**Hackaton - Lucas Carralas e Pablo Sivila**

Web Scrapping/API - Report

Cases:

* **Omelete Web Scrapping**

A primeira coisa que fiemos, foi acessar o site de entretenimento Omelete, aonde tentamos a princípio acessar o RSS, que não estava disponível no momento. Como não houve condições de realizar dessa maneira, decidimos realizar o Web Scrapping buscando uma lista das últimas notícias gerais e por categoria.

*Passos:  
  
-Iniciamos aplicando o uso de classe pois compactuam bem com o código.   
-Em seguida, para cada categoria do site criamos um arquivo diferente, inclusive para a categoria geral que reúne todas.  
-Então pudemos observar que poderíamos reduzir a área de busca na sopa para não retornar lixo.  
-Utilizamos então uma URL base para todas as cinco categorias que buscávamos.  
-E finalmente rodamos um looping para todas as categorias.*

**- Rotten Tomatoes Web Scrapping**

Buscamos fazer um Web Scraping no ranking dos melhores filmes de todos os tempos segundo o ranking do site de avaliações Rotten Tomatoes (<https://www.rottentomatoes.com/top/bestofrt/>).  
  
*Passos:*

*-Como no outro Web Scrapping, iniciamos aplicando o uso de classe.  
-Em seguida, para cada categoria do site criamos um arquivo diferente, inclusive para a categoria geral que reúne todas.  
-Mesmo reduzindo a área de busca na sopa, ainda assim percebemos que a lista carregava lixo.  
-Por este motivo, limitamos o retorno aos elementos de índice 1 ao -20  
-Desta forma retornamos exatamente a lista de 100 filmes onde também procuramos deixar todas as informações numéricas, incluindo porcentagens, em variáveis apenas numéricas.  
-Por fim cortamos os 13 espaços em branco no começo do título do filme os últimos 7 caracteres referentes ao ano e colocamos o ano em uma coluna separada.*

* **Final Fantasy Web Scrapping**

Buscamos criar uma tabela mostrando as habilidades de cada um dos tipos de personagem para o jogo Final Fantas Tactics.  
  
*Passos:*  
  
*-Iniciamos aplicando o uso de classe pois compactuam bem com o código.   
- Como aqui precisávamos limpar as informações mais a fundo, preferimos criar uma lista antes de imprimir.  
-Também adicionamos a categoria na lista para termos o nome da personagem nos dados  
-Reduzimos a área de busca da sopa para evitar retorno de lixo.  
-Percebemos que para três categorias específicas havia um valor a mais e precisamos fazer um retorno diferente.  
-Em seguida utilizamos uma url base que era comum a todas as categorias que iríamos buscar.  
-Finalmente utilizamos um looping para rodar todas as categorias que queríamos, mas percebemos que especificamente para 5 categorias o link não seguia o padrão exato dos outros e precisamos fazer um retorno diferente por precaução, como eram mais de 30 acessos, colocamos um intervalo de 1 segundo para evitar quaisquer problemas.  
- Uma das linhas continha uma categoria com informações duplas pois, dependendo da condição que estava aplicada, o valor seria diferente. Assim, criamos uma linha nula logo após e preenchemos ela com as informações extras ao mesmo tempo que as removíamos da linha anterior.  
- Também classificamos a taxa de crescimento de atributos por um índice numérico indo de 1 a 5, no lugar de categorias que iam de "Very Low" a "Very High" removemos o símbolo de porcentagem dos dados de uma coluna para ela ser totalmente numérica.  
- Por fim imprimimos o arquivo*

* **Age of Empires API**

Também por meio do banco de APIs públicos, acessamos o do game “Age of Empires 2” na URL (<https://age-of-empires-2-api.herokuapp.com/docs/>), aonde procuramos reunir as informações de todas as civilizações presentes no dito jogo. Além do próprio nome da civilização, buscamos o tipo de exército, sua unidade exclusiva, tecnologia diferenciada e seu bônus team.  
  
*Passos:*  
  
*- Accessamos um api que retorna informações do jogo Age of Empires 2 e pedimos as informações de civilizações  
- Filtramos a parte do json que nos traria as informações de cada civilização.  
- Imprimimos as informações que julgamos relevantes.*

* **Asteroids API**

Asteroides com orbitas semi circulares ou aproximadamente circulares.  
  
*Passos:*  
  
*- Acessamos um api que retorna informações de asteroides pedimos as informações de 100 asteróides com órbita circular ou o mais próximo disso  
- Imprimimos todas as informações desses asteróides. precisamos de um looping para imprimir as chaves como nomes de coluna e de outro looping para imprimir as informações de cada asteróide*

**- Football API**

Acessamos a um banco de dados de API público disponível no GitHub (<https://www.football-data.org/index>) e buscamos criar uma tabela com o as informações de todos os jogadores que jogam na primeira divisão do campeonato brasileiro de 2019, por time. Além de seu respectivo nome, apresentaríamos sua posição e sua nacionalidade.

*Passos:  
-* *#acessamos um api que retorna informações sobre futebol e pedimos as informações de competições  
-* *#filtramos a parte do json que nos traria as informações de cada competição  
-* *#imprimimos os arquivos. precisamos especificar os casos em que o campeonato ainda não tinha nenhum vencedor e também o caso em que a região buscada não estava com nenhum campeonato disponível.  
- Na segunda parte pedimos as informações de todas as equipes do campeonato "Série A" no brasil. Era necessário fornecer uma chave de acesso para obter as informações aqui.  
-* *Imprimimos as informações que julgamos relevantes para cada time  
- Na terceira parte, por fim, pedimos todas as informações de todos os times do campeonato "Série A" no brasil,e selecionamos apenas as informações de cada jogador. junto a elas colocamos para qual time cada um jogava.  
-Para evitar que o site nos bloqueasse pois eram mais de 800 acessos contínuos colocamos um intervalo .de 5 segundos entre cada pedido (tentamos com valores menores que ainda nos prendiam).  
-* *Finalmente, imprimimos tudo.*