

**Relatório**

Processamento de Linguagens

**Alunos**

José Cosgrove 18826

André Cardoso 18848

**Professor**

Alberto Simões

**Licenciatura em Engenharia de Sistemas Informáticos**

Barcelos, 22 de novembro de 2020

Resumo

Foi proposto para trabalho prático desta Unidade Curricular a elaboração de um projeto em Python, onde puséssemos em prática conceitos adquiridos nas aulas. Utilizamos ferramentas como o Lexer do Ply (ply.lex) e o Regex (re) para auxiliar na criação de código. Utilizamos ainda HTML para fazer o output dos dados.

Índice

[1. TAP – Test Anything Protocol 4](#_Toc57042143)

[2. Expressões Regulares 5](#_Toc57042144)

[3. PLY e REGEX 6](#_Toc57042145)

[4. Implementação HTML 7](#_Toc57042146)

[5. Conclusão 8](#_Toc57042147)

# TAP – Test Anything Protocol

Este é um protocolo de impressão de resultados de testes executados a unidades de software.

Segue um formato constante de identificação de testes: estado, posição e descrição. Os subtestes estão formatados do mesmo modo. Para além de testes e subtestes, pode encontrar-se uma linha identificadora do total de testes ou sequência de subtestes – esta encontra-se no fim ou início da sequência. Também é possível encontrar-se comentários, linhas de texto não relevantes ao processamento.

Todo o seu conteúdo de um ficheiro TAP tem de ser reconhecido para uma boa execução do programa criado, pois caso alguma expressão não seja reconhecida, todo o processo falha.

# Expressões Regulares

De modo a ler os ficheiros e reconhecer devidamente os elementos inseridos, fazemos uso de expressões regulares, a ferramenta Regex.

Uma expressão regular é uma sequência de caracteres que definem um padrão, neste caso, o padrão com que os ficheiros TAP são escritos.

* **Total**

Toda a sequência de testes ou subtestes tem, ou no início ou no fim desta, uma indicação do seu total, por exemplo:

*1..20*

Visto que toda a sequência lógica de testes e subtestes deverá iniciar pelo primeiro ou seja o número ‘1’, podemos considerar o primeiro valor constante, assim como os pontos entre os dois números. Quanto ao número referente ao total, podemos assumir que pelo menos um número entre ‘1’ e ‘9’ tem de ser e pode ou não ser um número com mais de dois dígitos.

Sabendo isto podemos deduzir que a expressão regular que define o total de casos da sequência é:

“1\.\.[1-9]\d\*”

* + ‘1’

Caracter ‘1’

* + ‘\.’

Dado que o caracter ‘.’ representa um caracter especial no regex, este para ser interpretado literalmente tem de ser acompanhado por ‘\’.

* + ‘[1-9]’

Quando colocámos caracteres entre ‘[’ e ‘]’ indicamos que deve ser escolhido apenas um deles. No entanto, se colocarmos o ‘–’ entre dois caracteres, é identificada uma sequência ASCII e assim é escolhido um valor dentro dessa sequência, neste caso entre ‘1’ e ‘9’.

* + \d\*

Este caracter representa qualquer dígito, é o equivalente a “[0-9]”. Quando acompanhado por ‘\*’, este indica que pode repetir o reconhecimento de qualquer dígito zero ou infinitas vezes. O caracter ‘\*’ é aplicável a qualquer expressão válida.

* **Estado**

Varia apenas entre “ok” e “not ok” e indica o sucesso da avaliação. Resume-se à seguinte expressão:

*“ok|not ok”*

Dado que os únicos valores possíveis para este campo são conhecidos e não variáveis, aplicamos uma expressão definida. O caracter ‘|’ representa a operação lógica OR, permitindo reconhecer uma das duas expressões intercaladas.

# PLY e REGEX

# Implementação HTML

Uma vez que já temos acesso aos dados e sua estrutura temos de os apresentar.

Poderíamos apresentar os dados na consola, mas isso não iria ter a apresentação e interpretação que desejávamos. Uma boa solução para este problema era a utilização dos dados através de uma página HTML.

Esta página é criada utilizando a framework bootstrap que produz um aspeto visual mais apelativo e ainda várias classes e formatações já definidas.

É inicialmente criado um ficheiro através dos métodos de ficheiros de python e é escrito nesse ficheiro as tags do html e são inseridos através do python, no ficheiro HTML, os dados de leitura.

A página começa por apresentar os diversos ficheiros que são lidos. Cada título de ficheiro apresenta-se a verde ou vermelho, para representar se os testes correram todos bem ou se houve erros.

Cada ficheiro tem um número de testes, a cada teste podem corresponder subtestes, a cada subteste podem corresponder outros subtestes e assim sucessivamente.

Estes testes e subtestes são também apresentados com a indentação correspondente, com cor verde, caso esteja tudo ok, e a vermelho se não estiver.

# Conclusão

Na elaboração deste projeto, pudemos aprofundar mais os nossos conhecimentos no que toca a trabalhar em python (era uma linguagem nova até ao início do semestre) e tentar pôr em prática todas as boas práticas que esta linguagem implica.

Tivemos também de nos colocar à prova em termos criativos e lógicos, pois existiam diferentes formas de pensar e elaborar este projeto.

No que toca ao trabalho em equipa, conseguimos fazer uma correta gestão e partição do trabalho e fomo-nos ajudando um ao outro sempre que algum de nós se encontrava com mais dificuldades.

Achamos o trabalho algo acessível, uma vez que sempre fomos a todas as aulas, o que ajudou bastante no desenvolvimento do projeto, pois ao longo das aulas fomos fazendo casos práticos que envolveram conceitos que utilizamos neste projeto.

Para finalizar o relatório, apenas acrescento a ideia de que achamos este tipo de trabalhos muito importantes no nosso percurso académico, pois achamos que eles nos vão preparando cada vez mais para sairmos deste curso engenheiros capazes e capacitados.