

Universidade do Minho
Departamento de Informática



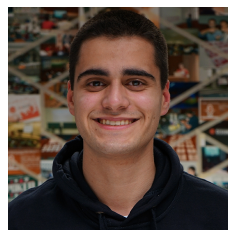
Processamento de Linguagens

Grupo 24

22 de maio, 2022



Beatriz
Rodrigues
(a93230)



Francisco Neves
(a93202)



Guilherme
Fernandes
(a93216)

Índice

1	Introdução	3
2	Analisador Léxico	4
3	Gramática	5
4	<i>Parser</i> e <i>AST</i>	6
5	Dicionários	7
5.1	JSON <i>parser</i>	7
5.1.1	Analisador Léxico	7
5.1.2	Gramática	7
5.1.3	<i>Parser</i>	8
6	<i>Pipes</i>	9
7	<i>Templates</i> Implementados	10
7.1	HTML	10
7.2	Markdown	11
7.3	Asciiidoc	12
7.4	Plain Text	13
7.5	Bash Tree	14
7.6	Manual de Receitas	15
7.7	Pokédex	16
7.8	Carta de Motivação	18
8	Extras Estéticos	20
8.1	Tratamento de Erros	20
8.2	<i>Syntax Highlighting</i>	21
9	Conclusões e Trabalho Futuro	22
10	Bibliografia	23

1. Introdução

koal@ foi desenvolvido no contexto da unidade curricular de Processamento de Linguagens. Retirando inspiração da linguagem de *templates* **Pandoc**, foi desenvolvida uma ferramenta que permite, de forma semelhante, recorrer a um dicionário (*JSON* ou *YAML*) e a um *template* seguindo a *syntax* definida para dar origem a um resultado final no formato correspondente ao *template* utilizado.

No entanto, ao longo do processo, foi decidido que seria interessante adicionar novas funcionalidades vistas como potencialmente úteis, a descrever em mais detalhe no presente relatório.

Para além disso, como possível valorização, foi implementado ainda um *parser* de ficheiros *JSON*, a utilizar em conjunto com a linguagem desenvolvida.

2. Analisador Léxico

Em primeiro lugar, foram definidos os *tokens* desta linguagem:

- **Texto** → Todos os símbolos contidos entre aspas
- **VAR** → Símbolos iniciados por `_`, que representam possíveis entradas do dicionário, ou seja, as variáveis da *template*;
- **TMPVAR** → Símbolos iniciados por `#`, que podem ser definidas no *template*, representando assim variáveis temporárias;
- **NEWLINE** → Símbolo representativo do *newline* que permite efetuar o controlo de linhas;
- **ALIASNAME** → Representam funções o nome de funções que podem ser posteriormente utilizadas;
- **IF** → Palavra reservada (`if`) que indica uma condição do tipo *if*;
- **ELIF** → Palavra reservada (`elif`) que indica uma condição do tipo *elif*;
- **ELSE** → Palavra reservada (`else`) que indica uma condição do tipo *else*;
- **FOR** → Palavra reservada (`for`) que indica um ciclo *for*;
- **ALIAS** → Palavra (`alias`) que indica a definição de um *alias*;
- **INCLUDE** → Palavra reservada (`include`) que indica a inclusão dos *alias* de um outro ficheiro indicado em seguida;
- **PIPE** → Caractere `'/'` que indica que se deverá seguir um *pipe* do programa.

Foram ainda definidos os seguintes caracteres como *literals* da linguagem em questão: `'{'`, `'}'`, `':'`, `'('`, `')'` e `','`.

Além disto, foi ainda definida a regra `t_ignore` para ignorar caracteres não pretendidos e a regra `t_ignore_comment` de forma a ignorar os comentários (linhas iniciadas por `'//'`).

Por fim, de forma a efetuar um correto controlo de *newlines* foi criada uma *flag* denominada *ignore_newline* que, tal como o nome indica, tem como propósito indicar em que circunstâncias é pretendido que se ignore um *newline* e vice-versa.

3. Gramática

Tendo isto, passou-se então para a definição da gramática necessária para a linguagem de *templates* pretendida.

Assim, foi decidido que a linguagem (*lang*) seria definida por *statements*, sendo que *statements* seria uma lista de *statement* com caso de paragem em vazio. Por sua vez, um *statement* poderia ser as diversas possibilidades oferecidas pela linguagem, desde expressões condicionais, até ciclos *for*, variáveis, *aliases* ou texto corrido.

Foi então definida a seguinte gramática:

```
Rule 0      S' -> lang
Rule 1      lang -> statements
Rule 2      statements -> statements statement
Rule 3      statements -> <empty>
Rule 4      statement -> variable
Rule 5      statement -> NEWLINE
Rule 6      statement -> if elifs else
Rule 7      if -> IF variable block
Rule 8      elifs -> elifs elif
Rule 9      elif -> ELIF variable block
Rule 10     elifs -> <empty>
Rule 11     else -> ELSE block
Rule 12     else -> <empty>
Rule 13     statement -> FOR TMPVAR : variable block
Rule 14     statement -> ALIAS ALIASNAME tmpvars block
Rule 15     tmpvars -> tmpvars TMPVAR
Rule 16     tmpvars -> <empty>
Rule 17     statement -> ALIASNAME ( args )
Rule 18     args -> variables
Rule 19     args -> <empty>
Rule 20     variables -> variables , variable
Rule 21     variables -> variable
Rule 22     statement -> INCLUDE TEXT NEWLINE
Rule 23     block -> { statements }
Rule 24     variable -> VAR
Rule 25     variable -> TMPVAR
Rule 26     variable -> TEXT
Rule 27     variable -> variable FIELD
Rule 28     variable -> variable PIPE
```

4. *Parser* e *AST*

Tendo em conta a biblioteca utilizada para efetuar o *parsing* da linguagem (módulo `yacc` do `ply`), foi observado que se estava perante uma estratégia de *parsing bottom-up*, pelo que, foi prioritária a utilização de recursividade à esquerda.

Além disto, reparou-se ainda que, devido a esta estratégia de *parsing*, por diversas ocasiões como em avaliações de expressões condicionais ou de ciclos era, primeiramente efetuada a verificação da expressão no corpo da funcionalidade, sendo que, apenas após isso se validava a condição da expressão, algo que, não era de todo o pretendido.

Assim, foi decidido que seria benéfica a criação de uma *AST* (árvore de sintaxe abstrata) que permite inverter o processo de avaliação acima referido e confere uma maior abstração à avaliação da linguagem desenvolvida. Para isto, foi criado o módulo *koala_ast* que possui diversas classes abstratas que são utilizadas nas classes representativas dos diversos tipos da linguagem criada, quando adequadas.

5. Dicionários

Como referido anteriormente, a linguagem desenvolvida é capaz de interpretar dicionários do tipo **JSON** ou **YAML**. Sendo que, de forma a valorizar o trabalho efetuado, para os dicionários do tipo **JSON** foi também criada uma ferramenta de *parsing* que permite a sua transformação em dicionários de **Python**. Por outro lado, para a utilização de dicionários do tipo **YAML** foi utilizada a biblioteca **yaml** recorrendo ao seu método `load`, utilizando o *Loader* `yaml.FullLoader`.

5.1 JSON *parser*

De forma a efetuar o *parsing* de dicionários **JSON** para dicionários de *Python*, foi criado um *parser* para este tipo de ficheiros.

5.1.1 Analisador Léxico

Inicialmente, foram decididos os *tokens* que o *parser* deveria ser capaz de reconhecer, foram eles:

- **STRING** → Texto colocado entre aspas;
- **TRUE** → Palavra reservada *true*;
- **FALSE** → Palavra reservada *false*;
- **NULL** → Palavra reservada *null*;
- **NUMBER** Número real que poderá encontrar-se em notação científica ou não.

Além destes, foram ainda utilizados os seguintes *literals*: '{', '}', ',', ':', '[' e ']'.

5.1.2 Gramática

Tendo isto, foi então criada a seguinte gramática:

Rule 0	S' -> json
Rule 1	json -> element
Rule 2	value -> object
Rule 3	value -> array
Rule 4	value -> STRING
Rule 5	value -> NUMBER
Rule 6	value -> TRUE
Rule 7	value -> FALSE

```
Rule 8      value -> NULL
Rule 9      object -> { }
Rule 10     object -> { members }
Rule 11     members -> member
Rule 12     members -> members , member
Rule 13     member -> STRING : element
Rule 14     array -> <empty>
Rule 15     array -> [ elements ]
Rule 16     elements -> element
Rule 17     elements -> elements , element
Rule 18     element -> value
```

5.1.3 *Parser*

Tendo em conta o objetivo deste trabalho prático, o *parser* foi definido tendo em mente apenas a definição de um dicionário em `Python` passível de utilização pelo `koal@`. Assim, recorrendo ao módulo `yacc` do `ply` e dando prioridade à recursividade à esquerda na definição da gramática, foi definido o *parser* para os ficheiros pretendidos.

6. *Pipes*

A implementação de *pipes* surge de forma a permitir que o utilizador possa efetuar diferentes operações consoante a posição relativa de um elemento numa dada lista ou, até mesmo, converter *Strings* para maiúsculas ou minúsculas e inverter a ordem de uma lista ou de uma *String*. Com a sua utilização, permitimos que, por exemplo, todos os elementos da lista sejam terminados com o caractere ',' exceto o último, conforme uma típica enumeração.

Assim, com isto em mente, foram implementados os seguintes *pipes*:

- ***First***: Utiliza-se com `/first` e permite obter o primeiro elemento da lista;
- ***Last***: Utiliza-se com `/last` e permite obter o último elemento da lista;
- ***Head***: Utiliza-se com `/head` e permite obter todos os elementos de uma lista exceto o último;
- ***Tail***: Utiliza-se com `/tail` e permite obter todos os elementos de uma lista exceto o primeiro;
- ***Upper***: Utiliza-se com `/upper` e permite colocar uma *String* em maiúsculas;
- ***Lower***: Utiliza-se com `/lower` e permite colocar uma *String* em minúsculas;
- ***Reverse***: Utiliza-se com `/reverse` e permite inverter a ordem de uma *String* ou de uma lista.

7. *Templates* Implementados

De forma a serem efetuados alguns testes na linguagem criada, foram desenvolvidos diversos *templates* de linguagens *markup* conhecidas ou outras utilidades consideradas interessantes para a linguagem em questão. Entre eles, podem-se considerar os seguintes:

7.1 HTML

```
include "examples/html/htmlhelp.koa"
"<!DOCTYPE html>"
<head>
  <meta charset="utf-8" />
  <meta name="generator" content="koala" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0, user-scalable=yes" />

if @authors {
  for #author : @authors {
    " <meta name="author" content="\"#author \" />"
  }
}

if @date {
  " <meta name="dcterms.date" content="\"@date \" />"
}

if @keywords {
  for #keyword : @keywords {
    " <meta name="keywords" content="\"#keyword \" />"
  }
}

if @description {
  " <meta name="description" content="\"@description \" />"
}

if @title {
  "<title>@title</title>"
}
for #css : @css {
  "<link rel="stylesheet" href="\"#css \" />"
}
```

Figura 7.1: Excerto do *template* de *HTML*

7.2 Markdown

```
examples > markdown > mdhelp.koa
1  alias extendTOC #item {
2    if #item.head {
3      #item.head
4      if #item.sub {
5        for #subitem : #item.sub {
6          extendTOC(#subitem)
7        }
8      }
9    } else {
10     #item
11   }
12 }
13 alias extendSections #evalsec {
14   "# "#evalsec.title
15   #evalsec.content
16
17   if #evalsec.subsections {
18     for #subsection : #evalsec.subsections {
19       extendSections(#subsection)
20     }
21   }
22 }
```

Figura 7.2: *Alias* definidos para o *template* de *Markdown*

```
examples > markdown > markdown.koa
1  include "examples/markdown/mdhelp.koa"
2  if @titleblock {
3    @titleblock
4  }
5
6  if @tableofcontents {
7    "# Table of Contents"
8
9    for #elem : @tableofcontents {
10      extendTOC(#elem)
11    }
12 }
13
14 for #section : @sections {
15   extendSections(#section)
16 }
17
18 if @bibliography {
19   "# References"
20
21   for #bibentry : @bibliography {
22     "* " #bibentry.source ". Last checked on " #bibentry.lastchecked
23   }
24 }
```

Figura 7.3: *Template* de *Markdown*

7.3 Asciiidoc

```
examples > asciidoc > asciidoc.koa
1 include "examples/asciidoc/getters.koa"
2 if @titleblock{
3   "= " @title
4   if @author{
5     for #aut : @author/head {AuthorName(#aut) " " AuthorEmail(#aut); "} AuthorName(@author/last) AuthorEmail(@author/last)
6   }
7   if @date {
8     @date
9   }
10  if @keywords{
11    ":keywords: " for #keyword : @keywords/head {#keyword"; "} @keywords/last
12  }
13  if @lang{
14    ":lang: " @lang
15  }
16  if @toc{
17    ":toc:"
18  }
19 }
20
21 }
22 if @abstract{
23   "[abstract]"
24   "== Abstract"
25   @abstract
26 }
27
28 for #item : @headerincludes{
29   #item
30 }
31
32 for #item : @includebefore{
33   #item
34 }
35
36 @body
37 for #item : @includeafter{
38   #item
39 }
40 }
```

Figura 7.4: *Template de Asciiidoc*

7.4 Plain Text

```
if @titleblock{
    @titleblock
}
for #item : @headerincludes{
    #item
}
for #item : @includebefore{
    #item
}
if @tableofcontents{
    @tableofcontents
}
@body
for #item : @includeafter{
    #item
}
```

Figura 7.5: *Template de Plain text*

7.5 Bash Tree

```
alias cd #dir {  
    "cd " #dir  
}  
alias mkdir #dir {  
    "mkdir " #dir  
}  
alias cp #source #dest {  
    "cp " #source " " #dest  
}  
alias touch #file {  
    "touch " #file  
}
```

Figura 7.6: Comandos *bash*

```
examples > bash_tree > tree.koa  
1  include "examples/bash_tree/bash.koa"  
2  alias traverse #dir {  
3      mkdir(#dir.name)  
4      cd(#dir.name)  
5      for #file : #dir.files {  
6          if #file.path {  
7              cp(#file.path, ".")  
8          } else {  
9              touch(#file.name)  
10             }  
11         }  
12         for #subdir : #dir.subdirs {  
13             traverse(#subdir)  
14             cd("../")  
15         }  
16     }  
17     traverse(@dir)
```

Figura 7.7: *Template de Bash Tree*

7.6 Manual de Receitas

```
1 <!DOCTYPE html>
2 <table cellpadding="20">
3   <thead>
4     <tr>
5       <th></th>
6       <th>Nome</th>
7       <th>Dificuldade</th>
8       <th>Tempo de Preparação</th>
9       <th>Porções</th>
10      <th>Vegetariano</th>
11    </tr>
12  </thead>
13
14  <tbody>
15    <tr>
16      <td></td>
17      <td><img src=""#recipe.image" width="150" height="150"></td>
18      <td><a href=""#recipe.site" >"#recipe.name"</a></td>
19      <td><img src=""#recipe.image" width="150" height="150"></td>
20      <td><img src=""#recipe.image" width="150" height="150"></td>
21      <td><img src=""#recipe.image" width="150" height="150"></td>
22      <td><img src=""#recipe.image" width="150" height="150"></td>
23    </tr>
24  </tbody>
25 </table>
```

Figura 7.8: *Template* do manual de receitas

```
recipes:
- name: Bacalhau com Natas
  image: https://static.clubedaanamariabraga.com.br/wp-content/uploads/2019/11/bacalhau-com-natas-1024x576.jpg
  site: https://anamariabraga.globo.com/receita/bacalhau-com-natas/
  difficulty: Fácil
  veggie: Não
  doses: 4
  time: 1h00min
- name: Arroz de Pato
  image: https://ti.rg.ltmcdn.com/pt/posts/9/1/2/arroz-de-pato-a-portuguesa-219-600.jpg
  site: https://www.tudoreceitas.com/receita-de-arroz-de-pato-a-portuguesa-219.html
  difficulty: Média
  veggie: Não
  doses: 6
  time: 1h30min
```

Figura 7.9: Fragmento do dicionário exemplificativo





	Nome	Dificuldade	Tempo de Preparação	Porções	Vegetariano
	Bacalhau com Natas	Fácil	1h00min	4	Não
	Arroz de Pato	Média	1h30min	6	Não
	Bacalhau à Gomes de Sá	Difícil	3h00min	4	Não
	Cozido à Portuguesa	Fácil	1h41min	8	Não

Figura 7.10: Fragmento do resultado do manual de receitas

7.7 Pokédex

```
examples > pokédex > pokédex.koa
1  include "examples/std.koa"
2  "<!DOCTYPE html>"
3  <table cellpadding="20">
4      <thead>
5          <tr>
6              <th></th>
7              <th>Number</th>
8              <th>Name</th>
9              <th>Type</th>
10             <th>Evolutions</th>
11         </tr>
12     </thead>"
13
14     <tbody>"
15     for #poke : @pokemon {
16         <tr>"
17         <td ALIGN="CENTER"></td>"
18         <td ALIGN="CENTER">#poke.num</td>"
19         <td ALIGN="CENTER">#poke.name</td>"
20         <td ALIGN="CENTER">join(#poke.type, "<br>")</td>"
21         <td ALIGN="CENTER">"
22             if #poke.next_evolution {
23                 for #evo : #poke.next_evolution/head {
24                     #evo.name
25                     <br>"
26                 }
27                 #poke.next_evolution/last.name
28             </td>"
29         }
30     </tr>"
31 }
32 </tbody>"
33 </table>"
```

Figura 7.11: *Template da Pokédex*

```
examples > std.koa
1  alias join #list #delim { for #elem : #list/head {#elem #delim} #list/last }
```

Figura 7.12: Ficheiro *stdalias*


```
{
  "pokemon": [{
    "id": 1,
    "num": "001",
    "name": "Bulbasaur",
    "img": "http://www.serebii.net/pokemongo/pokemon/001.png",
    "type": [
      "Grass",
      "Poison"
    ],
    "height": "0.71 m",
    "weight": "6.9 kg",
    "candy": "Bulbasaur Candy",
    "candy_count": 25,
    "egg": "2 km",
    "spawn_chance": 0.69,
    "avg_spawns": 69,
    "spawn_time": "20:00",
    "multipliers": [1.58],
    "weaknesses": [
      "Fire",
      "Ice",
      "Flying",
      "Psychic"
    ],
    "next_evolution": [{
      "num": "002",
      "name": "Ivysaur"
    }, {
      "num": "003",
      "name": "Venusaur"
    }
  ]
}]
```

Figura 7.13: Fragmento do dicionário da *Pokédex*







	Number	Name	Type	Evolutions
	001	Bulbasaur	Grass Poison	Ivysaur Venusaur
	002	Ivysaur	Grass Poison	Venusaur
	003	Venusaur	Grass Poison	
	004	Charmander	Fire	Charmeleon Charizard
	005	Charmeleon	Fire	Charizard
	006	Charizard	Fire Flying	

Figura 7.14: Fragmento do resultado da *Pokédex*

7.8 Carta de Motivação

```
\\newcommand{\\myname}{@name}
\\newcommand{\\mytitle}{if @title {@title} else {"Applicant"}}
\\newcommand{\\myemail}{@email}
\\newcommand{\\mypage}{if @page {@page} elif @git {@git} else { @linkedin }}
\\newcommand{\\myphone}{@phone}
\\newcommand{\\mylocation}{@location}
if @recipient{
  "\\newcommand{\\recipient}{@recipient}"
}
\\newcommand{\\greeting}{if @greeting{@greeting", "} else {"Greetings, "}}
\\newcommand{\\closer}{if @closer{@closer", "} else{"Best wishes and regards, "}}
\\newcommand{\\company}{@company}
if @company_street{
  "\\newcommand{\\street}{@company_street}"
}
if @company_city {
  "\\newcommand{\\city}{@company_city}"
}
if @company_state {
  "\\newcommand{\\state}{@company_state}"
}
if @company_zip {
  "\\newcommand{\\zip}{@company_zip}"
}
"\\begin{document}
\\AddToShipoutPictureBG{
\\color{gr}
\\AtPageUpperLeft{\\rule[-1.3in]{\\paperwidth}{1.3in}}
}

\\begin{center}
{\\fontsize{28}{0}\\selectfont\\scshape \\myname}

\\href{mailto:\\myemail}{\\faEnvelope\\enspace \\myemail}\\hfill"
if @page{
  "\\href{\\mypage}{\\faSafari\\enspace My Website}\\hfill"
}
elif @git{
  "\\href{\\mypage}{\\faGit*\\enspace Git Page}\\hfill"
}
else{
  "\\href{\\mypage}{\\faLinkedin\\enspace LinkedIn}\\hfill"
}
\\href{tel:\\myphone}{\\faPhone\\enspace \\myphone}\\hfill
\\faMapMarker\\enspace \\mylocation
\\end{center}
```

Figura 7.15: Excerto do *Template* de Cartas de Motivação

```

name: Arlindo Barnabé
age: 20
job: Student
title: Software Engineering Student
company: Sports Interactive
company_state: London
company_city: Stratford
company_country: England
company_sector: Games Programming
university: Universidade do Minho
course: Software Engineering
year: 3rd
area: Software Engineering
recipient: Miles Jacobson
greeting: Dear
page: https://beasrodrigues24.github.io/yaPP/
email: arlindobarney2209@gmail.com
phone: +351 911111111
location: Portugal
application_for: Summer Internship

```

Figura 7.16: Dicionário exemplificativo para carta de motivação

ARLINDO BARNABÉ

✉ arlindobarney2209@gmail.com 🌐 Personal Page 📞 +351 911111111 📍 Portugal

May 22, 2022

Miles Jacobson

Sports Interactive
Stratford, London

Dear, Miles Jacobson

My name is Arlindo Barnabé, I'm 20 years old and I'm a 3rd year student of Software Engineering at Universidade do Minho. Currently, I'm looking for a Summer Internship and I really think your company is the perfect opportunity for it. My academic formation is, currently, preparing me for a professional career on the Software Engineering area, although, in my opinion, the learning process is never completed and that's why I always demonstrate a great desire to learn new things and improve the ones that I already know. This way, I think that by your side I could improve my knowledge and discover even more about this subject, since Sports Interactive is a reference in this sector.

At the moment, I don't have any professional experience at this area, although I can't think about any company who could be a better place to start than Sports Interactive.

Having in consideration the main areas of your company and my knowledge, I must say that I have a special interest on the area of the Games Programming but, as I already mentioned earlier, I have a great desire to learn more, so, any other area would also be of my interest.

I appreciate the dispended attention and make myself available for any desired information.

Best wishes and regards,
Arlindo Barnabé
Software Engineering Student

Figura 7.17: Ficheiro LaTeX gerado

8. Extras Estéticos

8.1 Tratamento de Erros

Considerando a perspectiva do utilizador, foi considerada relevante a implementação de um tratamento de erros elegante e apelativo. Assim, foi desenvolvida uma classe denominada *PrettyPrint* que recebendo uma *String*, possui métodos que permitem a exibição de erros ou *warnings* ao utilizador de forma colorida.

```
[tgvp@tgvp-mkx recipes]$ python3 ../../src/koala.py -dt yaml recipes.koa recipes.yaml recipes.html  
Error: recipes.koa:15: Syntax error. Token 'recipe' not expected.
```

Figura 8.1: Exemplo de Mensagem de Erro

```
[tgvp@tgvp-mkx recipes]$ python3 ../../src/koala.py -dt yaml recipes.koa recipes.yaml recipes.html  
Warning: recipes.koa:16: Invalid token: ';'. Skipping...
```

Figura 8.2: Exemplo de Mensagem de *Warning*

8.2 *Syntax Highlighting*

De forma a permitir uma utilização mais simples e prazerosa do programa, foi ainda criada uma extensão para *Microsoft Visual Studio Code* que permite adicionar *syntax highlight* a ficheiros com a extensão `.koa`.

Para isso, o utilizador deverá dirigir-se ao *marketplace* do *VS Code* e a partir de lá efetuar a instalação de uma extensão externa através do ficheiro `.vsix` fornecido.

Podem ser vistos exemplos desta utilidade através das imagens de *templates* de exemplo anteriormente fornecidas.

9. Conclusões e Trabalho Futuro

Este projeto permitiu o reforço de conhecimentos obtidos na unidade curricular de Processamento de Linguagens, permitindo ainda a compreensão do poder de uma linguagem de *templates* e a verificação do vasto leque de utilizações possíveis. Considera-se que foram implementadas bastantes funcionalidades criativas e convenientes, que tornam esta ferramenta interessante e útil.

É possível afirmar que foi produzido um resultado positivo, que explorou as potencialidades que este programa poderia ter para além das mecânicas inspiradas pelo **Pandoc**.

Por fim, considera-se ainda que o **koal@** é um programa passível de evolução podendo oferecer melhores e mais vantagens aos seus utilizadores como, por exemplo, os diversos *pipes* que o **Pandoc** oferece que ainda não foram implementados ou a inserção de novos *templates* pré-definidos que permitiriam que o utilizador pudesse apenas alterar o conteúdo dos dicionários.

10. Bibliografia

1. **Documentação do ply** . Consultado pela última vez a 22 de maio de 2022.
2. **Documentação do Python** . Consultado pela última vez a 22 de maio de 2022.
3. **Documentação do JSON** . Consultado pela última vez a 22 de maio de 2022.
4. **Documentação do YAML** . Consultado pela última vez a 22 de maio de 2022.
5. **Documentação do Pandoc** . Consultado pela última vez a 22 de maio de 2022.
6. **Documentação do AsciiDoc** . Consultado pela última vez a 22 de maio de 2022.
7. **W3Schools** . Consultado pela última vez a 22 de maio de 2022.
8. ***Dataset* de Pokémon GO** . Consultado pela última vez a 22 de maio de 2022.
9. **Template da Carta de Recomendação em LaTeX** . Consultado pela última vez a 22 de maio de 2022.