ETL do Boadica - Manual do Usuário

Autores: Leonardo Izaú, Mariana Fortes, Diana Lobo

Instalação

O objetivo do programa é a partir de um banco de dados MySQL com tabelas de peças (CPU, GPU e Placa Mãe), consultar os preços de cada peça no Boadica e salvar no banco. Segue abaixo o passo a passo de instalação:

Passo 1: Criar uma conta no ParseHub, uma interface online que permite o scraping (extração) de dados em um website. Após criar a conta, o usuário deve importar o projeto do Boadica; está definido neste projeto como o ParseHub irá consultar os preços das peças. A Figura 1 mostra o projeto do Boadica, disponível no arquivo "*Projeto_Boadica.phj*".

Figura 1: Projeto Boadica

Uma consulta no ParseHub funciona a partir de *templates*, de modo que cada template corresponde a uma série de operações que serão executadas num site. No projeto do Boadica, há dois *templates*; *main_template* realiza comandos na página principal do Boadica, e preços_resultados trabalha sobre a página de resultados de uma peça. A extração de preços é feita da seguinte forma: uma string JSON de peças é carregada pelo projeto, no formato "{"pecas":[nomePeca1,nomePeca2, ... , nomePecaN]}". Para cada peça na lista, o programa insere o nome da peça na barra de pesquisa do Boadica e aperta o botão para navegar até a página de resultados. Em *preços_resultados*, caso a peça tenha sido encontrada no site, o programa retorna o nome da peça e seu preço. No final da consulta, o ParseHub retorna um JSON com os nomes de cada peça buscada e os preços associados. As Figuras 2 e 3 exibem os comandos feitos nos templates *main_template e precos_resultados* respectivamente.

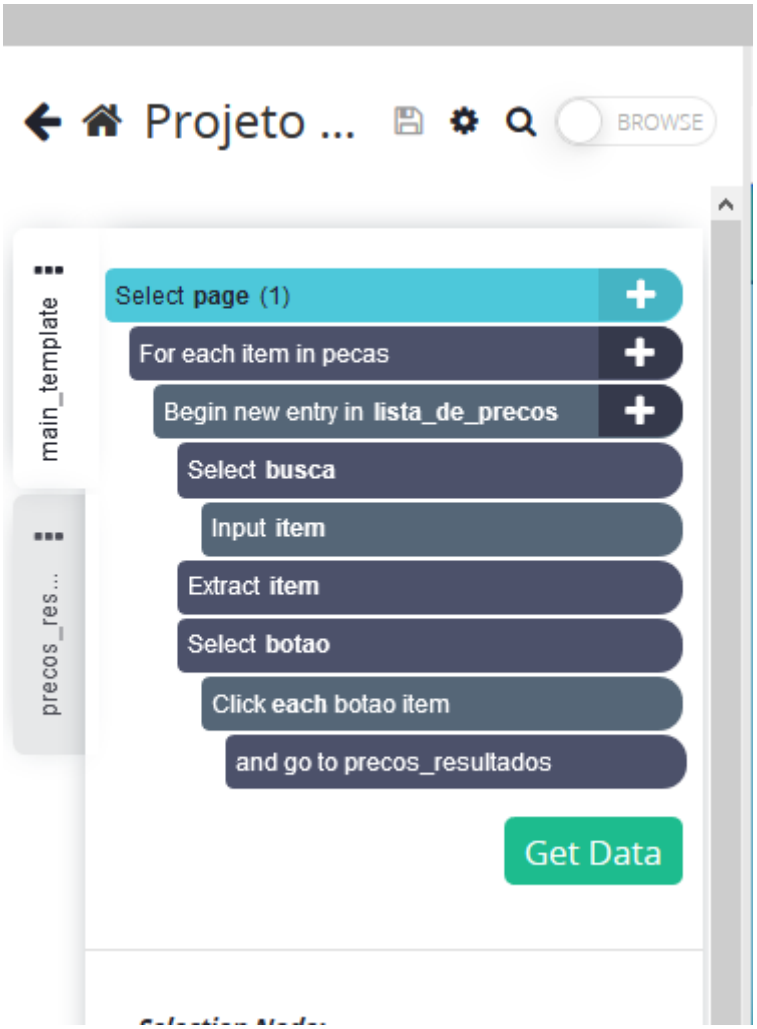

Figura 2: Template principal

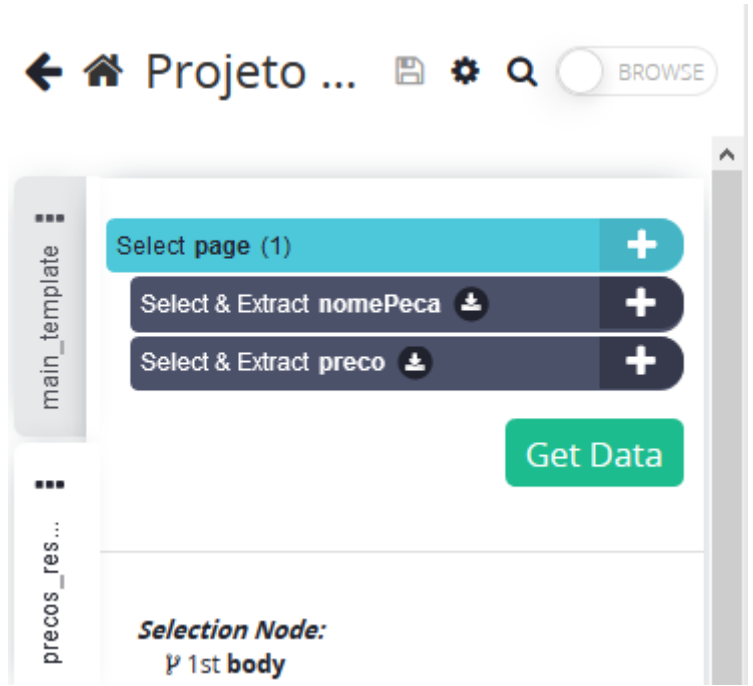

Figura 3: Template de resultados

Após verificar se o projeto ParseHub está configurado corretamente, verifique alguns dos parâmetros do projeto (na aba de configurações). Em particular, anote a chave da API e o token do projeto, pois serão necessários para executar o código Python no futuro. Também observe que o número máximo de páginas extraídas em uma consulta é 126 (1 *main_template* + *125 precos_resultados*).

Passo 2: Instalar os pacotes do Python 3 necessários para rodar o código (pode ser feita pelo pip). Os pacotes são:

- PyMySQL (interface do python para trabalhar com banco MySQL)
- Cryptography (complemento do PyMySQL para evitar problemas de criptografia do banco)
- Requests (necessário para fazer requisições GET e POST no protocolo HTTP)
- Numpy (utilizado para trabalhar com vetores)

Outros pacotes utilizados foram “csv” (para abrir arquivos CSV em leitura e/ou escrita), “json” (para transformar strings JSON em dicionários e vice-versa) e “time” (para temporizar a execução de certos trechos do código).

Passo 3: Instalar o MySQL na máquina, e criar o banco de dados das peças. As tabelas de cada peça podem ser criadas pelo arquivo “Criação_das_tabelas.sql”. Anote o seu nome de usuário, senha e nome do banco criado, os quais serão utilizados no código para abrir a conexão com o banco.

Execução

Para verificar se a consulta de preços está funcionando, primeiro execute o arquivo *TesteScraping.py*. O código deste arquivo cria uma tabela de consulta do tipo de peça desejado (CPU, GPU ou Placa Mãe), realiza consultas de preço no ParseHub com base nessa tabela e atualiza os preços no banco de dados. Considerando que todas as tabelas de cada peça têm 1000 entradas, foi necessário realizar 8 consultas (a cada 125 entradas) por tabela. É importante ressaltar que para o programa funcionar, os parâmetros do ParseHub e do MySQL devem ser inicializados.

Com os preços atualizados no banco, execute o arquivo *TesteConsultas.py* para verificar se os preços foram atualizados corretamente. Este arquivo retorna duas saídas; a primeira são os dados de uma peça (marca, modelo, preço), dado um tipo de peça, uma marca e um modelo; a segunda retorna todas as peças na tabela desejada em que o preço é não-nulo. Novamente, para o programa funcionar é preciso inserir no código os parâmetros do MySQL. Também foi criada uma versão

alternativa desse programa, *TesteConsultasMenu.py*, caso o usuário deseje fazer as consultas em modo interativo.