

# SPLN - TP2

## Serviço remoto de filtro-linha-comando genérico

Tiago Silva a93277 and Gonçalo Afonso a93178

Universidade do Minho, Braga, Portugal

### 1 Introdução

O presente relatório segue o trabalho realizado para o segundo trabalho prático da unidade curricular de Scripting e Processamento de Linguagens Naturais. Neste segundo trabalho, foi proposto pelos docentes a concretização de uma solução para um dos temas num conjunto por eles eleito, tendo o nosso grupo, dentro destes, escolhido a realização do tema 4: Serviço remoto para um serviço filtro-linha-de-comando genérico.

Este problema revolve na criação de um serviço remoto, que disponibilize diferentes ferramentas para realizar pedidos de uso destas. Um dos principais requisitos neste trabalho é a necessidade de criar um serviço cuja acessibilidade seja garantida mesmo para pessoas não versadas com o uso 'puro' das ferramentas (ex.: em linha de comando), assim como a possibilidade de facilmente criar serviços com conjuntos de ferramentas diferentes de formar mais fácil automática.

O serviço que criamos é então um programa em python, instalável, que recebendo um ficheiro de configuração de um servidor, modifica um servidor modelo feito em Node.js com uso de *express-generator*[2], criando as suas rotas, instalando dependências e com opção para o iniciar automaticamente, assim como expor para um url público. A este programa demos o nome de *Tool Server Generator* [1]

De igual modo, para a criação do servidor, temos como dependência a instalação do *Node.js* numa versão igual ou superior a 10.5, para suportar o módulo de *worker\_threads*

### 2 Solução

A solução desenvolvida envolveu dois pontos principais, o programa em Python para a criação automática do servidor em Node.js e o desenvolvimento do próprio modelo do servidor.

#### 2.1 Programa

Começando pelo programa em python, este é instalável realizando:

```
pip install tool_server_generator
```

, sendo que o seu uso segue a chamada do comando com a indicação obrigatório de um ficheiro de texto ou json, que deve ter a configuração do servidor:

```
ts <ficheiro de configuração>
```

Outros modos de uso envolvem a utilização da opção *'-s'* que indica a intenção de iniciar automaticamente o servidor assim que ele estiver pronto e *'-ng'* que indica a intenção de, se a flag de iniciar o servidor estiver acionado, expor o servidor utilizando *ngrok*[3]. Esta última opção, assume no entanto que o *ngrok* já se encontra disponível no sistema, assim como a autenticação dele realizada.

## 2.2 Configurao do servidor

O ficheiro de configurao  o que ir definir desde a porta e rota do servidor, como as ferramentas que ele possui, at  paleta de cores das pginas web.

Este ficheiro pode ser interpretado de dois modos, em formato de texto, ou em formato JSON, sendo que a identificao de um ficheiro JSON  feita automaticamente (sem uso de opes), atravs da extenso do ficheiro.

No caso do ficheiro JSON, naturalmente ele ter de seguir os campos especficos que explicaremos em seguida. Quanto ao ficheiro de texto, este seguir uma sintaxe por ns construída com a utilizao do mdulo *lark* e do *Interpreter* deste, tendo portanto de seguir uma estrutura mais rigorosa no que toca  ordem dos campos.

### Campos de Configurao Servidor

- *Nome* : nome do servidor.
- *Diretoria*: diretoria onde o servidor vai ser criado.
- *Porta*: porta do servidor.
- *Rota*: rota base do website. Opcional, '/' por defeito.
- *Trabalhadores*: nmero de trabalhadores disponveis para processar pedidos. Opcional, 1 por defeito.

### Ferramentas

- *Famlia* : famlia da ferramenta. Permite agrupar ferramentas da mesma famlia, por exemplo que tenham o mesmo comando base mas opes diferentes. Determina a rota da ferramenta.
- *Ttulo* : nome da ferramenta. Determina a rota da ferramenta.
- *Descrio* : descrio da ferramenta.
- *Comando*: comando que correr a ferramenta, com os inputs variveis (pelo utilizador) delimitados utilizando a palavra exclusiva 'INPUT<number>'.
- *Inputs*: para cada ferramenta, os seus inputs variveis tm de ser explicitamente configurados, caso contrrio sero tratados como texto simples no comando. Opcional.

### Para cada Input de uma ferramenta

- *Nome*: nome do input. Opcional.
- *Descrio*: descrio do input. Opcional.
- *Tipo*: tipo do input (STR|NUM|FILE).

### Visuais (Opcional)

- *Favicon*: icon do servidor.
- *Colors*: paleta de 6 cores do servidor. 2 cores para o *background* (primria e secundria), 2 cores para o texto (primria e secundria), 1 cor para as *labels*, e uma para as bordas.

### Exemplo de ficheiro de configurao em texto

```
* Servidor
- Nome: Servidor
- Diretoria: "teste"
- Porta: 15213
* Ferramentas
--
- Famlia:Grep
- Ttulo: "Grep"
```

- Descrição: "grep searches for PATTERNS in each FILE.  
 PATTERNS is one or more patterns separated by newline characters, and grep prints each line that matches a pattern.  
 Typically PATTERNS should be quoted when grep is used in a shell command."  
 -Comando: cat INPUT1 | grep INPUT2  
 -Inputs:

- INPUT1:
  - Nome : file
  - Descrição : "Ficheiro onde se vai aplicar a procura"
  - Tipo : FILE
- INPUT2:
  - Nome : pattern
  - Descrição : "Padrao de procura"
  - Tipo : STR

\* Visuais

- Favicon : "image.png"
- PrimaryBgColor : #000
- SecondaryBgColor : #fff
- PrimaryTextColor : #fff
- SecondaryTextColor : #3f51b5
- LabelColor : #3f51b5
- BorderColor : #9e9e9e

É importante realçar novamente, que no formato de ficheiro de texto simples, a ordem dos campos é importante, assim como os seus delimitadores (ex.: '\* Servidor', '-' antes de cada ferramenta, etc.)

### Exemplo de ficheiro de configuração JSON

```
{
  "nome" : "Servidor",
  "diretoria" : "teste",
  "porta" : 20202,
  "ferramentas" : {
    "Grep" : {
      "Grep" : {
        "descricao" : "Descrição do comando",
        "comando" : "cat INPUT1 | grep INPUT2 & sleep 10",
        "inputs" : [
          {
            "id": "INPUT1",
            "opcoes" : {
              "nome" : "file",
              "descricao" : "Ficheiro onde se vai aplicar a procura",
              "tipo" : "FILE"
            }
          },
          {
            "id": "INPUT2",
            "opcoes" : {
              "nome" : "pattern",
              "tipo" : "STR",
              "descricao" : "Padrao de procura"
            }
          }
        ]
      }
    }
  }
}
```



### 3 Conclusão

Em suma, fizemos uma aplicação em Python que apenas com uma configuração simples (ficheiro de texto simples ou em json) é capaz de criar um serviço remoto para uma ou várias ferramentas. Para tal, fizemos uso dos conhecimentos obtidos ao longo de todo o perfil de Engenharia de Linguagens. Utilizamos boa parte do conhecimento obtido em Representação e Processamento de Conhecimento na Web (RPCW) na criação do modelo de servidor inteiramente em nodejs. Exploramos a ferramenta lark, como aprendemos em Engenharia Gramatical (EG), na criação de uma gramática para ler o ficheiro de texto com as configurações do servidor. E como seria de esperar, aplicamos os conhecimentos de scripting obtidos em Scripting no Processamento de Linguagem Natural (SPLN) com a criação da aplicação em si feita em Python, de forma a facilmente automatizar a tarefa de criação de servidor e com a sua disponibilização no Pypi com a ajuda do flit.

O produto final atingiu o resultado esperado e ainda mais, sendo que objetivo era fornecer ferramentas de linha de comando mais simples de forma remota, como filtros unix mas, se o utilizador entender, pode introduzir ferramentas mais complexas, como por exemplo o Whisper, exemplo por nós testado.

Entendemos ainda, que esta aplicação que criamos tem muita margem de progressão naquilo que pode ser adicionado como configurações para facilitar a vida do utilizador. Configurações estas como: autenticação no servidor, limite de usos das ferramentas por cliente do servidor, limite no tamanho dos ficheiros dos inputs, limite no tempo que os ficheiros outputs ficam no servidor, mais opções nas ferramentas (flags alternativas numa mesma ferramenta invés de criar duas semelhantes), maior facilidade para alterar as views do servidor, etc.

### Referências

1. Tool Server Generator
2. Express-generator
3. Ngrok