

# Trabalho de aprofundamento AP2

André Patacas, Gil Teixeira



# Aplicação para o cálculo de Largura de Banda e de latência

DETI

André Patacas, Gil Teixeira  
(93357) andrepatacas@ua.pt, (88194) gilteixeira@ua.pt

9 de Abril de 2019

# Conteúdo

<b>1</b>	<b>Introdução</b>	<b>1</b>
<b>2</b>	<b>Metodologia</b>	<b>2</b>
<b>3</b>	<b>Aplicação de Speed Test</b>	<b>3</b>
3.1	client.py . . . . .	3
3.1.1	main . . . . .	3
3.1.2	calc_download() . . . . .	3
3.1.3	calc_latency() . . . . .	4
3.1.4	country_test() . . . . .	4
3.1.5	create_signed_document . . . . .	4
3.1.6	id_test() . . . . .	4
3.1.7	random_test() . . . . .	4
3.1.8	report() . . . . .	5
3.1.9	run_tests() . . . . .	5
3.1.10	validate() . . . . .	5
3.1.11	usage() . . . . .	5
3.1.12	log() . . . . .	6
3.1.13	log_error() . . . . .	6
3.1.14	log_warning() . . . . .	6
3.1.15	log_verbose() . . . . .	6
3.1.16	load_server() . . . . .	6
3.2	test_client . . . . .	6
3.3	speed_test_result . . . . .	7
3.3.1	Construtor . . . . .	7
3.3.2	getObjDict . . . . .	7
<b>4</b>	<b>Análise de um exemplo e Conclusão</b>	<b>8</b>
4.1	Exemplo de utilização . . . . .	8

## **Resumo**

Este relatório pretende descrever uma aplicação desenvolvida para calcular a largura de banda e a latência da máquina onde a aplicação se encontra a correr. Calculam-se estes valores para um determinado servidor ou para um conjunto, de cardinalidade especificável, de servidores de um país também este especificável. No final a aplicação cria um relatório, (*report.csv*) , em csv, e assina-o com uma chave privada, (*key.priv*), a ser fornecida pelo utilizador.

# Capítulo 1

## Introdução

A aplicação foi desenvolvida em python3 no âmbito da disciplina de Laboratórios de Informática, no ano letivo 2018/2019. A adicionar às especificações básicas pedidas, segundo o guião sobre regras do segundo trabalho de aprofundamento, construiu-se ainda suporte para pydocs, para haja uma explicação mais detalhada de cada classe e método do nosso projeto. O programa foi escrito com base em test driven development (Capítulo 2) e, como tal, os testes unitários e funcionais foram criados primeiro, seguidos por um esboço do programa que se pretendia e finalmente, por vários updates a ambos até chegar ao estado em que a aplicação se encontra agora. Finalmente é demonstrado em detalhe um exemplo de utilização da aplicação (Capítulo 3).

## Capítulo 2

# Metodologia

1. Criar o primeiro teste (*test\_client*) de forma a que, a cada método que é construído, se possa testar imediatamente se esse método cumpre exatamente com o que estava especificado;
2. Criar o esqueleto da aplicação;
3. Criar o pydoc para a aplicação e para os testes;
4. Ajustar os métodos de forma a que a que passem a todos testes;
5. Testar o programa manualmente e com testes funcionais;
6. Iterar o processo de debugging e correção de erros.

## Capítulo 3

# Aplicação de Speed Test

### 3.1 client.py

A forma de utilizar este programa está descrita em detalhe na Subseção 3.1.11. Toda a descrição feita neste relatório remete na mesma para a documentação criada a quando do desenvolvimento da aplicação, em pydoc.

#### 3.1.1 main

Ao correr a aplicação a ordem pela qual os métodos são chamados é a seguinte:

1. `load_server()` - Subseção 3.1.16;
2. `validate()` - Subseção 3.1.10;
3. `run_tests()` - Subseção 3.1.9;
4. `report()` - Subseção 3.1.8;
5. `create_signed_document` - Subseção 3.1.5.

#### 3.1.2 calc\_download()

Cálculo de largura de banda.

Este método pede ao *target\_server* um download de 100 mb. Durante 10 segundos é feito download dos dados enviados pelo mesmo. No final dos 10 segundos, se o download tiver sido superior a 10mb prossegue, caso contrário, discarta este servidor.

**Argumentos:** *target\_server*(dicionário com informação sobre o target server).

**Retorna:** (float)  $1/time\_download\_1mb$

### 3.1.3 calc\_latency()

Cálculo da latência.

Este método troca dez *PING-PONG* com o *target-server* e calcula o tempo médio em milisegundos entre estas trocas.

**Argumentos:** target server (dicionário com informação sobre o target server).

**Retorna:** (int) *average\_trade\_time* em ms.

### 3.1.4 country\_test()

Este método serve para calcular a largura de banda e latency da conexão a um servidor random do país passado como argumento.

**Argumentos:** (str) *target\_country*.

**Retorna:** objeto *SpeedTestResult* com as informações relativas aos resultados do teste.

### 3.1.5 create\_signed\_document

Este método gera um *signature file* assinando o *report* com a chave privada no *key\_path* especificado.

**Argumentos:**

1. *key\_path* (str): The path to the file that contains the key;
2. *report\_name* (str): Nome do *report* a ser assinado;
3. *signature\_name* (str): Nome do *signature file* que será gerado.

**Retorna:** *None*.

### 3.1.6 id\_test()

Este método serve para calcular a largura de banda e latency da conexão a um servidor com o id passado como argumento.

**Argumentos:** (int) *target\_id*.

**Retorna:** objeto *SpeedTestResult* com as informações relativas aos resultados do teste.

### 3.1.7 random\_test()

Este método serve para calcular a largura de banda e latency da conexão a um servidor random.

**Argumentos:** *None*.

**Retorna:** objeto *SpeedTestResult* com as informações relativas aos resultados do teste.



### 3.1.8 report()

Este método vai gerar um *test\_report* baseado numa lista de objetos *SpeedTestResult* passados como argumentos.

**Argumentos:**

1. List[objeto *SpeedTestResult*];
2. *report\_name* (str) - nome do ficheiro a ser gerado.

**Retorna:** *None*. O ficheiro *test\_report* será gerado

### 3.1.9 run\_tests()

Este método serve para calcular a largura de banda e latency da conexão a um *num* de servidores num país, ou a um server com o id passado por argumento. Nota: se o terceiro argumento for um id, a função realizará *num* testes a esses servidor, se for um país, fará *num* testes usando a função Subseção 3.1.4 e se não foi passado terceiro argumento realiza um teste random (Subseção 3.1.7).

**Argumentos:**

1. *interval*: intervalo de tempo entre cada teste realizado;
2. *num*: número de testes a realizar;
3. *id\_or\_country*: país (str) ou id (int) de um server;
4. *option*: -v se pretender correr a aplicação em modo *verbose*.

**Retorna:** objeto *SpeedTestResult* com as informações relativas aos resultados do teste.

### 3.1.10 validate()

Este método trata da validação dos argumentos passados pela variável sys.argv.

**Argumentos:** *None*.

**Retorna:** *None*.

### 3.1.11 usage()

Este método imprime a mensagem de erro passada como argumento e imprime a ajuda para utilização da aplicação. No campo *option* pode usar -v para entrar em modo verbose.

**Argumentos:** *message* (str).

**Retorna:** *None*.

### 3.1.12 log()

Este método imprime a mensagem de passada como argumento com a cor passada como argumento.

**Argumentos:**

1. *message* (str);
2. *colour* (str).

**Retorna:** *None*.

### 3.1.13 log\_error()

Este método chama Subseção 3.1.12 com a mensagem igual à passada como argumento mas com cor vermelho.

**Argumentos:** *message* (str).

**Retorna:** *None*.

### 3.1.14 log\_warning()

Este método chama Subseção 3.1.12 com a mensagem igual à passada como argumento mas com cor amarela.

**Argumentos:** *message* (str).

**Retorna:** *None*.

### 3.1.15 log\_verbose()

Este método chama Subseção 3.1.12 com a mensagem igual à passada como argumento mas com cor verde se o modo *verbose* estiver ativado.

**Argumentos:** *message* (str).

**Retorna:** *None*.

### 3.1.16 load\_server()

Este método lê o ficheiro *"servers.json"* e cria um dicionário global com a lista de servidores.

**Argumentos:** *None*.

**Retorna:** *None*.

## 3.2 test\_client

Este programa é constituída por métodos que são testes unitários aos da aplicação principal (Seção 3.1). Lista *{teste unitário : método (da Seção 3.1)}*:

1. *test\_calc\_download()* : Subseção 3.1.2;
2. *test\_calc\_latency()*: Subseção 3.1.3;

3. `test_country_test()`: Subseção 3.1.4;
4. `test_create_signed_document()`: Subseção 3.1.5;
5. `test_id_test()`: Subseção 3.1.6;
6. `test_random_test()`: Subseção 3.1.7;
7. `test_report()`: Subseção 3.1.8;
8. `test_run_test()`: Subseção 3.1.9.

### 3.3 speed\_test\_result

Este programa serve para criar objetos `SpeedTestResult` que têm, cada um, as informações respectivas a um teste. Tem apenas um construtor e um método:

#### 3.3.1 Construtor

O construtor da classe cria um objeto com os parametros passados como argumentos: **Argumentos:**

1. *server\_id* (int);
2. *download\_speed* (float);
3. *latency* (int);

#### 3.3.2 getObjDict

Este método devolve um dicionário com os resultados do teste relativo ao objeto.

**Argumentos:** *None*.

**Retorna:** *testResult* (dict).

## Capítulo 4

# Análise de um exemplo e Conclusão

### 4.1 Exemplo de utilização

Com exemplo ir-se-á correr a aplicação Seção 3.1 com os argumentos:

1. *interval* = 5 (segundos);
2. *num* = 3 (testes);
3. *id\_or\_country* = Portugal;
4. *option* = -v (*verbose*).

Ao correr a aplicação com estes argumentos, segundo a Subseção 3.1.11, vêm os seguintes resultados:

```
gil@gil-teixeira:~/Desktop/lab1-ap02$ python3 client.py 5 3 Portugal -v
Starting Test Phase
  Starting a Network Test to Server in Portugal
    Starting Download Speed Test to porto.speedtest.net.zon.pt
    Download Speed Test done to porto.speedtest.net.zon.pt: 1.3364588900746175MB/s
    Starting Latency Test to porto.speedtest.net.zon.pt
    Latency Test done to porto.speedtest.net.zon.pt: 15ms
  Starting a Network Test to Server in Portugal
    Starting Download Speed Test to speedtest1.meo.pt
    Download Speed Test done to speedtest1.meo.pt: 1.700130344277497MB/s
    Starting Latency Test to speedtest1.meo.pt
    Latency Test done to speedtest1.meo.pt: 12ms
  Starting a Network Test to Server in Portugal
    Starting Download Speed Test to speedtest3.meo.pt
    Download Speed Test done to speedtest3.meo.pt: 1.904441075354276MB/s
    Starting Latency Test to speedtest3.meo.pt
    Latency Test done to speedtest3.meo.pt: 11ms
Test Phase Ended
Starting Report Creation Phase
Report Created  Starting to Sign the Report
Report Signed
```

Criando-se dois novos ficheiros na pasta onde está a aplicação:

- report.sig, contendo uma assinatura do relatório pela chave privada fornecida (*key.priv*).
- report.csv, um ficheiro Comma-separated values (CSV) com os resultados dos três testes efetuados:

Contador	Id Do Servidor	Data e Hora no Formato ISO	Latencia	Largura de Banda	Check
1	9729	2019-04-19 23:04:48.745678	13.1	3335593613565648	aa8c139784e517d2f79a785fa224767b8714d494d8a4ede60a755c184019763e
2	9729	2019-04-19 23:05:03.921081	14.1	682293124975788	d12a70a22302919711a5f9233340b27b6cfeb4d2ac81ed2f3575048bca51012a
3	1902	2019-04-19 23:05:19.024825	6.1	882504664392915	2f1b6bc5511c4db3e74143fc903bf11a82dad81d0dcfb6b2fb3927af85542c1e

# Acrónimos

**CSV** Comma-separated values