Relatório do Trabalho Prático 2

Processamento de Linguagens

ESI-2021/2022

Realizado por:

Marco Henriques nº16961

Índice

Introdução

Foi proposta a realização de um trabalho para a disciplina de Processamneto de Linguagens do curso de Engenharia de Sistemas Informáticos. A ferramenta desenvolvida pretende ser um reconhecedor léxico e de gramática de comandos SQL.

Estes comandos serão de Manuseamento de Tabelas, Execução de Queries, Procedimentos, entre outros.

Abordagem

Em primeiro lugar foi escrita a gramática dos comandos, sendo que a mesma foi sendo alterada ao longo do projeto consoante a necessidade de adotar novos comandos ou novas estratégias para o funcionamento do reconhecedor.

Nas figuras está apresentado o ficheiro parser.out, que é gerado automaticamente com as regras de reconhecimento.

O axioma será o A representado seguido de todas as regras que constituem o projeto.

```
S' -> A
A -> procedurecommand
A -> multicommand
procedurecommand -> function VAR staticnames multicommand staticnames ;
multicommand -> command
multicommand -> multicommand command
command -> function commandcreate ;
command -> function commanddropshow ;
command -> function commandalter ;
command -> function commandcomment ;
command -> function commandsave ;
command -> function commandselect ;
function -> CREATE
function -> DROP
function -> ALTER
function -> COMMENT
function -> SHOW
function -> SAVE
function -> SELECT
function -> PROCEDURE
commandcreate -> staticnames VAR ( argument )
commandcreate -> staticnames VAR staticnames ( argument2 )
commanddropshow -> staticnames VAR
commandalter -> staticnames VAR staticnames ( argument )
commandcomment -> staticnames staticnames VAR staticnames comment
commandsave -> staticnames VAR staticnames COMMASTRING
commandselect -> * staticnames VAR
argument -> subargument
argument -> <empty>
```

```
argument2 -> staticnames VAR staticnames VAR staticnames VAR > NUMBER
argument2 -> staticnames VAR staticnames VAR staticnames VAR < NUMBER
argument2 -> staticnames VAR staticnames VAR staticnames VAR = NUMBER
argument2 -> staticnames VAR staticnames VAR staticnames VAR > = NUMBER
argument2 -> staticnames VAR staticnames VAR staticnames VAR < = NUMBER
argument2 -> staticnames VAR staticnames VAR staticnames VAR < > NUMBER
subargument -> VAR VAR ( NUMBER )
staticnames -> TABLE
staticnames -> FROM
staticnames -> SELECT
staticnames -> WHERE
staticnames -> AS
staticnames -> ADD
staticnames -> IS
staticnames -> ON
staticnames -> MODIFY
staticnames -> DO
staticnames -> END
comment -> COMMASTRING
```

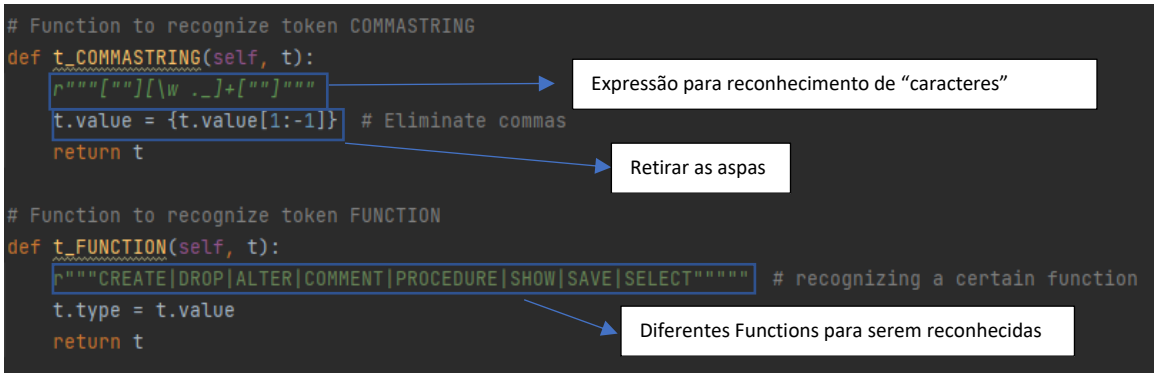
Ficheiro logic_lexer.py

Em seguida foi criado o ficheiro logic_lexer.py que teria os tokens para serem reconhecidos assim como os caracteres como mostra a imagem seguinte.

```
# Creating tokens for recognition
tokens = ("CREATE", "TABLE", "NUMBER", "VAR", "FROM", "SELECT", "WHERE", "AS", "ADD", "DROP", "ALTER", "STATICNAMES",
          "IS", "COMMENT", "COMMASTRING", "ON", "PROCEDURE", "DO", "END", "SHOW", "SAVE", "MODIFY")

# Literals to recognize single characters
literals = ['(', ')', ';', ',', '*', '>', '<', '>', '=', '<', '=', '<', '>', '=']
```

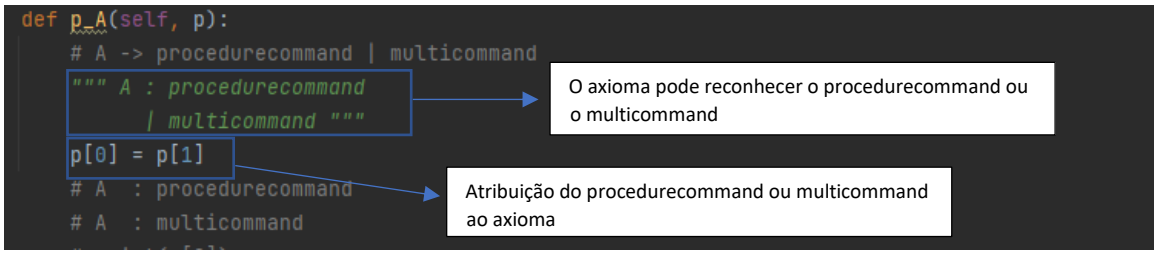
Cada token terá de ter uma expressão para o seu reconhecimento

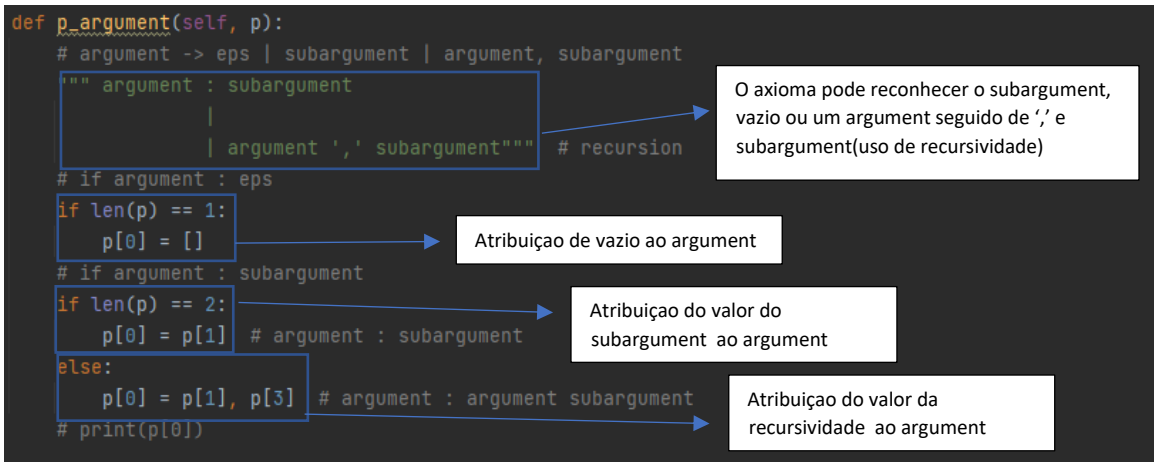


Ficheiro logic_gramar.py

Para a criação da gramática foi necessário criar um ficheiro logic_grammar.py onde estariam as regras de sintaxe para serem reconhecidas pelo projeto.

As seguintes figuras reprsentam algumas das regras desenvolvidas.





Ficheiro logic.py

Por ultimo foi criado um ficheiro logic.py onde foi criado um ciclo infinito para reconhecimento de comandos escritos no terminal pelo utilizador, sendo esses comandos escritos num ficheiro .txt criado para esse efeito.