MERCADEIRO

> Documentação ConecSync v 1.0.

Índice

1 - Informações gerais

O que é o ConecSync

Alguns sistemas desenvolvidos pela Conecdata podem ser integrados (recebendo informações) com sistemas de desenvolvedoras parceiras (aka parceiros) por meio de sua API.

Essa integração permite sincronização de dados específicos de cadastros desses parceiros com alguns projetos da Conecdata, esse processo entretanto ocupa desenvolvedores (dos parceiros), leva algum tempo e requer testes para garantir seu correto funcionamento.

Para facilitar integrações, a Conecdata desenvolveu o **ConecSync** (aka script), um script em NodeJs/Typescript que faz ele mesmo os acessos às suas APIs, e por já ter sido desenvolvido e testado, o tempo e o custo da integração são reduzidos drasticamente.

Acesso direto API x Utilização ConecSync

Logo, existem duas maneiras de se integrar com os sistemas da Conecdata, Acessando **diretamente** as APIs (dentro de seu próprio código) e **indiretamente** utilizando-se o ConecSync (que requer modificações mínimas em seus cadastros e apenas algumas modificações em alguns de seus arquivos de configuração).

Normalmente, os parceiros iniciam a integração utilizando o ConecSync, e posteriormente, modificam seus fontes para acessar diretamente nossas APIs.

Na verdade, a situação mais comum é transferir apenas algumas origens para as APIs e manter outras (normalmente a origem **Estoque**) integradas pelo ConecSync (mais adiante será explicado o motivo disso).

Característica	Acesso direto API	Utilização ConecSync
Facilidade	**MENOR** Requer modificações de arquivos fontes e testes de funcionamento.	**MAIOR** Requer apenas modificações mínimas nos cadastros e configuração do script.
Mão de obra/Custo	**MAIOR** Devido à necessidade de ocupação de programadores.	**MENOR** Modificações/configurações necessárias são simples e de rápida implementação.
Velocidade implementação	**MENOR** Adaptações mais complexas que levam mais tempo.	**MAIOR** Ajustes rápidos e simples.
Velocidade integração	**MAIOR** As modificações em seus cadastros são replicadas instantaneamente.	**MENOR** As modificações só são replicadas a cada execução do script.

> Conhecendo os prós e contras, caso você opte por realizar a integração diretamente pelas APIs dos projetos, o restante dessa documentação é irrelevante (pois só diz respeito ao uso do ConecSync) e recomendamos que consulte a

> documentação de cada API em seu lugar.

Passo a passo utilização script

- Instalação do nodejs.
- Baixando do repositório do GitHub.
- Instalação das dependências.
- Configuração.
- Execução periódica.

2 - Instalação do NodeJs

Basicamente, o único programa necessário para a execução do script é o Node Js. Para instalá-lo, siga as recomendações específicas para seu sistema operacional no site https://nodejs.org.

Junto do **nodejs** (necessário para executar o script) será instalado automaticamente o **npm**, um gerenciador de pacotes javascript, que permitirá a instalação de algumas dependências internas do script.

Verificando instalação nodejs

Para confirmar o sucesso da instalação, basta executar o seguinte comando em um terminal:
`node --version`

Se instalado, será exibida a versão do nodejs:
`v14.15.4`

Verificando instalação npm

O mesmo teste pode ser feito para testar a instalação do npm, pelo comando:
`npm --version`

Que exibirá sua versão:
`7.5.4`

LTS x current

O nodejs disponibiliza dias versões para download, a **LTS** (Testada) e **Current** (com as últimas modificações). Opte sempre pela versão LTS igual ou superior às exibidas acima para evitar problemas.

3 - Baixando do repositório do GitHub

A versão mais atualizada do script estará sempre disponível em seu repositório público no Github https://github.com/conecdata/conecsync.

O script pode ser instalado de duas maneiras:

- Clonando seu repositório Gibhub.
- Baixando seu arquivo ZIP.

Clonando o repositório do Github

> Essa modalidade requer o CLI do git instalado. Sua instalação pode ser verificada pelo comando `git --version` em um terminal.

Passos:

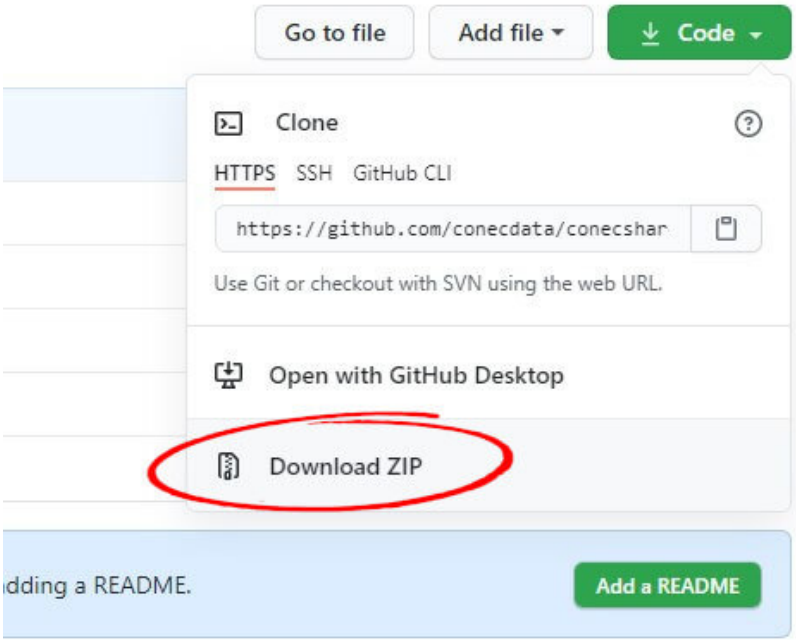
1. Abra um terminal.

2. Acesse a pasta dentro da qual deseja criar a pasta **conecsync** (parent).
3. E execute o comando `git clone https://github.com/conecdata/conecsync`.

Baixando arquivo zip

Passos:

1. Acesse o repositório no github em https://github.com/conecdata/conecshare.git.
2. Baixe o arquivo zip pelo link dentro do botão Code.



3. Descompacte o arquivo baixado dentro da pasta desejada.

> No restante da documentação vamos nos referir à pasta onde o script está instalado como `PASTA_CONECSYNC`.

4 - Instalação das dependências

Apenas os arquivos do próprio script estão disponíveis no repositório, situação comum em arquivos que são projetos nodejs, uma lista completa das dependências que ele precisa baixar/instalar para funcionar corretamente estão relacionadas no arquivo `PASTA_CONECSYNC/package.json`.

Para instalar as dependências do projeto, siga os seguintes passos:

1. Abra um terminal.
2. Acesse a pasta do script (`PASTA_CONECSYNC`).
3. Execute o comando `npm install` (Windows) ou `sudo npm install` (Linux ou Mac).
4. Aguarde o download das dependências. Elas serão gravadas dentro da pasta `PASTA_CONECSYNC/node_modules`.

5 - Configuração

Basicamente a utilização do script consiste em ler uma ou mais origens de dados e transferir suas informações para lojas de um ou mais projetos da Conecdata. Vamos conhecer melhor essas partes envolvidas antes de ver detalhes de suas configurações.

- Origens de dados
- Projetos

O que são as origens de dados?

Origem de dados (aka origem), é um cadastro (ou de um grupo de cadastros) do parceiro em um formato contendo (pelo menos) os campos obrigatórios e com nomes específicos requeridos pelo script.

As origens de dados no script pode ser de dois tipos:

- Arquivos CSV.
- VIEWS no banco de dados.

Origens em arquivos CSV

Para cada origem de dados que deseja utilizar, o parceiro deve, ele mesmo, ler seus cadastros e gravar um arquivo em uma pasta específica (sua configuração será mostrada mais adiante) no formato CSV usando o ';' (Ponto e vírgula) como delimitador.

Exemplo `PASTA_CSVS/formas-pgto.csv`

```
id_interno;nome_forma;id_externo;id_loja;forma_ativa
1;Dinheiro;dinheiro;1;1
2;Debito - Elo;debito_elo;1;1
3;Debito - Maestro;debito_maestro;1;1
4;Debito - Redeshop;debito_redeshop;1;1
5;Debito - Visa Electron;debito_visa_electron;1;1
6;Credito - American express;credito_amex;1;1
...
```

> O ConecSync sempre utiliza ';' (Ponto e vírgula) como delimitador de colunas.

Origens em views

Para cada origem de dados que deseja utilizar, o parceiro deve gerar uma view em seu cadastro por meio de um comando similar ao seguinte:

Exemplo de criação de view formas-pgto para MySql

```
DROP VIEW IF EXISTS view_conecdata_formas;

CREATE VIEW
  view_conecdata_formas
AS SELECT
  fpg_pk AS id_interno,
  fpg_c_forma AS nome_forma,
  fpg_c_id_externo AS id_externo,
  1 AS id_loja,
  1 AS forma_ativa
FROM
  formas_pgto
WHERE
  fpg_c_id_externo IS NOT NULL;
```

> Os comandos para criação de views variam de um tipo de banco de dados para outro. Existem exemplos de criação de views para cada banco de dados suportado pelo script, comentados dentro do arquivo de configuração de cada origem.

Bancos de dados suportados atualmente

- MySql
- Maria Db
- Postgres

- MsSQL
- Firebird

Vantagens da utilização das views para leitura do banco de dados:

- Facilidade de se renomear suas colunas.
- Junção de tabelas para compor as informações necessárias.
- Possibilidade de configuração de um usuário exclusivo para integração e com permissões específicas de leitura apenas das views criadas. Dessa forma, você garante que o script tenha acesso (read-only) apenas às informações relevantes à integração.

> É importante observar que o ConecSync **NUNCA** escreve em suas tabelas, ele apenas lê as views indicadas nas configurações. E como é você quem configura qual view é acessada por cada origem bem como as credenciais de conexão (que podem indicar um usuário com acesso limitado), sabe exatamente o que está sendo lido (e que nunca nada é escrito) pelo script em seus cadastros.

Origens disponíveis

Origem	Descrição	Requerimento
Produtos	Produtos com departamentos (obrigatório) e subdepartamentos (opcional) incluídos.	-
Estoque	Ativa/desativa venda online de produtos.	Origem **Produtos** não indicada.
Formas pagamento	Relacionamento de suas formas de pagamento com nossas, relação completa.	-
Promoções	Promoções (sem indicação dos produtos).	Origem **Produtos promoções**.
Produtos promoções	Produtos presentes em cada promoção.	Origem **Promoções**.

> Para que a integração seja executada, pelo menos uma das origens deve estar configurada corretamente.

> Leia o capítulo Detalhes de origens de dados para maiores detalhes de cada origem.

O que são projetos

Agora que sabemos que as origens de dados são as informações que queremos ler, podendo assim considerá-las uma **origem**, essa leitura será feita com intenção de gravá-las em um ou mais lugares, ou seja algum **destino**.

Esse destino ou destinos, são exatamente os projetos da Conecdata com apis de integração compatíveis com o ConecSync. Mais especificamente gravaremos essas origens dentro de lojas de um ou mais desses projetos.

Projetos atualmente disponíveis:

- Mercadeiro - Marketplace de mercados

Pasta de configuração

Conhecendo origens e projetos e sabendo que as origens podem ser lidas de cadastros e/ou de arquivos CSV, estamos aptos a ver como isso é configurado no ConecSync.

> Nesse capítulo, só vamos explicar as configurações gerais do script, para um exemplo completo de como configurar as origens e destinos, leia o capítulo Exemplo prático.

Todas modificações de configuração do script devem ser feitas nos arquivos dentro da pasta `PASTA_SCRIPT/src/app/config/` ou de suas subpastas.

Arquivo	`PASTA_SCRIPT/src/app/config`
config.ts	/
config-estoque.ts	/origens
config-formas-pgto.ts	/origens
config-produtos-promocoes.ts	/origens
config-produtos.ts	/origens
config-promocoes.ts	/origens
config-mercadeiro.ts	/projetos

> Mais adiante serão explicados alguns **arquivos de log** e a **pasta de armazenagem de hashes** que podem ser modificados/excluídos sem problemas. **TIRANDO ELES E A PASTA DE CONFIGURAÇÃO, NÃO MODIFIQUE MAIS NADA NO SCRIPT.**

Arquivo `PASTA_SCRIPT/src/app/config/config.ts`

Nesse arquivo podemos configurar o seguinte:

- Credenciais de conexões com os bancos de dados.
- Pasta origem para arquivos CSV.
- Selecionar entre modo sandbox (testes de integração) ou produção (integração real).
- Ligar/desligar modo verbose.

Configurando conexão MySql, Maria DB e Postgres.

Caso tenha alguma view criada em algum desses bancos de dados, é necessário indicar suas credenciais de conexão nessa parte da configuração:

PASTA_SCRIPT/src/app/config/config.ts (visão parcial)

```
export const CONFIG = {
  db: {
    conexao: {
      host: '127.0.0.1',
      tabela: 'hypico',
      usuario: 'conecdata_user',
      senha: 'sua_senha_secreta',
      tipo: 'mysql', /* 'mysql' | 'mariadb' | 'postgres' */
    }
  },
  .
  .
  .
}
```

Configurando conexão Firebird.

Caso tenha alguma view criada em um banco de dados Firebird, é necessário indicar suas credenciais de conexão nessa parte da configuração:

PASTA_SCRIPT/src/app/config/config.ts (visão parcial)

```
export const CONFIG = {
  .
```

```
  .
  .
  fb: { // Firebird
    conexao: {
      host: '127.0.0.1',
      port: 3050,
      database: 'C:\\arquivo_firebird.FDB',
      user: 'SYSDBA',
      password: 'masterkey',
      lowercase_keys: false,
      role: null,
      pageSize: 4096
    }
  },
  .
  .
  .
}
```

Configurando pasta origem de arquivos csv.

Caso tenha exportado algum arquivo csv, é necessário indicar a pasta dele(s) nessa parte da configuração:

PASTA_SCRIPT/src/app/config/config.ts (visão parcial)

```
export const CONFIG = {
  .
  .
  .
  csv: {
    path: 'C:\\conecdata_csvs'
  },
  .
  .
  .
}
```

sandbox

Sandbox é um ambiente de testes de integração que grava em um projeto indisponível para compras reais.

Valor	Descrição
true	Acessa uma cópia da plataforma para fins de teste de integração (não disponibilizado para compras reais).
false	Acessa a plataforma real em que os pedidos realizados são enviados de fato para as lojas.

Posteriormente veremos que, para acessarmos tanto a api do modo sandbox, como a do modo produção, precisamos de um token (um tipo de chave que permite acessos de escrita) para cada loja. Como o os tokens de cada um dos modos são diferentes, se o script estiver configurado aqui com um tipo e você indicar um token do outro, todas tentativas de acesso à API retornarão um erro de **Acesso negado**.

PASTA_SCRIPT/src/app/config/config.ts (visão parcial)

```
export const CONFIG = {
  .
  .
  .
```

```
  /*
    TRUE = plataforma de testes
    FALSE = plataforma definitiva ( CUIDADO )
  */
  sandbox: true,
  .
  .
  .

}
```

verbose

Os passos da execução do script sempre são enviados para o arquivo de log `PASTA_SCRIPT/ok.log`, opcionalmente, **{ verbose: true }** permite que ele seja também enviado para o terminal.

PASTA_SCRIPT/src/app/config/config.ts (visão parcial)

```
export const CONFIG = {
  .
  .
  .
  /*
    TRUE = Envia mensagens para terminal (se disponível)
    FALSE = Não envia mensagens, apenas grava no arquivo de log
  */
  verbose: true
}
```

6 - Execução periódica

Cada vez que o script é executado, baseado em suas configurações, ele lê as origens de dados e transfere (apenas as informações que mudaram desde a execução anterior) para as lojas dos projetos.

Para executar o ConecSync, basta executar o comando:
`PASTA_SCRIPT/npm start`

> O ideal é que você inclua a execução do script em algum tipo de agendamento periódico em seu sistema operacional para automatizar o processo de integração.

7 - Origens de dados

IDS

Os valores indicados como **IDS** devem ser utilizados tanto **nas views** (renomeando seus campos) como nos **cabeçalhos nos arquivos CSV** (identificando suas colunas).

Tipos de campos

Tipo	Valores
Boolean	true (**'S'** ou **'T'** ou **'1'** ou **nro > 0**). false Demais valores.
Inteiro	Números **sem** casas decimais.

Tipo	Valores
Número	Números **com/sem** casas decimais.
String	Valores alfanuméricos.

Origem Produtos

ID	Detalhes	Tipo	Obrigatório	Condições/Default
atacado_qtde	Qtde mínima habilitação preço atacado.	Número	Se **atacado_status** true.	Deve ser > 0 se indicado. Default 0.
atacado_status	Status promoção atacado do produto.	Boolean	Não	Default **false**.
atacado_preco	Novo valor produto se **atacado_qtde** atingido.	Número	se **atacado_status** true.	> 0 e < **preco_venda** se indicado. Default 0.
ativo_departamento	Status do departamento do produto.	Boolean	Não	Default **true**.
ativo_produto	Status do produto.	Boolean	Não	Default **true**.
ativo_subdepartamento	Status do subdepartamento do produto.	Boolean	Não	Default **true**.
barcode_produto	Código de barras do produto.	String	Não	Se indicado, produto é **industrializado**. Default ''.
descricao_produto	Informações adicionais do produto.	String	Não	Default ''.
destaque	Status exibição privilegiada do produto.	Boolean	Não	Default **false**.
estoque_controlado	Status indicativo de verificação de estoque.	Boolean	Não	Default **false**.
id_departamento	Sua chave primária do departamento do produto.	Inteiro/String	**SIM**	-
id_produto	Sua chave primária do produto.	Inteiro/String	**SIM**	-
id_subdepartamento	Sua chave primária do subdepartamento do produto.	Inteiro/String	Não	Default ''.

ID	Detalhes	Tipo	Obrigatório	Condições/Default
nome_departamento	Nome do departamento.	String	**SIM**	-
nome_produto	Nome do produto.	String	Se **barcode** = ''.	Default ''.
nome_subdepartamento	Nome do subdepartamento.	String	Se **id_subdepartamento** indicado.	Default **''**.
online_departamento	Status disponibilidade do departamento online.	Boolean	Não	Default **true**.
online_produto	Status de disponibilidade do produto online.	Boolean	Não	Default **true**.
fracionado_status	Status condição pesável do produto.	Boolean	Não	Default **false**.
fracionado_perc_desc_promo_auto	Percentual de desconto automático do produto fracionado.	Número	Se **fracionado_status** true.	>= 0 e <= 100 se indicado. Default **0** (sem desconto).
fracionado_fracao	Qtde adicionada/removida do produto por vez.	Número	Se **fracionado_status** true.	> 0 se indicado. Default **0**.
fracionado_tipo	Unidade de fracionamento do produto.	String	Se **fracionado_status** true.	'K', 'KG', 'G', 'GR', 'L', 'LT', 'ML', 'M' ou 'CM'. Default **''**.
percentual_limite_venda	% máximo do estoque disponível permitido para compra.	Número	Não	>= 0 e <= 100 se indicado. Default **0** (desativado).
preco_venda	Preço normal do produto.	Número	**SIM**	> 0.
qtde_estoque_atual	Qtde atual no estoque.	Número	Se **estoque_controlado** true.	-
qtde_estoque_minimo	Disponibilidade mínima do produto para status regular de estoque.	Número	Se **estoque_controlado** true.	>= 0 se indicado. Default **0**.
qtde_limite_venda	Qtde máxima do produto por pedido.	Número	Não	Default **0** (ilimitada).
id_loja	Sua chave primária da loja do produto.	Número/String	**SIM**	-

Condições para exibição na plataforma

Para que o produto seja exibido na plataforma, TODOS valores abaixo devem ser TRUE.

- **ativo_departamento**
- **ativo_produto**
- **ativo_subdepartamento** (se **id_subdepartamento** indicado)
- **online_departamento**
- **online_produto**

Condições para venda na plataforma

Produtos exibidos na plataforma podem estar indisponíveis para compra caso estejam com seu estoque em quantidade crítica, o que é calculado com base na seguinte lógica:

```
SE (estoque_controlado) {
  ESTOQUE_CRITICO = qtde_estoque_atual < qtde_estoque_minimo
} ELSE {
  ESTOQUE_CRITICO = false
}
```

Atacado

Atacado é um grupo que permite a criação de uma promoção embutida no próprio produto, baseado nas seguintes propriedades.

Propriedade	Descrição
atacado_status	Liga/desliga modo atacado.
atacado_qtde	Quantidade do produto necessária para utilizar **atacado_preco** no lugar do preço.
atacado_preco	Novo preço do produto se **atacado_status** = true e qtde do produto >= **atacado_qtde**.

> Quando habilitado o grupo atacado, uma nova promoção surgirá na lista de promoções do produto.

Produtos industrializados

São produtos que tem a propriedade **barcode_produto** indicada. Produtos industrializados tem algumas propriedades buscadas automaticamente de uma base de produtos da plataforma.

- **nome_produto**

> Além dessas propriedades, imagem(ns) do produto também são buscadas na base de produtos e não podem ser indicadas diretamente, nem pelo script, nem pela API. Depois de cadastrado o produto, novas imagens podem ser adicionadas/modificadas pelo módulo **lojistas** da plataforma.

Produtos em destaque

Produtos marcados como destaque possuem uma exibição privilegiada na plataforma, eles são apresentados na tela principal do site/app e nos resumos dos subdepartamentos, quando um departamento (que não possua produtos próprios) é selecionado.

Produtos fracionados

Produtos fracionados são os que permitem venda a granel, tendo uma unidade e fração indicadas. Esse grupo é composto pelas seguintes propriedades.

- **fracionado_status**
- **fracionado_perc_desc_promo_auto**

- **fracionado_tipo**
- **fracionado_fracao**

Suponhamos que um produto com o nome "Banana Kg" foi cadastrado com fracionado (**fracionado_status** true). Para produtos com o status fracionado ativo, temos que indicar tanto um valor para **fracionado_tipo** como um para **fracionado_fracao**. Para cada tipo fracionado, existem sempre dois valores com correspondência entre si, nesse caso seriam 'KG' (ou 'K') e 'GR' (ou 'G'). Suponhamos que queremos vender esse produto (banana) de 300 em 300 gramas, as duas indicações seguintes surtiriam o mesmo efeito.

Tipo	Fração	Resultado
KG ou K	0, 3	0, 3Kg 0, 6Kg 0, 9Kg 1, 2Kg...
GR ou G	300	0, 3Kg 0, 6Kg 0, 9Kg 1, 2Kg...

> A exibição na plataforma das unidades fracionadas se fará sempre pela de maior unidade (no caso Kg), sendo realizada automaticamente a conversão quando for necessário.

Produtos fracionados não podem constar em promoções, entretanto, caso se queira por exemplo apresentar um produto como: `banana de R$ 4,00 por R$ 3,00`, pode-se indicar o campo **fracionado_perc_desc_promo_auto** com valor 25 (25%), o que exibirá sempre o valor integral do produto (figurativo) junto do valor com desconto automático (utilizado).

Limitando a quantidade no pedido

Existem duas maneiras de se limitar a quantidade de cada produto no carrinho, uma direta (pela propriedade **qtde_limite_venda**) e outra que depende do status e da situação do estoque do produto. A lógica utilizada para se chegar ao valor limite é algo como:

```
SE (estoque_controlado) {
  qtdeLimiteVendaEstoque = percentual_limite_venda / 100 * qtde_estoque_atual;
  IF (qtdeLimiteVendaEstoque > qtde_limite_venda) {
    QTDE_LIMITE_VENDA = qtde_limite_venda;
  } ELSE {
    QTDE_LIMITE_VENDA = qtdeLimiteVendaEstoque;
  }
} ELSE {
  QTDE_LIMITE_VENDA = qtde_limite_venda;
}
```

> Ou seja, se apenas uma das propriedades for indicada (**percentual_limite_venda** ou **qtde_limite_venda**), ela será o valor máximo permitido por pedido para esse produto, caso ambas sejam indicadas, será utilizada a que tiver o menor valor.

Origem Estoque

ID	Detalhes	Tipo	Obrigatório	Condições/Default
barcode_produto	Código de barras do produto.	String	Não	Default **''**.
estoque_controlado	Liga/desliga verificação de estoque desse produto.	Boolean	Não	Default **false**.
id_produto	Sua chave primária do produto.	Inteiro/String	**SIM**	-
nome_produto	Nome do produto.	String	Se **barcode** = ''.	Default ''.
qtde_estoque_atual	Qtde atual no estoque.	Número	Se **estoque_controlado** true.	-

ID	Detalhes	Tipo	Obrigatório	Condições/Default
qtde_estoque_minimo	Disponibilidade mínima do produto para status regular de estoque.	Número	Se **estoque_controlado** true.	>= 0 se indicado. Default **0**.

> Quando a origem **Produtos** está habilitada, o cálculo do estoque já é realizado nela (se configurada para isso), e caso a origem **Estoque** também seja indicada, ela simplesmente será ignorada. Essa origem foi criada para o caso da integração de produtos ser feita com o ERP chamando diretamente a API da plataforma e o cálculo da disponibilidade ou não dos produtos (origem **Estoque**) ser feita pelo ConecSync.

Controlando a disponibilidade dos produtos pela API

Quando uma integradora ajusta seus fontes para acionar a API da plataforma, ela não pode enviar uma modificação do produto a cada venda e/ou entrada de estoques no ERP, sob pena de ter sua integração bloqueada. É desnecessário comunicar todas modificações no estoque (vendas ou entradas) acionando-se a API da plataforma (isso geraria custos exorbitantes para ela uma vez que a tecnologia utilizada para seu desenvolvimento cobra por cada leitura/escrita nos cadastros), e a API só deve ser acionada quando o produto entrar ou sair de um estado de estoque crítico diferente do atual, o que deve ser verificado antes de se chamar a API. Seguindo uma lógica similar à seguinte:

```
// Busca status atual no cadastro
statusEstoqueAtual = db.read(status_critico);
// Calcula novo status
statusEstoqueCritico = qtde_estoque_atual < qtde_estoque_minimo
// Verifica se foi modificado para acionar API
IF (statusEstoqueAtual != statusEstoqueCritico) {
  chamaAPI(
    {
      ESTOQUE_CRITICO: statusEstoqueCritico
    }
  )
  db.write(statusEstoqueCritico)
} ELSE {
  // Não é necessário acionar API
}
```

A utilização do script simplifica esse processo uma vez que ele só aciona a API quando alguma coisa mudou da versão anterior do cadastro para atual, incluindo os valores que são calculados a partir das leituras.

Origem Formas pagamento

ID	Detalhes	Tipo	Obrigatório	Condições/Default
forma_ativa	Habilita/desabilita forma de pgto na plataforma.	Boolean	Não	Default **true**.
id_externo	Nosso ID da forma de pgto.	String	**SIM**	-
id_interno	Sua chave primária da forma de pgto.	Inteiro/String	**SIM**	-
id_loja	Sua chave primária da loja do produto.	Número/String	**SIM**	-
nome_forma	Nome da forma.	String	**SIM**	-

> As formas habilitadas que forem integradas estarão disponíveis no módulo **lojistas** da plataforma para distribuição nas categorias **Retirada**, **Entrega** e **Online**.

Origem Promoções

ID	Detalhes	Tipo	Obrigatório	Condições/Default
descricao	Descrição da promoção.	String	**SIM**	-
id_loja	Sua chave primária da loja do produto.	Número/String	**SIM**	-
id_promocao	Sua chave primária da promoção.	Inteiro/String	**SIM**	-
tipo	Tipo da promoção.	String	**SIM**	**'LP'** (Leve... Pague...) ou **'APD'** (A Partir De...)
lim_desc_apd	Qtde máxima de produtos para aplicação do desconto.	Número	Ignorada se tipo != 'APD'.	Default **0** (ilimitado).
perc_desc_apd	Percentual de desconto da promoção.	Número	Se tipo = 'APD'.	> 0 e <= 100 se indicado. Default **0**.
promocao_ativa	Liga/desliga promoção.	Boolean	Não	Default **true**.
qtde_apd	Qtde mínima para habilitação da promoção.	Número	Se tipo = 'APD'.	> 0 se indicada. Default **0**.
qtde_leve_lp	Qtde mínima a ser atingida para habilitação da promoção.	Número	Se tipo = 'LP'.	> 0 se indicada. Default **0**.
qtde_pague_lp	Qtde do produtos contabilizada a cada múltiplo de **qtde_leve_lp**.	Número	Se tipo = 'LP'.	> 0 e < **qtde_leve_lp** se indicada. Default **0**.

> Para que sejam cadastradas corretamente cada promoção deve ter algum produto relacionado a ela na origem **ProdutosPromocoes**.

Promoção Leve... Pague...

tipo = **LP**. Tipo de promoção em que cada vez que a quantidade **leve** é alcançada, é substituída pela **pague** (sempre menor). Alguns exemplos:

Qtde produto	Preço	Leve	Pague	Total sem desconto	Total com desconto	Detalhes
3	10,00	3	3	3 * 10,00 = 30,00	-	Qtde pague inválida, igual a leve. PROMOÇÃO INVÁLIDA.
3	10,00	3	4	4 * 10,00 = 40,00	-	Qtde pague inválida, maior que leve. PROMOÇÃO INVÁLIDA.
3	10,00	3	2	3 * 10,00 = 30,00	2 * 10,00 = 20,00	3 - 1 desconto, 0 sobras PROMOÇÃO OK.
4	10,00	3	2	4 * 10,00 = 40,00	3 * 10,00 = 30,00	4 - 1 desconto, 1 sobras PROMOÇÃO OK.
6	10,00	3	2	6 * 10,00 = 60,00	4 * 10,00 = 40,00	6 - 2 descontos, 0 sobras PROMOÇÃO OK.

Promoção A partir de...

tipo = **APD**. Tipo de promoção em que a partir de uma quantidade específica, os produtos sofrem um desconto (com ou sem um limite).

Qtde produto	Preço	Qtde APD	% desc. APD	Limite	Total sem desconto	Total com desconto	Detalhes
3	10,00	3	0	0	3 * 10,00 = 30,00	-	% desconto inválido, = 0%. PROMOÇÃO INVÁLIDA.
3	10,00	3	105	0	3 * 10,00 = 30,00	-	% desconto inválido, = 105%. PROMOÇÃO INVÁLIDA.
3	10,00	3	50	0	3 * 10,00 = 30,00	(2 * 10,00) + (1 * 5,00) = 25,00	2 preços cheios, 1 preço descontado. PROMOÇÃO OK.
4	10,00	3	50	0	4 * 10,00 = 40,00	(2 * 10,00) + (2 * 5,00) = 30,00	2 preços cheios, 2 preços descontados. PROMOÇÃO OK.
4	10,00	3	50	1	4 * 10,00 = 40,00	(2 * 10,00) + (1 * 5,00) + (1 * 10,00) = 35,00	2 preços cheios, 1 preço descontado, 1 preço cheio (devido limite 1). PROMOÇÃO OK.
10	10,00	3	50	0	10 * 10,00 = 100,00	(2 * 10,00) + (8 * 5,00) = 60,00	2 preços cheios, 8 preços descontados. PROMOÇÃO OK.

Origem Produtos promoções

ID	Detalhes	Tipo	Obrigatório	Condições/Default
id_produto_promocao	Sua chave primária da integração produtos_promocoes.	Inteiro/String	**SIM**	-
id_produto_promocao_promocao	Id promoção.	String	**SIM**	-
id_produto_promocao_produto	Id produto.	String	**SIM**	-
id_loja	Sua chave primária da loja do produto.	Número/String	**SIM**	-

8 - Exemplo prático

Vamos ver na prática a integração de uma origem buscando seus dados tanto de um arquivo CSV como a partir views (que vamos gerar aqui) em um cadastro MySql.

Como o processo de configuração das de todas origens é idêntico, vamos utilizar uma origem pequena, a **Formas de pagamento** em nosso exemplo.

Lendo de um cadastro

Uma vez que optamos por utilizar um banco de dados MySql, temos que indicar os dados de sua conexão na seguinte parte do arquivo `PASTA_SCRIPT/src/app/config.ts`:

```
export const CONFIG = {
  db: {
    conexao: {
```

```
        host: '127.0.0.1',
        tabela: 'hypico',
        usuario: 'conecdata_user',
        senha: 'sua_senha_secreta',
        tipo: 'mysql', /* 'mysql' | 'mariadb' | 'postgres' */
      }
    },
    .
    .
    .
  }
```

Agora podemos gerar quantas views quisermos nesse cadastro, e lê-las a partir dessa conexão. Vamos fazer isso para a tabela em nosso exemplo.

Estrutura da tabela formas-pgto (MySql)

Campo	Tipo	Tamanho/Valor
fpg_pk	Chave primária	-
fpg_c_forma	String	40
fpg_e_tipo	Enum	'C','D','O'
fpg_c_legenda	String	30
fpg_c_img	String	45
fpg_c_id_conecdata	String	20

Conteúdo da tabela formas-pgto

fpg_pk	fpg_c_forma	fpg_e_tipo	fpg_c_legenda	fpg_c_img	fpg_c_id_conecdata
1	Dinheiro	O	Dinheiro	dinheiro.png	dinheiro
2	Debito - Elo	D	Débito	elo.png	debito_elo
3	Debito - Maestro	D	Débito	maestro.png	debito_maestro
4	Debito - Redeshop	D	Débito	redeshop.png	debito_redeshop
5	Debito - Visa Electron	D	Débito	visa-electron.png	debito_visa_electron
6	Credito - American express	C	Crédito	amex.png	credito_amex
7	Credito - Diners	C	Crédito	diners.png	credito_diners
8	Credito - Elo	C	Crédito	elo.png	credito_elo
9	Credito - Hipercard	C	Crédito	hipercard.png	credito_hipercard
10	Credito - Mastercard	C	Crédito	mastercard.png	credito_mastercard
11	Credito - Policard	C	Crédito	policard.png	credito_policard
12	Credito - ValeCard	C	Crédito	valecard.png	credito_valecard
13	Credito - Visa	C	Crédito	visa.png	credito_visa
14	Cheque	O	Cheque	cheque.png	cheque
15	Alelo - Alimentacao	O	Alimentação	alelo.png	alelo_alimentacao
16	Alelo - Refeicao	O	Refeição	alelo.png	alelo_refeicao

fpg_pk	fpg_c_forma	fpg_e_tipo	fpg_c_legenda	fpg_c_img	fpg_c_id_conecdata
17	Policard - Alimentacao	O	Alimentação	policard.png	policard_alimentacao
18	Policard - Refeicao	O	Refeição	policard.png	policard_refeicao
19	Sodexo - Refeicao	O	Refeição	sodexo.png	sodexo_refeicao
20	Ticket Rest. Eletronico	O	Ticket Rest.	ticket-restaurante.png	ticket_rest_eletronico
21	ValeCard - Alimentacao	O	Alimentação	valecard.png	valecard_alimentacao
22	ValeCard - Refeicao	O	Refeição	valecard.png	valecard_refeicao
23	Visa - Vale	O	Visa Vale	visa-vale.png	visa_vale
24	Voucher	O	Voucher	voucher.png	voucher

Para transformarmos esse cadastro em uma origem de dados, vimos que temos que gerar uma view a partir dele dentro do banco de dados. Podemos fazer isso pelo seguinte comando:

```
DROP VIEW IF EXISTS view_conecdata_formas;

CREATE VIEW
  view_conecdata_formas
AS SELECT
  fpg_pk AS id_interno,
  fpg_c_forma AS nome_forma,
  fpg_c_id_conecdata AS id_externo,
  1 AS id_loja,
  1 AS forma_ativa
FROM
  formas_pgto
WHERE
  fpg_c_id_externo IS NOT NULL;
```

Os valores destacados em verde, podem/devem ser substituídos por:

- Nomes de tabelas suas.
- Nomes de campos seus.
- Valores que substituam colunas/campos requeridos mas que não constem nas tabelas (devem ter os mesmos tipos deles). Nesse caso, esse valor será utilizado repetidamente em todas as linhas dessa coluna na tabela.

Comparando os campos presentes na tabela e os nescessários na composição da view, vamos perceber o seguinte:

Campo View	Campo DB	Situação
id_interno	fpg_pk	**Encontrado**
nome_forma	fpg_c_forma	**Encontrado**
-	fpg_e_tipo	Campo extra
-	fpg_c_legenda	Campo extra
-	fpg_c_img	Campo extra
id_externo	fpg_c_id_conecdata	**Encontrado**
id_loja	-	**NÃO ENCONTRADO**
forma_ativa	-	**NÃO ENCONTRADO**

Analisando a tabela e a view, veremos que encontramos campos na tabela que correspondem a um ID da view, campos que não correspondem a nenhum ID da view e finalmente campos da view que não tem correspondente na tabela. Veja como

proceder em cada um desses casos para gerar a view corretamente:

Situação	Tipo de ocorrência	Ação na view
Encontrado	IDS da view ENCONTRADOS no cadastro.	Indicar o **nome do campo** correspondente na tabela.
NÃO ENCONTRADO	IDS da view NÃO ENCONTRADOS no cadastro.	Se o campo for obrigatório, indicar um **valor fixo** do **mesmo tipo** do campo inexistente, se for opcional, não o inclua na view e seu valor default será utilizado automaticamente.
Campo extra	Campos no cadastro que não são necessários na view (campos extras).	Nenhuma.

> A documentação específica de cada origem, indica quais campos estão disponíveis nela, quais deles são opcionais e quais são obrigatórios. Campos obrigatórios, caso não indicados fazem a execução do script falhar, os opcionais que não forem indicados terão seus valores padrão (também indicado em sua documentação) utilizados.

Uma vez que gerada a view, temos que indicar seu tipo de conexão (**db** pois nossa conexão é MySql) e seu nome (**view_conecdata_formas**) na seguinte parte do arquivo `PASTA_SCRIPT/src/app/config/origens/config-formas-pgto.ts`:

```
export const CONFIG_FORMAS = {
  tipo: 'db',
  nomeView: 'view_conecdata_formas'
}
```

Lendo de um arquivo CSV

Vamos assumir que você já tenha gravado o arquivo seguinte arquivo CSV na pasta `c:/conecdata_csvs`.

Arquivo exemplo `c:/conecdata_csvs/formas-pgto.csv`

```
id_interno;nome_forma;id_externo;id_loja;forma_ativa
1;Dinheiro;dinheiro;1;1
2;Debito - Elo;debito_elo;1;1
3;Debito - Maestro;debito_maestro;1;1
4;Debito - Redeshop;debito_redeshop;1;1
5;Debito - Visa Electron;debito_visa_electron;1;1
6;Credito - American express;credito_amex;1;1
7;Credito - Diners;credito_diners;1;1
8;Credito - Elo;credito_elo;1;1
9;Credito - Hipercard;credito_hipercard;1;1
10;Credito - Mastercard;credito_mastercard;1;1
11;Credito - Policard;credito_policard;1;1
12;Credito - ValeCard;credito_valecard;1;1
13;Credito - Visa;credito_visa;1;1
14;Cheque;cheque;1;1
15;Alelo - Alimentacao;alelo_alimentacao;1;1
16;Alelo - Refeicao;alelo_refeicao;1;1
17;Policard - Alimentacao;policard_alimentacao;1;1
18;Policard - Refeicao;policard_refeicao;1;1
19;Sodexo - Refeicao;sodexo_refeicao;1;1
20;Ticket Rest. Eletronico;ticket_rest_eletronico;1;1
21;ValeCard - Alimentacao;valecard_alimentacao;1;1
22;ValeCard - Refeicao;valecard_refeicao;1;1
```

```
23;Visa - Vale;visa_vale;1;1
24;Voucher;voucher;1;1
```

Basta então configurar a seguinte parte do arquivo `PASTA_SCRIPT/src/app/config/origens/config-formas-pgto.ts`:

arquivo `PASTA_CONECSYNC/src/app/config/config-formas-pgto.ts`

```
export const CONFIG_FORMAS = {
  tipo: 'csv',
  nomeView: 'conecdata_formas.csv'
}
```

> Os arquivos indicados com o tipo csv, serão buscados na pasta origem de arquivos csvs configurada no arquivo **config.ts**.

Configurando arquivos de projetos

Projetos são containers para uma lista de lojas para as quais se deseja enviar os dados lidos das origens de dados.

Tokens de lojas

Obviamente é necessário evitar que qualquer um escreva livremente em qualquer loja da plataforma, é ai entram os **tokens de lojas**, que podem ser considerados chaves de acesso encriptadas que devem ser indicadas a cada chamada das APIs.

`PASTA_SCRIPT/src/app/config/projetos/mercadeiro.ts`

```
export const CONFIG_MERCADEIRO = {
  lojas: [
    {
      id: ID_LOJA_NO_SEU_CADASTRO,
      token: TOKEN_MERCADEIRO_LOJA_CORRESPONDENTE
    },
    {
      id: ID_OUTRA_LOJA_NO_SEU_CADASTRO,
      token: TOKEN_MERCADEIRO_LOJA_CORRESPONDENTE
    },
    ...
  ]
}
```

> Em caso de suspeita de utilização indevida de um token de loja, a integradora ou o lojista podem solicitar à Conecdata a **revogação** do token atual. Ao ser revogado, o token deixa de ser válido e todas suas referências deixam de funcionar e devem ser substituídas pelo novo token. O novo token pode ser acessado pelo módulo lojistas apenas por integradoras e agentes locais da Conecdata.

> Quando uma API é chamada sem um token de loja ou com um token inválido, é retornado um erro de **Acesso negado**. Já foi citado anteriormente que tokens de lojas são diferentes entre o modo produção e sandbox, e a configuração do modo do script com um tipo (no arquivo config.ts) e a indicação de outro nessa lista gerará erros quando o script usá-los para chamar as apis.

9 - Diversidades

Arquivos de log

> A cada execução do script, novas linhas são adicionadas no final dos arquivos de log. Eles podem ser apagados sem risco de causar problemas ao funcionamento do script.

PASTA_SCRIPT/ok.log

Arquivo `PASTA_SCRIPT/ok.log`

```
2021-04-10 16:14:18 INTEGRAÇÃO DE PRODUTOS ENCONTRADA. IGNORANDO INTEGRAÇÃO DE ESTOQUE:
config/origens/config-estoque.ts: tipo
2021-04-10 16:14:18 Verificando integração ESTOQUE.
2021-04-10 16:14:18 Verificando integração PRODUTOS_PROMOCOES.
2021-04-10 16:14:18 Encontrado: view_conecdata_produtos_promocoes
2021-04-10 16:14:18 Buscando produtos do DB.
2021-04-10 16:14:18 {"estoque":{"total":0,"sincronizados":0}}
2021-04-10 16:14:18 Verificando promoções.
2021-04-10 16:14:18 Encontrado: view_conecdata_promocoes
2021-04-10 16:14:18 Buscando promoções do DB.
2021-04-10 16:14:18 3 promoções encontrada(s).
2021-04-10 16:14:18 Sincronizando promoções.
2021-04-10 16:14:18 {"estoque":{"total":0,"sincronizados":0}}
2021-04-10 16:14:18 Verificando integração FORMAS PGTO.
2021-04-10 16:14:18 Encontrado: view_conecdata_formas
2021-04-10 16:14:18 Buscando formas pgto do DB.
2021-04-10 16:14:18 24 formas(s) pgto encontrada(s).
2021-04-10 16:14:18 Sincronizando formas pgto.
2021-04-10 16:14:37 {"formas":{"total":24,"sincronizados":24}}
2021-04-10 16:14:37 {"produtos":{"total":71,"sincronizados":0},"departamentos":
{"total":12,"sincronizados":0},"subdepartamentos":{"total":2,"sincronizados":0},"formas":
{"total":24,"sincronizados":24},"produtosPromocoes":
{"total":0,"sincronizados":0},"promocoes":{"total":3,"sincronizados":0},"estoque":
{"total":0,"sincronizados":0}}
```

> Dependendo da configuração da propriedade { verbose: BOOLEAN } no arquivo `config.ts` essas informações podem também ser exibidas ou não no terminal durante a execução do script.

PASTA_SCRIPT/errors.log

Arquivo `PASTA_SCRIPT/errors.log`

```
2021-04-10 16:14:15 Produto 21389: 500 - {"error":"geral","errors":{"geral":"Produto não
consta no banco de produtos."}}
2021-04-10 16:14:16 Produto 21390: 500 - {"error":"geral","errors":{"geral":"Produto não
consta no banco de produtos."}}
2021-04-10 16:14:17 Produto 21702: 500 - {"error":"geral","errors":{"geral":"Produto não
consta no banco de produtos."}}
2021-04-10 16:14:17 Produto 21703: 500 - {"error":"geral","errors":{"geral":"Produto não
consta no banco de produtos."}}
2021-04-10 16:14:18 Produto 21954: 500 - {"error":"geral","errors":{"geral":"Produto não
consta no banco de produtos."}}
2021-04-10 16:14:18 Produto 21955: 500 - {"error":"geral","errors":{"geral":"Produto não
consta no banco de produtos."}}
```

Hashes

Para evitar chamadas desnecessárias às apis, cada vez que o script consegue acioná-la com sucesso, um **hash** do objeto enviado é gravado. Toda vez que uma informação é lida de uma origem, um hash do novo objeto gerado a partir de seus dados

é comparado com seu hash anterior, um novo chamado à api só é acionado caso eles sejam diferentes. Excluindo esse registro (hash) de um registro, você força uma nova chamada incondicional à api.

Esses hashes, são armazenados dentro da pasta `PASTA_SCRIPT/lojas/`, separados em pastas com o **id de cada loja**, em arquivos (com a extensão.NeDB) com o nome de cada **origem de dados**.

exemplo `PASTA_SCRIPT/lojas/1/formas-pgto.NeDB`


```
{"id":1,"hash":"a775bd17b121bfec174aa449ff4b8cf625a091a6","_id":"2rMzVsKS8QwEsAAk"}
{"id":2,"hash":"f974a2c10dff627fb5d23d934410d9b0a5a3784a","_id":"ggQ9dl2wkBE0DVzl"}
{"id":3,"hash":"9436614fb8db85c8dbc54a4100c5006ecb8db33e","_id":"79KOyl8sSxiy4OOx"}
{"id":4,"hash":"d583c1ec1ca5970ee2d973a69a788a429b52cfe0","_id":"qp3Jo0rIlQJl0Fxr"}
{"id":5,"hash":"d2cdc34e8b5e2d2afc7db8c57c98407b2c87365b","_id":"Smm6SNoStZsdKai5"}
{"id":6,"hash":"417e446511f94336c648145279be382f2a13be4f","_id":"7WdLQZ587DKJvF4o"}
{"id":7,"hash":"39cc59c3811d968ec655100ff4bb173965b33318","_id":"bGuFwokQNsYAIX0R"}
{"id":8,"hash":"bf90c116195b5c038c2e31a98804ff66eaeb95cc","_id":"7yU4PYRjiK3KzVlk"}
{"id":9,"hash":"062843239ca8fda12714854f55176d97a7adf74e","_id":"7q0Iqi4G71yNpUz9"}
{"id":10,"hash":"646eeb6f51ba35374b1c961f1bbb5e8116a634df","_id":"tdV0XQzgGe2EqHK8"}
{"id":11,"hash":"3b696a80423bceea6cf1f03fac66563352ab1c83","_id":"VhRdHXWzkKXFWNGp"}
{"id":12,"hash":"8aa4d5b1bd7f0a4274583b64852495bf8e38d732","_id":"mE9J77Va9VlaK2XX"}
{"id":13,"hash":"f2facc8ea416804405ac479c871f6124e9d9f53b","_id":"r0rVm4CLvaHi2ulZ"}
{"id":14,"hash":"658d8891e605db9144dec236c6658384b58cd1c7","_id":"rkaU5ljK9JWx5k66"}
{"id":15,"hash":"ecf2c1fb0a6594033d977f435d5bea70a07dfccf","_id":"bYm7h5fLoCJzUDCI"}
{"id":16,"hash":"390d47c9a9416cce46751220e5fdb0281a452dcd","_id":"xM0DjTvxnyiGSrXr"}
{"id":17,"hash":"32b941bb258707635af634c09a48f0c5c1c64137","_id":"sWz06s8ytlP1Dsdt"}
{"id":18,"hash":"aa4c5fb011cf612f65b5f6bee28edcb3359ae90b","_id":"m05Dmx4jfotiyQn8"}
{"id":19,"hash":"0bd0ea387e494977cc3796543d0b4f69fedcf9a0","_id":"f9PCLvaOg1JqpOul"}
{"id":20,"hash":"3c032cb4c87b9be104c97fd1e13703609fc78de9","_id":"ocGX6Ekqe8zG2dFv"}
{"id":21,"hash":"8967eb3a26dfeb2299631d91dd1558bd75d9ff51","_id":"0AtmzbDgvw31y3f9"}
{"id":22,"hash":"5da63e8dfadaa4246df9093c9309140167b6e80b","_id":"vHrPTwE7PlbqSCv0"}
{"id":23,"hash":"9c9aa3a48bb80710777faaaf495732b0b63fd4e0","_id":"oXE6Aj79uiVmHL6u"}
{"id":24,"hash":"dbc099fbecadca5ab475dafac2c73a979e6d1add","_id":"vIXZFwLzxiv8vuz2"}
```


> Para forçar novas chamadas à api independente de seu conteúdo atual, você pode apagar hashes em diversos níveis:

Excluindo	Apis que serão acionadas incondicionalmente na próxima execução do script
Linhas dentro de um arquivo de hash	De registros correspondentes a essas linhas.
Um arquivo de hash	De registros correspondentes a essa origem.
Uma pasta de arquivos de hash	De registros de todas origens dessa loja.
A pasta lojas	De registros de todas origens de todas lojas.

> Excluir linhas/arquivos/pastas dentro da (ou a própria) pasta **lojas** não geram erros de execução do script, apenas forçam o envio das informações correspondentes aos itens excluídos.

Resetando origens

O resultado da execução do script pode ser acompanhado no próprio site do projeto, ou em sua loja dentro de módulo lojistas, entretanto nenhum desses locais permite que você elimine as informações recebidas via integração.

As informações correspondentes à uma origem podem ser excluídas acionando o endpoint **reset** de cada api (consulte as documentações específicas para maiores detalhes).

Tags para reset das origens

Origem	Tag origem	Resultado

Origem	Tag origem	Resultado
Estoque	estoque	Todos produtos da loja tem seu estoque regularizado.
Formas pagamento	formas-pgto	Formas de pagamento da loja são excluídas.
Produtos	produtos	Departamentos, subdepartamentos e produtos da loja são excluídos.
Promoções	promocoes	Promoções da loja são excluídas.

> Apenas origens do modo sandbox podem ser resetadas, em hipótese nenhuma dados em lote do modo produção podem ser excluídos por meio de chamadas à api (diretas ou indiretas via o ConecSync).