

RELATÓRIO DO AP2

Universidade de Aveiro

Pompeu Gabriel Simões da Costa, Tomás Cerca
Rodrigues



RELATÓRIO DO AP2

MIECT

Universidade de Aveiro

Pompeu Gabriel Simões da Costa, Tomás Cerca Rodrigues
(103294) pompeu@ua.pt, (104090) tcercarodrigues@ua.pt

27 de maio de 2021

Índice

1	Introdução e funcionamento	1
2	Validação e <i>ficheiro csv</i>	2
2.1	Cliente	2
2.1.1	Código	2
2.2	Servidor	4
2.2.1	Código	5
3	Jogo	6
4	Interação e troca de dicionários	8
5	Segurança	9

Capítulo 1

Introdução e funcionamento

O tema do trabalho foi criar um servidor que suporte a geração de um número inteiro aleatório entre 0 e 100. Este número é mantido e segredo tal como o número de tentativas concedidas, que por sua vez variam entre 10 e 30. Ou seja, um jogo de adivinha o número secreto.

O servidor nunca deverá aceitar mais que um cliente em simultâneo com a mesma identificação e deverá criar um ficheiro denominado **report.csv** onde irá escrever os resultados dos clientes quando estes terminam o jogo.

O cliente poderá desistir a qualquer altura e o jogo se dará por finalizado quando não restarem mais tentativas, ou quando o cliente adivinhar o número secreto. Caso o cliente exceda o número de tentativas, o jogo será considerado sem sucesso mesmo que ele tenha adivinhado o número. Quando o jogo acaba conforme as regras o cliente deverá escrever no monitor uma mensagem a indicar se adivinhou ou não o número secreto e quantas jogadas efectuou. Em seguida, o servidor deverá acrescentar ao ficheiro as informações do jogo: cliente, número secreto, número máximo de tentativas, tentativas efectuadas e o resultado obtido (desistência, sucesso, ou fracasso).

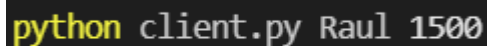
Capítulo 2

Validação e *ficheiro csv*

A comunicação entre os clientes e o servidor é suportada por *sockets* TCP. E para isso tanto os clientes, como o servidor necessitam de atender a certos requisitos.

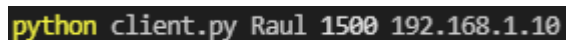
2.1 Cliente

Deve ser invocado no formato: `python3 client.py cliente porto [máquina]`



```
python client.py Raul 1500
```

Figura 2.1: Exemplo sem especificar a máquina



```
python client.py Raul 1500 192.168.1.10
```

Figura 2.2: Exemplo a especificar a máquina

2.1.1 Código

Dentro da função *main()* temos o primeiro *if* que verifica se o número de argumentos está correto.

Em seguida com um *try* verificamos se os valores relacionados com o porto são inteiros e com o *if* contido neste, é verificado se os valores são maiores que 0.

E por fim temos outro *if* contendo vários *if*'s dentro, verificando o quarto argumento e se este obedece a todos os requisitos propostos.

```
209 def main():
210     # validate the number of arguments and eventually print error message and exit with error
211     # verify type of of arguments and eventually print error message and exit with error
212     if len(sys.argv) != 3 and len(sys.argv) != 4:
213         print("Erro. Numero errado de argumentos.")
214         print("Uso: python3 client.py <client id> <porto> [maquina]")
215         sys.exit(1)
216
217     try:
218         port = int(sys.argv[2])
219
220         if port <= 0:
221             print("Porto tem de ser maior que 0")
222             sys.exit(1)
223     except ValueError:
224         print("Porto tem de ser um valor inteiro")
225         sys.exit(1)
226
227     hostname = "127.0.0.1"
228
229     if len(sys.argv) == 4:
230         ip = sys.argv[3].split(".")
231
232         if len(ip) != 4:
233             print("ERRO. O ip deve ser do tipo X.X.X.X")
234             sys.exit(1)
235
236         if int(ip[0]) <= 0 or int(ip[0]) > 255:
237             print("ERRO. O primeiro octeto deve estar entre ]0 , 255]")
238             sys.exit(1)
239
240         if int(ip[1]) < 0 or int(ip[1]) > 255 or int(ip[2]) < 0 or int(ip[2]) > 255:
241             print(
242                 "ERRO. O segundo e terceiro octeto devem estar entre [0 , 255]")
243             sys.exit(1)
244
245         if int(ip[3]) <= 0 or int(ip[3]) >= 255:
246             print("ERRO. O ultimo octeto deve estar entre ]0 e 255[")
247             sys.exit(1)
248
249         hostname = ip
250
```

Figura 2.3: Validação dos argumentos

2.2 Servidor

Caso seja iniciado com um número incorreto de argumentos ou com qualquer argumento inválido, será apresentada uma mensagem com o respetivo erro mencionado. Desta forma o ficheiro *report.csv* tem a seguinte estrutura: identificador do cliente, número secreto, número máximo de jogadas concedidas, número de jogadas realizadas pelo cliente e resultado.

O resultado será um dos três seguintes: *QUIT* caso o cliente desista; *SUCCESS* caso o cliente acerte o número dentro do número de tentativas que lhe é disponibilizado; *FAILURE* caso esgote o número de tentativas sem ter acertado o número secreto.

```
1 cliente,numero secreto,numero maximo de jogadas,numero de jogadas,resultado
2
3 Carlos,76,16,4,SUCCESS
4
5 Raul,90,29,7,SUCCESS
6
7 Tomas,61,28,6,SUCCESS
8
9 Ismael,15,19,8,SUCCESS
10
11 Adolfo,58,30,30,FAILURE
12
13 Mateus,19,15,3,QUIT
```

Figura 2.4: report.csv

2.2.1 Código

```
# Suporte da criação de um ficheiro csv com o respectivo cabeçalho
def create_file():
    print("A criar ficheiro csv")
    # Inicializa o ficheiro csv e escreve o cabeçalho
    file = open(f_name, "w")
    writer = csv.writer(file)
    writer.writerow(header)
    file.flush()
    file.close()
    print("Ficheiro criado com sucesso")

# Suporte da actualização de um ficheiro csv com a informação do cliente e resultado
def update_file(client_id, secret_number, max_attempts, attempts, result):
    try:
        print("A escrever dados de " + client_id + " para o ficheiro")
        # Abre em modo append para não sobrepor o ficheiro criado anteriormente em create_file
        file = open(f_name, "a")
        writer = csv.writer(file)
        line = [client_id, secret_number, max_attempts, attempts, result]
        writer.writerow(line)
        file.flush()
        file.close()
        print("Dados escritos com sucesso")
    except OSError:
        print("Erro ao escrever no ficheiro")
        print("linha -> " + client_id + " , " + secret_number +
              " , " + max_attempts + " , " + attempts + " , " + result)
```

Figura 2.5: Código da criação e actualização do *ficheiro csv*

Capítulo 3

Jogo

Após os processos de validação/preparação estarem completos cria-se um perfil de cliente e dá-se início ao jogo. O cliente será aceito caso ele ainda não esteja na lista de clientes ativos do servidor. Em sequência é gerado um número aleatório entre 0 e 100, e da mesma forma aleatória será dada, ao jogador, um número máximo de jogadas.

```
136 # Gerar o número secreto e o máximo de tentativas
137 secret_number = random.randint(0, 100)
138 max_attempts = random.randint(10, 30)
```

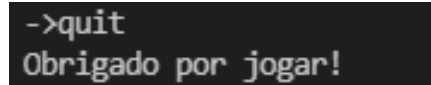
Figura 3.1: Código que gera o número secreto e o número de tentativas.

Em seguida é efetuado o registro dos clientes à estrutura de clientes ativos, assim evitando a inscrição de outro cliente com identificação idêntica e devolve um dicionário indicando que a operação de registo deste cliente foi feita com sucesso.

```
PS C:\Users\blues\Desktop\labi2021-ap2-g36\client-server> python client.py Lucas 1500
A tentar conectar ao servidor
conectado
Deseja comunicar com encriptacao? (S/N) ->n
Caso queira sair do jogo escreva: quit
Caso queria parar o jogo escreva: stop
```

Figura 3.2: Confirmação do registo.

Se o cliente quiser desistir do jogo (operação **QUIT**) terá apenas que escrever "*quit*" e o jogo será encerrado. Caso o cliente decida jogar normalmente até

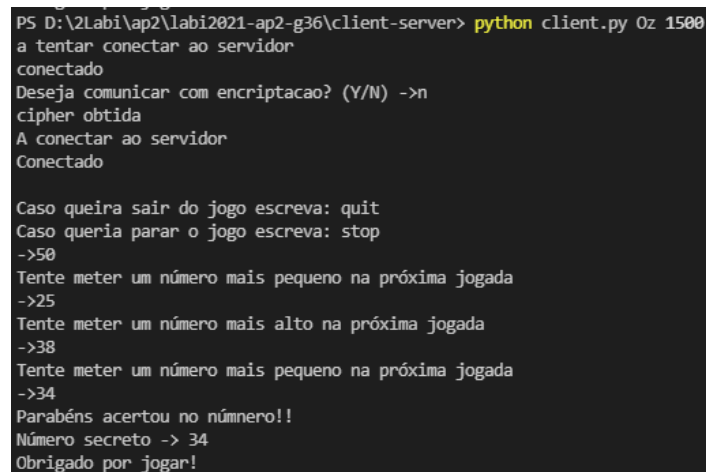


```
->quit  
Obrigado por jogar!
```

Figura 3.3: Dististência de um cliente.

ser bem sucedido acontecerão os seguintes processos. O cliente irá contactar o servidor para fazer uma jogada (operação **GUESS**). Caso esteja tudo certo com os processos de validação, o servidor irá responder, caso o cliente não tenha acertado o número, que o número está baixo ou alto, dependendo do valor referido pelo cliente em relação ao número secreto.

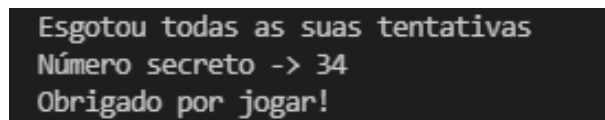
Se o cliente acertar o número secreto, o servidor irá notificar o cliente da sua vitória.



```
PS D:\2Labi\ap2\labi2021-ap2-g36\client-server> python client.py 0z 1500  
a tentar conectar ao servidor  
conectado  
Deseja comunicar com encriptacao? (Y/N) ->n  
cipher obtida  
A conectar ao servidor  
Conectado  
  
Caso queira sair do jogo escreva: quit  
Caso queria parar o jogo escreva: stop  
->50  
Tente meter um número mais pequeno na próxima jogada  
->25  
Tente meter um número mais alto na próxima jogada  
->38  
Tente meter um número mais pequeno na próxima jogada  
->34  
Parabéns acertou no número!!  
Número secreto -> 34  
Obrigado por jogar!
```

Figura 3.4: Jogo bem-sucedido de um cliente.

Caso o cliente não consiga acertar o número, assim gastando todas as tentativas, o que será mostrado no terminal é o seguinte:



```
Esgotou todas as suas tentativas  
Número secreto -> 34  
Obrigado por jogar!
```

Figura 3.5: Derrota de um cliente.

Capítulo 4

Interação e troca de dicionários

Para se comunicarem o servidor e o cliente trocam entre si dicionários, os quais são transformados em formato JSON antes de serem enviados.

Na recepção de uma mensagem (em formato JSON) esta é transformada em dicionário e depois abordada pelo sistema.

```
msg = {  
    "op": "GUESS",  
    "status": True,  
    "result": result  
}  
  
send_msg(client_sock, msg)
```

Figura 4.1: Dicionário com mensagem e função de envio.

Tanto no servidor como no cliente todas as mensagens são enviadas através de *sockets* que usam o protocolo TCP.

Como mencionado no Capítulo 3, durante a realização de uma jogada, o cliente contacta o servidor e é contactado pelo mesmo. Esta comunicação requer o uso dos dicionários.

Capítulo 5

Segurança

Após o cliente estar conectado é lhe perguntado se este deseja comunicar com encriptação. O cliente poderá escolher se **sim** (S), ou **não** (N).

Caso este escolha "**sim**"(S), o código do cliente irá gerar uma chave. Essa chave depois é usada pela cifra *aes*. Esta encripta os dados, os quais são codificados em *base64* e enviados através dos dicionários.

```
while response.lower() != "s" and response.lower() != "n":
    response = input("Deseja comunicar com encriptacao? (S/N) ->")

if response.lower() == "n":
    return None

if response.lower() == "s":
    return os.urandom(16)
```

Figura 5.1: Código referente à criação da chave.

Caso o cliente escolha não recorrer à encriptação nenhum valor será encriptado.

Contribuições dos autores

Pompeu Costa foi encarregado da programação do servidor, da conclusão do client e por testar. (65%)

Tomás Rodrigues também foi encarregado por testar, validar os argumentos no *client.py* e pelo relatório. (35%)