracao | int |  |  |
| idCorrelato | int |  |  |
| tipoRegistro | string | 6 | De acordo com o tipo criado por cada serviço que gerou o log |
| acao | string | 1 |  |
| statusRegistro | string | 1 | L=Liberado, P=Pendente |
| dataRequisicao | date |  |  |
| horaRequisicao | date |  |  |
| pin | string | 20 |  |
| dataIntegracao | date |  |  |
| horaIntegracao | time | 6 |  |
| usuarioIntegracao | string | 3 |  |
| tipoMensagem | string | 2 |  |
| codigoMensagem | int |  |  |
| descricaoMensagem | string | 256 |  |
| detalheMensagem | string | 500 |  |