

 tryber / **sd-010-a-inventory-report** Privategenerated from [betrybe/sd-0x-inventory-report](#)

☆ 0 stars 🍴 0 forks

☆ Star

👁 Watch ▾

Code

Issues

Pull requests 30

Actions

Projects

Wiki

Security

Insights

🔗 master ▾

...

**vbuxbaum** ...

12 days ago

[View code](#)

Termos de acordos

Ao iniciar este projeto, você concorda com as diretrizes do Código de Ética e Conduta e do Manual da Pessoa Estudante da Trybe.

Boas vindas ao repositório do projeto de Relatório de Estoque!

Você já usa o GitHub diariamente para desenvolver os exercícios, certo? Agora, para desenvolver os projetos, você deverá seguir as instruções a seguir. Fique atento a cada passo e, se tiver qualquer dúvida, nos envie por *Slack*! #vqv 🚀

Aqui você vai encontrar os detalhes de como estruturar o desenvolvimento do seu projeto a partir desse repositório, utilizando uma branch específica e um *Pull Request* para colocar seus códigos.

SUMÁRIO

- [Habilidades](#)
- [Data de entrega](#)
- [Entregáveis](#)

- O que deverá ser desenvolvido
- Desenvolvimento e testes
- Dados
- Antes de começar a desenvolver

Lista de requisitos:

- Requisitos obrigatórios
 - 1 - Criar um método `generate` numa classe `SimpleReport` do módulo `inventory_report/reports/simple_report.py`. Esse método deverá receber dados numa lista contendo estruturas do tipo `dict` e deverá retornar uma string formatada como um relatório
 - 2 - Criar um método `generate` numa classe `CompleteReport` do módulo `inventory_report/reports/complete_report.py`. Esse método deverá receber dados numa lista contendo estruturas do tipo `dict` e deverá retornar uma string formatada como um relatório
 - 3 - Criar um método `import_data` dentro de uma classe `Inventory` do módulo `inventory_report/inventory/inventory.py`, capaz de ler um arquivo CSV o qual o caminho é passado como parâmetro
 - 4 - Criar um método `import_data` dentro de uma classe `Inventory` do módulo `inventory_report/inventory/inventory.py`, capaz de ler um arquivo JSON o qual o caminho é passado como parâmetro
 - 5 - Criar um método `import_data` dentro de uma classe `Inventory` do módulo `inventory_report/inventory/inventory.py`, capaz de ler um arquivo XML o qual o caminho é passado como parâmetro
 - 6 - Criar uma classe abstrata `Importer` no módulo `inventory_report/importer/importer.py`, que terá três classes herdeiras: `CsvImporter`, `JsonImporter` e `XmlImporter`, cada uma definida em seu respectivo módulo
 - 7 - Criar uma classe `InventoryIterator` no módulo `inventory_report/inventory/inventory_iterator.py`, que implementa a interface de um iterator (`Iterator`). A classe `InventoryRefactor` deve implementar o método `__iter__`, que retornará este iterator
- Requisitos bônus
 - 8 - Preencha a função `main` no módulo `inventory_report/main.py` que, ao receber pela linha de comando o caminho de um arquivo e o tipo de relatório, devolve o relatório correto
- Depois de terminar o desenvolvimento (opcional)
 - Revisando um Pull Request

- [Avisos finais](#)

Habilidades

Nesse projeto, você será capaz de:



- Você vai aprender sobre paradigmas de programação
- Conceitos de OO na prática, criando classes e instâncias
- Leitura e escrita de arquivos

Data de entrega

- Projeto obrigatório;
- Serão 2 dias de projeto.
- Data de entrega para avaliação final do projeto: 11/01/2022 - 14:00h .

Entregáveis

Para entregar o seu projeto você deverá criar um *Pull Request* neste repositório. Este *Pull Request* deverá conter, para aprovação em todos os requisitos, os arquivos que se encontram neste diretório. Os códigos serão desenvolvidos nos arquivos presentes no diretório `inventory_report`: `main.py`, `reports/simple_report.py`, `reports/complete_report.py`, `importer/importer.py`, `importer/json_importer.py`, `importer/xml_importer.py`, `importer/csv_importer.py`, `inventory/inventory.py`, `inventory/inventory_iterator.py`.

 **É importante que seus arquivos tenham exatamente estes nomes!** 

Você pode adicionar outros arquivos se julgar necessário. Qualquer dúvida, procure a gente no Slack!.

O que deverá ser desenvolvido

No projeto passado você implementou algumas funções que faziam leitura e escrita de arquivos `JSON` e `CSV`, correto? Neste projeto nós vamos fazer algo parecido, mas utilizando a Programação Orientada a Objetos! Você implementará um gerador de relatórios que recebe como entrada arquivos com dados de um estoque e gera, como saída, um relatório acerca destes dados.

Esses dados de estoque poderão ser obtidos de diversas fontes:

- Através da importação de um arquivo `CSV` ;
- Através da importação de um arquivo `JSON` ;
- Através da importação de um arquivo `XML` ;

Além disso, o relatório final deverá poder ser gerado em duas versões: simples e completa.

Como o projeto deve ser executável

Após implementar o requisito bônus, seu programa deverá ser executável **via linha de comando** com o comando `inventory_report <argumento1> <argumento2>` :

- O **<argumento 1>** deve receber o caminho de um arquivo a ser importado. O arquivo pode ser um `CSV`, `JSON` ou `XML`.
- O **<argumento 2>** pode receber duas strings: `simples` ou `completo`, cada uma gerando o respectivo relatório.

Desenvolvimento e testes

Este repositório já contém um *template* com a estrutura de diretórios e arquivos, tanto de código quanto de teste criados. Veja abaixo:

```
.
├── dev-requirements.txt
├── inventory_report
│   ├── data
│   │   ├── inventory.csv
│   │   ├── inventory.json
│   │   └── inventory.xml
│   ├── importer
│   │   ├── csv_importer.py
│   │   ├── importer.py
│   │   ├── json_importer.py
│   │   └── xml_importer.py
│   ├── inventory
│   │   ├── inventory_iterator.py
│   │   └── inventory.py
```

```
|   ├── main.py
|   └── reports
|       ├── complete_report.py
|       └── simple_report.py
├── pyproject.toml
├── README.md
├── requirements.txt
├── setup.cfg
├── setup.py
└── tests
    ├── __init__.py
    ├── test_complete_report.py
    ├── test_csv_importer.py
    ├── test_importer.py
    ├── test_inventory.py
    ├── test_json_importer.py
    ├── test_main.py
    ├── test_simple_report.py
    └── test_xml_importer.py
```

Apesar do projeto já possuir uma estrutura base, você quem deve implementar as classes. Novos arquivos podem ser criados conforme a necessidade.

Para executar os testes, lembre-se de primeiro **criar e ativar o ambiente virtual**, além de também instalar as dependências do projeto. Isso pode ser feito através dos comandos:

```
$ python3 -m venv .venv
```

```
$ source .venv/bin/activate
```

```
$ python3 -m pip install -r dev-requirements.txt
```

O arquivo `dev-requirements.txt` contém todas as dependências que serão utilizadas no projeto, ele está agindo como se fosse um `package.json` de um projeto `Node.js`. Com as dependências já instaladas, para executar os testes basta usar o comando:

```
$ python3 -m pytest
```

Se quiser saber mais sobre a instalação de dependências com `pip`, veja esse artigo: <https://medium.com/python-pandemonium/better-python-dependency-and-package-management-b5d8ea29dff1>

Para verificar se você está seguindo o guia de estilo do Python corretamente, você pode executá-lo com o seguinte comando:

```
$ python3 -m flake8
```

Dados

Arquivos de exemplo nos três formatos de importação estão disponíveis no diretório `data` dentro do diretório `inventory_report`.

Importação de arquivos CSV

Os arquivos **CSV** são separados por vírgula, como no exemplo abaixo:

```
id,nome_do_produto,nome_da_empresa,data_de_fabricacao,data_de_validade,numero_  
  
1,Nicotine Polacrilex,Target Corporation,2020-02-18,2022-09-17,CR25 1551  
4467 2549 4402 1,morbi ut odio cras mi pede malesuada in imperdiet et  
commodo vulputate justo in blandit  
2,fentanyl citrate,"Galena Biopharma, Inc.",2019-12-06,2022-12-25,FR29 5951  
7573 740Y XKGX 6CSG D20,bibendum morbi non quam nec dui luctus rutrum nulla  
tellus in  
3,NITROUS OXIDE,Keen Compressed Gas Co. Inc.,2019-12-22,2023-11-07,CZ09  
8588 0858 8435 9140 2695,ipsum dolor sit amet consectetur adipiscing elit  
proin risus praesent
```

Importação de arquivos JSON

Os arquivos JSON seguem o seguinte modelo:

```
[  
  {  
    "id":1,  
    "nome_do_produto":"CALENDULA OFFICINALIS FLOWERING TOP, GERANIUM MACULATUM  
    "nome_da_empresa":"Forces of Nature",  
    "data_de_fabricacao":"2020-07-04",  
    "data_de_validade":"2023-02-09",  
    "numero_de_serie":"FR48 2002 7680 97V4 W6FO LEBT 081",  
    "instrucoes_de_armazenamento":"in blandit ultrices enim lorem ipsum dolor  
  }  
]
```

Importação de arquivos XML

Os arquivos **XML** seguem o seguinte modelo:

```
<?xml version='1.0' encoding='UTF-8'?>  
<dataset>
```

```
<record>
  <id>1</id>
  <nome_do_produto>valsartan and hydrochlorothiazide</nome_do_produto>
  <nome_da_empresa>Lake Erie Medical & Surgical Supply DBA Quality Care
  <data_de_fabricacao>2019-10-27</data_de_fabricacao>
  <data_de_validade>2022-08-31</data_de_validade>
  <numero_de_serie>MT08 VVDN 2131 9NFL C1JG KTDV RS1L LOZ</numero_de_serie>
  <instrucoes_de_armazenamento>at lorem integer tincidunt ante vel ipsum pra
</record>
</dataset>
```

Instruções para entregar seu projeto:

⚠ Se você estiver fazendo esse projeto em duplas ⚠

É **essencial** que cada integrante da dupla tenha pelo menos 1 Push com o código completo do projeto. A nota de cada pessoa será computada individualmente, então o

☰ README.md



Exemplo.

- Estudante `Rafa` implementa 50% dos requisitos obrigatórios
- Em seguida `Gabs`, sua dupla, complementa com os 50% restantes (atingindo 100% dos obrigatórios)
 - Neste momento **SOMENTE** `Gabs` obteve aprovação.
- `Rafa` faz um Push em sua branch, com o código completo (atingindo 100% dos obrigatórios)
 - Neste momento `Rafa` também obtém a aprovação

Para mais detalhes de como deve ocorrer a dinâmica, consulte este material

ANTES DE COMEÇAR A DESENVOLVER:

1. Clone o repositório

- `git clone git@github.com:tryber/sd-010-a-inventory-report.git`
- Entre na pasta do repositório que você acabou de clonar:
 - `sd-010-a-inventory-report`

2. Crie o ambiente virtual para o projeto

- `python3 -m venv .venv && source .venv/bin/activate`

Nota: após terminar o trabalho, para desativar o ambiente virtual digite `deactivate`

3. Instale as dependências

- `python3 -m pip install -r dev-requirements.txt`

4. Crie uma branch a partir da branch `master`

- Verifique que você está na branch `master`
 - Exemplo: `git branch`
- Se não estiver, mude para a branch `master`
 - Exemplo: `git checkout master`
- Agora crie uma branch à qual você vai submeter os `commits` do seu projeto
 - Você deve criar uma branch no seguinte formato: `nome-github-nome-do-projeto`
 - Exemplo: `git checkout -b exemplo-inventory-report`

5. Adicione as mudanças ao `stage` do Git e faça um `commit`

- Verifique que as mudanças ainda não estão no `stage`
 - Exemplo: `git status` (deve aparecer listada a pasta *exemplo* em vermelho)
- Adicione o novo arquivo ao `stage` do Git
 - Exemplo:
 - `git add .` (adicionando todas as mudanças - *que estavam em vermelho* - ao `stage` do Git)
 - `git status` (deve aparecer listado o arquivo *exemplo/README.md* em verde)
- Faça o `commit` inicial
 - Exemplo:
 - `git commit -m 'iniciando o projeto inventory-report'` (fazendo o primeiro `commit`)
 - `git status` (deve aparecer uma mensagem tipo *nothing to commit*)

6. Adicione a sua branch com o novo `commit` ao repositório remoto

- Usando o exemplo anterior: `git push -u origin exemplo-project-name`

7. Crie um novo `Pull Request` (*PR*)

- Vá até a página de *Pull Requests* do [repositório no GitHub](#)
- Clique no botão verde *"New pull request"*
- Clique na caixa de seleção *"Compare"* e escolha a sua branch **com atenção**
- Clique no botão verde *"Create pull request"*
- Adicione uma descrição para o *Pull Request* e clique no botão verde *"Create pull request"*
- **Não se preocupe em preencher mais nada por enquanto!**

- Volte até a [página de Pull Requests do repositório](#) e confira que o seu *Pull Request* está criado

Requisitos obrigatórios:

1 - Criar um método `generate` numa classe `SimpleReport` do módulo `inventory_report/reports/simple_report.py`. Esse método deverá receber dados numa lista contendo estruturas do tipo `dict` e deverá retornar uma string formatada como um relatório.

- Deve ser possível executar o método `generate` sem instanciar um objeto de `SimpleReport`
- O método deve receber de parâmetro uma lista de dicionários no seguinte formato:

```
[
  {
    "id": 1,
    "nome_do_produto": "CALENDULA OFFICINALIS FLOWERING TOP, GERANIUM MACU",
    "nome_da_empresa": "Forces of Nature",
    "data_de_fabricacao": "2020-07-04",
    "data_de_validade": "2023-02-09",
    "numero_de_serie": "FR48 2002 7680 97V4 W6FO LEBT 081",
    "instrucoes_de_armazenamento": "in blandit ultrices enim lorem ipsum d"
  }
]
```

- O método deverá retornar uma saída com o seguinte formato:

```
Data de fabricação mais antiga: YYYY-MM-DD
Data de validade mais próxima: YYYY-MM-DD
Empresa com maior quantidade de produtos estocados: NOME DA EMPRESA
```

- A data de validade mais próxima, somente considera itens que ainda não venceram.

Dica: O módulo [datetime](#) vai te ajudar.

As seguintes verificações serão feitas:

- 1.1 - Será validado que é possível que o método `generate` da classe `SimpleReport` retorne a data de fabricação mais antiga
- 1.2 - Será validado que é possível que o método `generate` da classe `SimpleReport` retorne a validade mais próxima

- 1.3 - Será validado que é possível que o método `generate` da classe `SimpleReport` retorne a empresa com maior estoque
- 1.4 - Será validado que é possível que o método `generate` da classe `SimpleReport` retorne o relatório no formato correto

2 - Criar um método `generate` numa classe `CompleteReport` do módulo `inventory_report/reports/complete_report.py`. Esse método deverá receber dados numa lista contendo estruturas do tipo `dict` e deverá retornar uma string formatada como um relatório.

- A classe `CompleteReport` deve herdar o método (`generate`) da classe `SimpleReport`, de modo a especializar seu comportamento.
- O método deve receber de parâmetro uma lista de dicionários no seguinte formato:

```
[
  {
    "id": 1,
    "nome_do_produto": "CALENDULA OFFICINALIS FLOWERING TOP, GERANIUM MACU",
    "nome_da_empresa": "Forces of Nature",
    "data_de_fabricacao": "2020-07-04",
    "data_de_validade": "2023-02-09",
    "numero_de_serie": "FR48 2002 7680 97V4 W6FO LEBT 081",
    "instrucoes_de_armazenamento": "in blandit ultrices enim lorem ipsum d"
  }
]
```

- O método deverá retornar uma saída com o seguinte formato:

```
Data de fabricação mais antiga: YYYY-MM-DD
Data de validade mais próxima: YYYY-MM-DD
Empresa com maior quantidade de produtos estocados: NOME DA EMPRESA
```

```
Produtos estocados por empresa:
- Physicians Total Care, Inc.: QUANTIDADE
- Newton Laboratories, Inc.: QUANTIDADE
- Forces of Nature: QUANTIDADE
```

As seguintes verificações serão feitas:

- 2.1 - Será validado que é possível que o método `generate` da classe `CompleteReport` retorne a data de fabricação mais antiga
- 2.2 - Será validado que é possível que o método `generate` da classe `CompleteReport` retorne a validade de fabricação mais próxima

- 2.3 - Será validado que é possível que o método `generate` da classe `CompleteReport` retorne a empresa com maior estoque
- 2.4 - Será validado que é possível que o método `generate` da classe `CompleteReport` retorne a quantidade de produtos por empresa
- 2.5 - Será validado que é possível que o método `generate` da classe `CompleteReport` retorne o relatório no formato correto

3 - Criar um método `import_data` dentro de uma classe `Inventory` do módulo `inventory_report/inventory/inventory.py`, capaz de ler um arquivo CSV o qual o caminho é passado como parâmetro.


- O método, receberá como parâmetro o caminho para o arquivo CSV e o tipo de relatório a ser gerado (`"simples"` , `"completo"`). De acordo com os parâmetros recebidos, deve recuperar os dados do arquivo e chamar o método de gerar relatório correspondente à entrada passada. Ou seja, o método da classe `Inventory` deve chamar o método `generate` da classe que vai gerar o relatório (`SimpleReport` , `CompleteReport`).

As seguintes verificações serão feitas:

- 3.1 - Será validado que ao importar um arquivo csv simples será retornado com sucesso
- 3.2 - Será validado que ao importar um arquivo csv completo será retornado com sucesso

4 - Criar um método `import_data` dentro de uma classe `Inventory` do módulo `inventory_report/inventory/inventory.py`, capaz de ler um arquivo JSON o qual o caminho é passado como parâmetro.

- O método, receberá como parâmetro o caminho para o arquivo JSON e o tipo de relatório a ser gerado (`"simples"` , `"completo"`). De acordo com os parâmetros recebidos, deve recuperar os dados do arquivo e chamar o método de gerar relatório correspondente à entrada passada. Ou seja, o método da classe `Inventory` deve chamar o método `generate` da classe que vai gerar o relatório (`SimpleReport` , `CompleteReport`).

 Atente que estamos utilizando o mesmo método do requisito anterior.

As seguintes verificações serão feitas:

- 4.1 - Será validado que ao importar um arquivo json simples será retornado com sucesso

- 4.2 - Será validado que ao importar um arquivo json completo será retornado com sucesso

5 - Criar um método `import_data` dentro de uma classe `Inventory` do módulo `inventory_report/inventory/inventory.py`, capaz de ler um arquivo XML o qual o caminho é passado como parâmetro.

- O método, receberá como parâmetro o caminho para o arquivo XML e o tipo de relatório a ser gerado ("simples" , "completo"). De acordo com os parâmetros recebidos, deve recuperar os dados do arquivo e chamar o método de gerar relatório correspondente à entrada passada. Ou seja, o método da classe `Inventory` deve chamar o método `generate` da classe que vai gerar o relatório (`SimpleReport` , `CompleteReport`).



Atente que estamos utilizando o mesmo método do requisito anterior.

As seguintes verificações serão feitas:

- 5.1 - Será validado que ao importar um arquivo xml simples será retornado com sucesso
- 5.2 - Será validado que ao importar um arquivo xml completo será retornado com sucesso

6 - Criar uma classe abstrata `Importer` no módulo `inventory_report/importer/importer.py`, que terá três classes herdeiras: `CsvImporter`, `JsonImporter` e `XmlImporter`, cada uma definida em seu respectivo módulo.

- A classe abstrata deve definir a assinatura do método `import_data` a ser implementado por cada classe herdeira. Ela deve receber como parâmetro o nome do arquivo a ser importado.
- O método `import_data` definido por cada classe herdeira deve lançar uma exceção caso a extensão do arquivo passado por parâmetro seja inválida. Por exemplo, quando se passa um caminho de um arquivo extensão CSV para o `JsonImporter`.
- O método deverá ler os dados do arquivo passado e retorná-los estruturados em uma lista de dicionários conforme exemplo abaixo:

```
[
  {
    "id": 1,
    "nome_do_produto": "CALENDULA OFFICINALIS FLOWERING TOP, GERANIUM MACU",
    "nome_da_empresa": "Forces of Nature",
    "data_de_fabricacao": "2020-07-04",
    "data_de_validade": "2023-02-09",
    "numero_de_serie": "FR48 2002 7680 97V4 W6FO LEBT 081",
```

```
"instrucoes_de_armazenamento": "in blandit ultrices enim lorem ipsum d
}
]
```

As seguintes verificações serão feitas:

- 6.1 - Será validado que a classe CsvImporter está herdando a classe Importer
- 6.2 - Será validado que a classe JsonImporter está herdando a classe Importer
- 6.3 - Será validado que a classe XmlImporter está herdando a classe Importer
- 6.4 - Será validado que a classe CsvImporter está importando os dados para uma lista
- 6.5 - Será validado que a classe JsonImporter está importando os dados para uma lista
- 6.6 - Será validado que a classe XmlImporter está importando os dados para uma lista
- 6.7 - Será validado que ao enviar um arquivo com extensão incorreta para o CsvImporter irá gerar um erro
- 6.8 - Será validado que ao enviar um arquivo com extensão incorreta para o JsonImporter irá gerar um erro
- 6.9 - Será validado que ao enviar um arquivo com extensão incorreta para o XmlImporter irá gerar um erro

🧠 Estamos separando a lógica em várias classes (estratégias), preparando para aplicarmos o padrão de projeto **Strategy**. É uma solução para o caso em que uma classe possui muitas responsabilidades (propósitos).

7 - Criar uma classe InventoryIterator no módulo

`inventory_report/inventory/inventory_iterator.py`, que implementa a interface de um iterator (`Iterator`). A classe `InventoryRefactor` deve implementar o método `__iter__`, que retornará este iterador.

- A classe `Inventory` deverá ser refatorada (copiada) em outro arquivo chamado `inventory_report/inventory/inventory_refactor.py`. Nesse arquivo você irá refatorar a classe `Inventory` chamando-a de `InventoryRefactor`.
- A classe `InventoryRefactor` deve utilizar as classes definidas no requisito 6 para lidar com a lógica de importação, via **composição** no método `import_data`.
- A classe `InventoryRefactor` deve receber por seu construtor a classe que será utilizada para lidar com a lógica de importação e armazenar em um atributo chamado `importer`.

- As classes `InventoryIterator` e `InventoryRefactor` devem implementar corretamente a interface do padrão de projeto **Iterator**, de modo que seja possível iterar sobre os itens em estoque.
- Ao importar os dados, os mesmos devem ser armazenados na instância, em adição aos itens já presentes naquela instância. O atributo de `InventoryRefactor` que armazena esses dados deve se chamar `data`.
- Os atributos e os métodos devem ser públicos.

👉 Para testar manualmente você pode fazer:

```
iterator = iter(inventory)
first_item = next(iterator)
```

As seguintes verificações serão feitas:

- 7.1 - Será validado que a instancia de `InventoryRefactor` é iterável (`Iterable`)
- 7.2 - Será validado que é possível iterar o primeiro item da lista usando `csv`
- 7.3 - Será validado que é possível iterar o primeiro item da lista usando `json`
- 7.4 - Será validado que é possível iterar o primeiro item da lista usando `xml`
- 7.5 - Será validado que é possível receber duas fontes de dados sem sobreescrita
- 7.6 - Será validado que não é possível enviar arquivo inválido

Requisitos bônus:

8 - Preencha a função `main` no módulo `inventory_report/main.py` que, ao receber pela linha de comando o caminho de um arquivo e o tipo de relatório, devolve o relatório correto.

- Deverá ser usado a classe `InventoryRefactor` para recuperar os dados e gerar o relatório.
- Ao chamar o comando no formato abaixo pelo terminal, deve ser impresso na tela o devido relatório no formato da saída dos requisitos 1 e 2:

```
$ inventory_report <caminho_do_arquivo_input> <tipo_de_relatório>
```

- Caso a chamada tenha menos de três argumentos (o nome `inventory_report` é considerado o primeiro argumento), exiba a mensagem de erro "Verifique os argumentos" na `stderr`.

📌 A função `sys.argv` deve ser utilizada para receber a entrada de dados da pessoa usuária.

👉 Teste manual: dentro de um ambiente virtual onde seu projeto foi configurado, digite o comando `inventory_report parametro_1 parametro_2`, assim você conseguirá interagir com o menu.

As seguintes verificações serão feitas:

- 8.1 - Será validado se o menu importa um arquivo csv simples
- 8.2 - Será validado se o menu importa um arquivo csv completo
- 8.3 - Será validado se o menu importa um arquivo json simples
- 8.4 - Será validado se o menu importa um arquivo json completo
- 8.5 - Será validado se o menu importa um arquivo xml simples
- 8.6 - Será validado se o menu importa um arquivo xml completo
- 8.7 - Será validado se houverem argumentos faltantes será retornando um erro

Depois de terminar o desenvolvimento

Para sinalizar que o seu projeto está pronto para o "Code Review" dos seus colegas, faça o seguinte:

- Vá até a página **DO SEU Pull Request**, adicione a label de "code-review" e marque seus colegas:
 - No menu à direita, clique no link "**Labels**" e escolha a label **code-review**;
 - No menu à direita, clique no link "**Assignees**" e escolha **o seu usuário**;
 - No menu à direita, clique no link "**Reviewers**" e digite `students`, selecione o time `tryber/students-sd-010-a`.

Caso tenha alguma dúvida, [aqui tem um video explicativo](#).

Revisando um pull request

Use o conteúdo sobre [Code Review](#) para te ajudar a revisar os *Pull Requests*.

Avisos finais

Ao finalizar e submeter o projeto, não se esqueça de avaliar sua experiência preenchendo o formulário. Leva menos de 3 minutos!

Link: [FORMULÁRIO DE AVALIAÇÃO DE PROJETO](#)

O avaliador automático não necessariamente avalia seu projeto na ordem em que os requisitos aparecem no readme. Isso acontece para deixar o processo de avaliação mais rápido. Então, não se assuste se isso acontecer, ok?

Releases

No releases published

[Create a new release](#)

Packages

No packages published

[Publish your first package](#)

Contributors 2



jeanpsv Jean Paulo Silva Vasconcelos



vbuxbaum Vitor Buxbaum Orlandi

Languages

● **Python** 100.0%