

Speedtest em Python

Universidade de Aveiro

José Vasco Carvalho de Sousa, José Luís
Rodrigues Costa,



Speedtest em Python

Departamento de Eletrónica Telecomunicações e
Informática

Universidade de Aveiro

José Vasco Carvalho de Sousa, José Luís Rodrigues Costa,
(93049) jvcs@ua.pt, (92996) joselcosta@ua.pt

23 de Abril de 2019

Resumo

Este relatório aborda a motivação, a implementação e apresenta testes que comprovam o funcionamento correto do programa Speedtest, desenvolvido em Python[1],

Atualmente, a largura de banda[2] e a latência[3] são dois dos principais fatores que levam os clientes a escolher uma determinada operadora de telecomunicações.

Com efeito de avaliar estes dois fatores, existem diversos programas que calculam de uma forma mais ou menos precisa ambos.

Este programa faz essas avaliações utilizando sockets e o protocolo TCP[4]. De uma forma sucinta, para o teste de latência o programa começa por registrar o tempo atual, de seguida envia uma mensagem ao servidor, recebe a resposta e por fim regista novamente o tempo e faz a diferença entre o tempo atual e o tempo anterior. No fim deste processo temos a latência entre a nossa máquina e o servidor em milissegundos (ms).

Para o teste de largura de banda o programa começa por registrar o tempo atual, de seguida envia uma mensagem ao servidor, recebe uma quantidade x Mb de dados e por fim regista novamente o tempo, faz a diferença entre o tempo atual e o tempo anterior e faz também a divisão pela quantidade x Mb. No fim deste processo temos a velocidade de download em mb/s (mb/s).

Conteúdo

1	Introdução	1
2	Metodologia	2
2.1	Linguagem	2
2.2	Código	2
2.2.1	Bibliotecas Utilizadas	2
2.2.2	Validação dos Argumentos	3
2.2.3	Escolha do Servidor	3
2.2.4	Teste de Latência	5
2.2.5	Teste de largura de banda	7
2.2.6	Escrita dos resultados num ficheiro CSV	9
2.2.7	Assinatura do ficheiro CSV	9
3	Testes de Depuração	11
3.1	Argumentos	11
3.2	Conexão ao servidor	12
3.3	Escrita do Ficheiro CSV	13
3.4	Assinatura Digital	14
4	Resultados	15
5	Conclusões	17

Capítulo 1

Introdução

Este relatório aborda a motivação, a implementação e apresenta testes que comprovam o funcionamento correto do programa Speedtest, desenvolvido em Python[1],

Este relatório está dividido em quatro capítulos. Depois desta introdução, no Capítulo 2 é apresentada a metodologia seguida, no Capítulo 3 são apresentados testes de depuração, no Capítulo 4 são apresentados os resultados obtidos, Finalmente, no Capítulo 5 são apresentadas as conclusões do relatório.

Capítulo 2

Metodologia

2.1 Linguagem

O programa Speedtest foi desenvolvido na linguagem de programação Python[1], sendo utilizada a versão 3.7 Esta linguagem é uma linguagem de alto nível desenvolvida pela comunidade e gerida pela organização Python Software Foundation[5].

Nos dias de hoje esta linguagem é bastante popular, sobretudo na área da inteligência artificial.

2.2 Código

2.2.1 Bibliotecas Utilizadas

Para facilitar a implementação de certas funcionalidades deste programa foram utilizadas as seguintes bibliotecas.

```
import socket
import time
import random
import datetime
import sys
import json
import csv
import hashlib
from Crypto.PublicKey import RSA
from Crypto.Hash import SHA256
from Crypto.Signature import PKCS1_v1_5
```

2.2.2 Validação dos Argumentos

Com o objetivo de validar os argumentos fornecidos pelo utilizador, foi implementado um algoritmo de validação

Em primeiro lugar esse algoritmo verifica se o número de argumentos fornecidos corresponde ao número de argumentos necessário para o correcto funcionamento do programa.

Por fim o algoritmo verifica se os argumentos interval (tempo que decorre entre dois testes realizados), num (número de testes a serem realizados) e [country or id] (nome do país do servidor ou identificador do servidor), são números positivos nos dois primeiros casos e número ou string no terceiro.

Caso estas duas condições não sejam verificadas, o programa não corre e mostra uma mensagem de erro.

```
#Verificar validade dos argumentos
def checkArguments():
    if len(sys.argv) < 4:
        print("Ajuda: python3 client.py interval num [country or id]")
        sys.exit(0)

    if not sys.argv[1].isdigit() and int(sys.argv[1]) > 0:
        print("Erro: argumento 'interval' tem um formato incorreto")
        sys.exit(0)

    if not sys.argv[2].isdigit() and int(sys.argv[2]) > 0:
        print("Erro: argumento 'num' tem um formato incorreto")
        sys.exit(0)

    if not (sys.argv[3].isnumeric() or sys.argv[3].isalpha()):
        print("Erro: argumento 'country or id' tem um formato incorreto")
        sys.exit(0)
```

2.2.3 Escolha do Servidor

Para efetuar a escolha do servidor que serve para efetuar os testes, o programa tem dois métodos que possibilitam esta escolha

Em primeiro lugar o programa converte o conteúdo do ficheiro servers.json num objecto JSON.

De seguida o algoritmo verifica se o último argumento é um ID (número) ou um nome de um país (string).

Caso seja um ID, é feita a iteração do objeto JSON e quando é encontrado o servidor com o ID correspondente ao ID fornecido no argumento, é escolhido esse mesmo servidor.

Caso seja um país, é feita a iteração do objeto JSON e quando é encontrado um servidor com o país correspondente ao país fornecido no argumento, este é armazenado num novo array. De seguida é gerado um número aleatório

que servirá como índice do novo array, que contém todos os servidores de um determinado país.

```
#Le o ficheiro servers.json e retorna-o como um objecto JSON
def loadServers():
    with open('servers.json') as f:
        data = json.load(f)
        return data

#Seleciona um servidor tendo em conta o argumento fornecido
def selectServer(arg):
    allServers = loadServers()

    if arg.isalpha():
        targetCountryServers = []
        i = 0

        #Caso o nome do pais inclua espacos como por exemplo United
        States
        if (len(sys.argv) > 4):
            for i in range(4, len(sys.argv)):
                arg += " " + sys.argv[i]

        #Caso o argumento seja um nome de um pais (alphanum)
        for server in allServers["servers"]:
            if server["country"].lower() == arg.lower():
                targetCountryServers.append(server)
                i += 1

        #Verificar se o array com os possiveis servidores esta vazio
        if len(targetCountryServers) == 0:
            print("Erro: Nao existe nenhum servidor localizado no
                pais fornecido")
            sys.exit(0)
        else:
            k = random.randint(0,(i - 1))
            return targetCountryServers[k]["host"].split(":")

    #Caso o argumento seja um ID (digit)
    elif arg.isnumeric():
        for server in allServers["servers"]:

            if server["id"] == int(arg):
                return server["host"].split(":")

    #Caso nao seja encontrado nenhum servidor com um id
    correspondente
    print("Erro: Nao existe nenhum servidor com o id fornecido")
```

```
sys.exit(0)

else:
    print("Erro: Nao foi encontrado nenhum servidor")
    sys.exit(0)
```

2.2.4 Teste de Latência

Com o objetivo de realizar o teste de latência o programa começa por criar um socket, conectando-se ao servidor previamente escolhido.

De seguida é enviado um comando para esse servidor pelo protocolo TCP[4], de seguida o servidor envia uma resposta com o timestamp atual. Por fim o programa regista novamente o tempo e faz a diferença entre o tempo atual e o tempo anteriormente registado. O resultado é a latência entre a máquina e o servidor em ms.

```
#Faz o teste de ping ao host fornecido
def ping(host, port):
    i = 0
    totalPing = 0
    print("> Teste de latencia iniciado < \n")

    while i < 10:
        #Cria o socket do tipo tcp
        client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

        #0 tempo maximo de espera por ligacao ao servidor e 10 segundos
        client.settimeout(10)

        #Tenta a ligacao ao servidor, caso falhe uma mensagem e mostrada
        #e o valor de ping e -1
        try:
            client.connect((host,port))

        except socket.timeout:
            print("Erro: Impossivel conectar ao servidor")
            return -1

        #Tempo em milisegundos antes de enviar o comando
        timeBefore = time.time() * 1000

        #Envio do comando para o servidor atraves de TCP
        client.send(("PING " + str(timeBefore) + "\n").encode("utf-8"))

        #Resposta do servidor
        response = client.recv(1024)

        #Tempo em milisegundos depois de receber a resposta do servidor
```

```

        timeAfter = time.time() * 1000

        #Fecha a conexao ao servidor
        client.close()

        #Mostra resposta do servidor
        print(response)

        #Delta do tempo, da a latencia em milisegundos
        deltaTime = timeAfter - timeBefore

        totalPing += deltaTime
        print("Latencia para " + str(host) + " e de " +
              str(round(deltaTime)) + "ms")
        i += 1
        time.sleep(2)

    print("\n")

    #Retorna o valor medio do ping
    return round(totalPing/10)

#Mostra resposta do servidor ao comando "HI \n"
def hello(host, port):
    #Cria o socket do tipo tcp
    client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    #Tempo maximo de espera por resposta do socket (10 segundos)
    client.settimeout(10)

    #Tenta a ligacao ao servidor, caso falhe uma mensagem e mostrada
    try:
        client.connect((host,port))

    except socket.timeout:
        print("Erro: Impossivel conectar ao servidor")

    #Envia o comando para o servidor
    client.send(("HI\n").encode("utf-8"))

    #Recebe a resposta do servidor
    response = client.recv(4096).decode('utf-8')

    #Mostra a resposta
    print(response)

    #Fecha a conexao com o servidor
    client.close()

```

2.2.5 Teste de largura de banda

Com o objetivo de realizar o teste de largura de banda o programa começa por criar um socket, conectando-se ao servidor previamente escolhido. Depois desta etapa, o programa regista o tempo atual, de seguida é enviado um comando para esse servidor pelo protocolo TCP[4], o servidor envia uma resposta com um tamanho aleatório entre 10Mb e 100Mb. Dentro de um ciclo infinito while, é lida a resposta enviada até que passem 10 segundos, neste caso o valor de download total em Mb é atualizado para o valor que realmente foi baixado, ou então até que o servidor deixe de enviar dados (fim do download). Quando uma destas condições é atingida o ciclo para e é registado novamente o tempo atual. Para efeitos de teste o conteúdo enviado pelo servidor é armazenado num ficheiro de texto.

A largura de banda é obtida com a divisão do download total em Mb e o tempo passado em segundos. Daí resulta a largura de banda em mb/s. Por fim o programa regista novamente o tempo e faz a diferença entre o tempo atual e o tempo anteriormente registado. O resultado é a latência entre a máquina e o servidor em ms.

```
#Faz o teste de ping ao host fornecido
def ping(host, port):
    i = 0
    totalPing = 0
    print("> Tesde de latencia iniciado < \n")

    while i < 10:
        #Cria o socket do tipo tcp
        client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

        #0 tempo maximo de espera por ligacao ao servidor e 10 segundos
        client.settimeout(10)

        #Tenta a ligacao ao servidor, caso falhe uma mensagem e mostrada
        #e o valor de ping e -1
        try:
            client.connect((host,port))

        except socket.timeout:
            print("Erro: Impossivel conectar ao servidor")
            return -1

        #Tempo em milisegundos antes de enviar o comando
        timeBefore = time.time() * 1000

        #Envio do comando para o servidor atraves de TCP
        client.send(("PING " + str(timeBefore) + "\n").encode("utf-8"))

        #Resposta do servidor
        response = client.recv(1024)
```

```

        #Tempo em milisegundos depois de receber a resposta do servidor
        timeAfter = time.time() * 1000

        #Fecha a conexao ao servidor
        client.close()

        #Mostra resposta do servidor
        print(response)

        #Delta do tempo, da a latencia em milisegundos
        deltaTime = timeAfter - timeBefore

        totalPing += deltaTime
        print("Latencia para " + str(host) + " e de " +
              str(round(deltaTime)) + "ms")
        i += 1
        time.sleep(2)

    print("\n")

    #Retorna o valor medio do ping
    return round(totalPing/10)

#Mostra resposta do servidor ao comando "HI \n"
def hello(host, port):
    #Cria o socket do tipo tcp
    client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    #Tempo maximo de espera por resposta do socket (10 segundos)
    client.settimeout(10)

    #Tenta a ligacao ao servidor, caso falhe uma mensagem e mostrada
    try:
        client.connect((host,port))

    except socket.timeout:
        print("Erro: Impossivel conectar ao servidor")

    #Envia o comando para o servidor
    client.send(("HI\n").encode("utf-8"))

    #Recebe a resposta do servidor
    response = client.recv(4096).decode('utf-8')

    #Mostra a resposta
    print(response)

    #Fecha a conexao com o servidor
    client.close()

```

2.2.6 Escrita dos resultados num ficheiro CSV

Para efetuar a escrita dos resultados obtidos num ficheiro CSV o programa começa por criar um novo ficheiro denominado como report.csv. De seguida são escritos os cabeçalhos ['counter', 'id', 'data', 'latency', 'bandwidth', 'check'] e depois as linhas com os valores obtidos com as funções previamente apresentadas.

```
#Iniciar o ficheiro para escrita / fazer testes

with open('report.csv', mode='w') as csv_file:
    fieldnames = ['counter', 'id', 'data', 'latency', 'bandwidth',
                  'check']

    #Escrever os cabecalhos
    writer = csv.DictWriter(csv_file, fieldnames=fieldnames)
    writer.writeheader()

    while True:
        serverData = selectServer(str(sys.argv[3]))
        curPing = ping(serverData[0], int(serverData[1]))
        curDownload = download(serverData[0], int(serverData[1]))

        numTests -= 1

        print("Resultados | Ping -> " + str(curPing) + "ms - Download
              -> " + str(curDownload) + "Mb/s")
        print("Info: Teste realizado com sucesso, faltam " +
              str(numTests) + " para terminar\n")

        #escrever as linhas
        writer.writerow({'counter': str(numTests), 'id':
                        str(matchHost(serverData[0] + ':' + serverData[1])),
                        'data': str(datetime.datetime.now().isoformat()),
                        'latency': str(curPing), 'bandwidth': str(curDownload),
                        'check': str(numTests) + str(matchHost(serverData[0] +
                        ':' + serverData[1])) +
                        str(datetime.datetime.now().isoformat()) + str(curPing)
                        + str(curDownload) })
```

2.2.7 Assinatura do ficheiro CSV

Com o objetivo de assinar o ficheiro CSV com uma chave privada, o programa começa por ler uma chave privada RSA denominada por key.priv.pem e faz

a assinatura digital do ficheiro report.csv linha a linha, e por fim escreve o resultado num ficheiro denominado por report.sig.

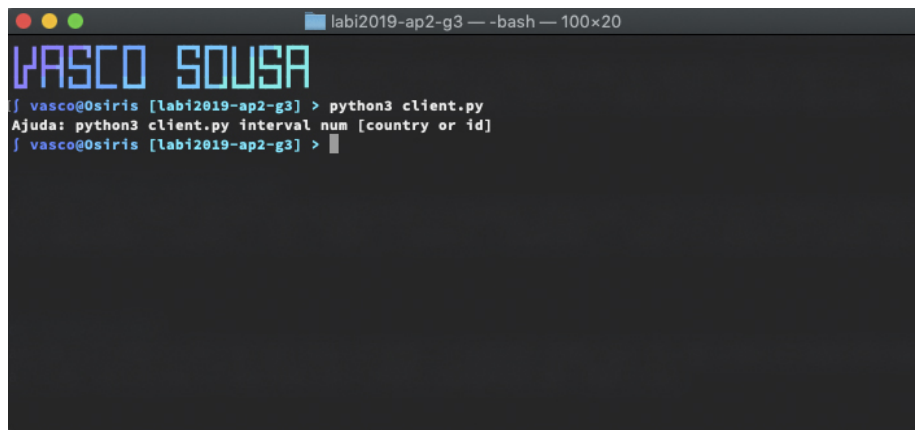
```
def signReport():
    privkey = RSA.importKey(open('key.priv.pem', 'r').read())
    signer = PKCS1_v1_5.new(privkey)
    unsigned_file = open('report.csv', 'r')
    signed_file = open('report.sig', 'wb')
    line = unsigned_file.readline()

    for line in unsigned_file:
        h = SHA256.new(line.encode('utf-8'))
        signed_line = signer.sign(h)
        signed_file.write(signed_line)
        line = unsigned_file.readline()
```

Capítulo 3

Testes de Depuração

3.1 Argumentos



```
labi2019-ap2-g3 — -bash — 100x20
VASC0 SOUSA
j vasco@0siris [labi2019-ap2-g3] > python3 client.py
Ajuda: python3 client.py interval num [country or id]
j vasco@0siris [labi2019-ap2-g3] > 
```

Figura 3.1: Output do programa quando se executa o mesmo sem fornecer argumentos



```
labi2019-ap2-g3 — -bash — 100x20
VASC0 SOUSA
[ vasco@osiris [labi2019-ap2-g3] > python3 client.py dez cinco 12
Erro: argumento 'interval' tem um formato incorreto
[ vasco@osiris [labi2019-ap2-g3] >
```

Figura 3.2: Output do programa quando se executa o mesmo fornecendo argumentos com o formato errado

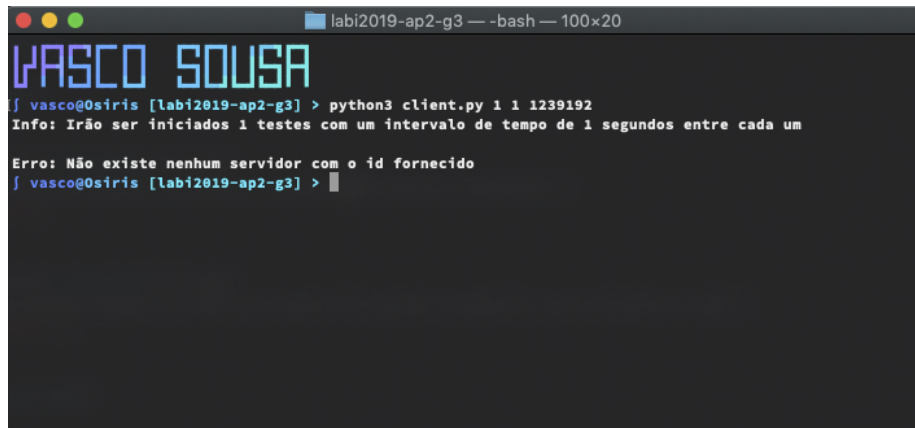
3.2 Conexão ao servidor



```
labi2019-ap2-g3 — -bash — 100x20
VASC0 SOUSA
[ vasco@osiris [labi2019-ap2-g3] > python3 client.py 1 1 oqiwiqsna
Info: Irão ser iniciados 1 testes com um intervalo de tempo de 1 segundos entre cada um

Erro: Não existe nenhum servidor localizado no país fornecido
[ vasco@osiris [labi2019-ap2-g3] >
```

Figura 3.3: Output do programa quando se executa o mesmo fornecendo um país em que não existem servidores



```
labi2019-ap2-g3 — -bash — 100x20
VASCO SOUSA
j vasco@0sirir [labi2019-ap2-g3] > python3 client.py 1 1 1239192
Info: Irão ser iniciados 1 testes com um intervalo de tempo de 1 segundos entre cada um

Erro: Não existe nenhum servidor com o id fornecido
j vasco@0sirir [labi2019-ap2-g3] >
```

Figura 3.4: Output do programa quando se executa o mesmo fornecendo um id que não responde a nenhum servidor



```
labi2019-ap2-g3 — -bash — 100x20
VASCO SOUSA
j vasco@0sirir [labi2019-ap2-g3] > python3 client.py 1 1 Portugal
Info: Irão ser iniciados 1 testes com um intervalo de tempo de 1 segundos entre cada um

> Tesde de latência iniciado <

Erro: Impossível conectar ao servidor
Erro: Impossível conectar ao servidor
Resultados | Ping -> -1ms - Download -> 0Mb/s
Info: Teste realizado com sucesso, faltam 0 para terminar

j vasco@0sirir [labi2019-ap2-g3] >
```

Figura 3.5: Output do programa quando se executa o mesmo sem conexão à internet ou quando não é possível conectar com o servidor

3.3 Escrita do Ficheiro CSV

```
labi2019-ap2-g3 — -bash — 100x20
VASC0 SOUSA
j vasco@0sirir [labi2019-ap2-g3] > python3 client.py 1 1 Portugal
Info: Irão ser iniciados 1 testes com um intervalo de tempo de 1 segundos entre cada um

O programa não consegue escrever no ficheiro report.csv, por favor verifique as permissões de escrita
j vasco@0sirir [labi2019-ap2-g3] >
```

Figura 3.6: Output do programa quando se executa o mesmo e não existem permissões para criar e escrever no ficheiro report.csv

3.4 Assinatura Digital

```
labi2019-ap2-g3 — -bash — 100x20
Latência para speedtest.nosmadeira.com é de 40ms
Latência para speedtest.nosmadeira.com é de 39ms
Latência para speedtest.nosmadeira.com é de 39ms
Latência para speedtest.nosmadeira.com é de 34ms
Latência para speedtest.nosmadeira.com é de 38ms
Latência para speedtest.nosmadeira.com é de 35ms
Latência para speedtest.nosmadeira.com é de 35ms
Latência para speedtest.nosmadeira.com é de 36ms
Latência para speedtest.nosmadeira.com é de 37ms

> Tesde de download iniciado <

Velocidade de download -> 87Mb descarregados em 14s (6.21Mb/s)

Resultados | Ping -> 37ms - Download -> 6.21Mb/s
Info: Teste realizado com sucesso, faltam 0 para terminar

O programa não consegue ler a chave privada! Por favor gere uma primeiro com python3 generatekeys.py
```

Figura 3.7: Output do programa quando se executa o mesmo e não existe um ficheiro key.priv.pem para ser utilizado ou não existem permissões para abrir esse mesmo ficheiro, para assinar digitalmente o report.csv

Capítulo 4

Resultados

Para testar o bom funcionamento do programa, foram feitos dez testes espaçados de dez segundos numa conexão à internet fibra de uma empresa de telecomunicações portuguesa.

Cinco destes testes foram feitos em servidores localizados em Portugal, os outros cinco foram realizados em servidores localizados na China.

Counter	Id	Date	Latency (ms)	Bandwidth (mb/s)
4	1758	2019-04-24T10:46:35.893562	19	4.38
3	1249	2019-04-24T10:47:36.168147	47	4.40
2	19253	2019-04-24T10:48:28.458914	27	4.21
1	1249	2019-04-24T10:49:19.441649	19	5.28
0	9729	2019-04-24T10:50:07.572682	21	5.77

Tabela 4.1: Resultados dos testes realizados em servidores portugueses

Counter	Id	Date	Latency (ms)	Bandwidth (mb/s)
4	21584	2019-04-24T10:55:01.546779	322	1.02
3	17228	2019-04-24T10:56:36.157917	345	0.90
2	17222	2019-04-24T10:57:18.259643	321	0.89
1	17437	2019-04-24T10:58:09.124659	310	0.89
0	9484	2019-04-24T10:59:12.772322	384	0.90

Tabela 4.2: Resultados dos testes realizados em servidores chineses

Comparando os resultados obtidos, podemos concluir que quanto mais próximo os servidores estão fisicamente da nossa máquina, menor a latência e menor também a largura de banda, como seria de esperar.

Apesar destes resultados serem bastante próximos da realidade, existem fatores, que podem ter influência nos resultados finais, que o programa speedtest não tem em conta.

Capítulo 5

Conclusões

Concluindo, este projeto utiliza todas as técnicas lecionadas no segundo semestre do primeiro ano da Unidade Curricular de Laboratórios de Informática para a realização de um Speedtest desenvolvido em Python[1], com toda a funcionalidade proposta e com toda a documentação necessária.

Foram também feitos os testes de depuração necessário para avaliar o bom funcionamento do programa.

Os resultados de latência e download são influenciados pela distância física da máquina ao servidor de teste, como era de esperar.

Salientamos que os resultados deste Speedtest tanto na latência, como na largura de banda aproximam os valores reais, mas não são totalmente precisos devido a influências de pequenos fatores como o "slow start" do protocolo TCP.

Contribuições dos autores

Para a realização deste trabalho, ambos os autores desenvolveram o código dos ficheiros .py, sendo que Vasco Sousa teve uma maior influência no desenvolvimento dos métodos envolvidos no cálculo da latência, da largura de banda e seleção de servidor. Já Luis Costa foi o principal responsável nos métodos responsáveis por ler a chave RSA, assinar digitalmente o ficheiro report.csv e escrever no ficheiro report.csv.

Apesar de o desenvolvimento do programa ter sido distribuído pelos dois autores, o desenvolvimento foi mútuo e partilhado pelos mesmos.

Ambos os autores contribuíram para a escrita de cada um dos capítulos deste documento.

De uma forma geral, as contribuições dos autores são de 50% de Vasco Sousa e 50% para Luis Costa.

Acrónimos

ms milissegundos

mb/s mb/s

Bibliografia

- [1] Wikipédia, *Python*, [Online; acedido em 23 de Abril de 2019]. URL: <https://pt.wikipedia.org/wiki/Python>.
- [2] —, *Bandwidth*, [Online; acedido em 23 de Abril de 2019]. URL: [https://en.wikipedia.org/wiki/Bandwidth_\(computing\)](https://en.wikipedia.org/wiki/Bandwidth_(computing)).
- [3] —, *Ping*, [Online; acedido em 23 de Abril de 2019]. URL: <https://pt.wikipedia.org/wiki/Ping>.
- [4] —, *TCP*, [Online; acedido em 23 de Abril de 2019]. URL: https://pt.wikipedia.org/wiki/Transmission_Control_Protocol.
- [5] Python Software Foundation, *Python Software Foundation*, [Online; acedido em 23 de Abril de 2019]. URL: <https://www.python.org/psf/1>.