

## **Relatório de ASIST**

### **Sprint 2**

#### **Turma 3DJ Grupo 58**

1211396 João Batista

1211415 David Dias

1211417 Ezequiel Estima

1211469 Marco Andrade

**Data: 26/11/2023**

## Índice

Índice de quadros e figuras.....	3
Sprint 1.....	<b>Error! Bookmark not defined.</b>
Divisão das user stories .....	4
User story 640 .....	4
User story 650 .....	5
User story 660 .....	6
User Story 800 .....	10
User Story 810 .....	11
User Story 820 .....	12

## **Índice de quadros e figuras**

**Não foi encontrada nenhuma entrada do índice de ilustrações.**

## Divisão das user stories

	US640	US650	US660	US670	US800	US810	US820	US830
1211396	X					X		
1211415		X			X			
1211417			X					X
1211469				X			X	

### User story 640

*‘Como administrador do sistema quero que o deployment de um dos módulos do RFP numa VM do DEI seja sistemático, validando de forma agendada com o plano de testes’*

Para executar esta User Story (US), é necessário criar um script que seja executado regularmente. Este script começa por realizar um *git pull* no repositório. Em seguida, verifica se algum dos scripts já está em execução. Caso esteja, termina a execução do outro script e realiza os testes. Se os testes forem bem-sucedidos, executa o comando *npm run start*.

```
GNU nano 5.4 /etc/semSpi
/bin/bash

repo_path="/etc/semSpi/semSpi_g58_23_24"
backend_path="/etc/semSpi/semSpi_g58_23_24/MDRoboISEP"
lock_file="/etc/semSpi/deploy.lock"

# Pull
cd "$repo_path" || exit 1
git pull

# Verifica se o script esta a correr
if [ -e "$lock_file" ]; then
    pid=$(cat "$lock_file" 2>/dev/null)

    if ps -p "$pid" > /dev/null; then
        kill "$pid"
    fi
    rm -f "$lock_file"
fi

echo "$$" > "$lock_file"

cd "$backend_path" || exit 1

# Run testes
node_modules/.bin/mocha --require ts-node/register tests/**/*.test.ts

if [ $? -ne 0 ]; then
    echo "Tests failed."
    exit 1
else
    node_modules/.bin/mocha --require ts-node/register tests/**/*.test.ts
    if [ $? -ne 0 ]; then
        echo "Testes falharam"
        exit 1
    else
        echo "Testes passaram"
    fi
fi

# Run
npm run start

# Delete lock
rm -f "$lock_file"
```

Figura 1 - Script para fazer deployment e validação diária do back-end

De seguida usamos o comando *crontab -e*, e adicionamos a linha *0 23 \* \* \* /etc/sem5pi/deploy.sh*, de forma que o script seja executado todos os dias as 23 horas.

```
0 23 * * * /etc/sem5pi/deploy.sh
```

Figura 2 – Adicionar ao crontab o script para o deployment ser executado todos os dias as 23h

## User story 650

*‘Como administrador do sistema quero que apenas os clientes da rede interna do DEI (cablada ou via VPN) possam aceder à solução’*

A principal ideia desta US é limitar o tráfego de pacotes de rede onde a solução está a correr. Para isso vamos utilizar o sistema Netfilter, já integrado no kernel Linux, que permite a criação de regras relativas aos pacotes de rede que são aceites ou descartados.

Para impor estas limitações é necessário criar uma regra ACCEPT para todo o tráfego vindo dos ips rede interna do DEI e uma regra DROP para todo o outro tráfego

Os ips da rede interna do DEI podem ser determinados pela análise do ip de uma máquina que esteja ligada à rede. Por este processo conseguimos concluir que a network tem ip 10.9.0.0/16 (por VPN) e 172.18.0.0/16

Com esta informação basta agora inserir as regras na tabela filter das iptables. Foram executados os comandos *iptables -A INPUT -s 10.8.0.0/255.255.0.0 -p tcp --dport 4000 -j ACCEPT*, *iptables -A INPUT -s 172.18.0.0/255.255.0.0 -p tcp --dport 4000 -j ACCEPT* e *iptables -A INPUT -p tcp --dport 4000 -j DROP*. Nestes casos a opção -A indica a que cadeia desejamos adicionar a nova regra, -s os endereços de origem dos pacotes, -p protocolo utilizado, --dport o número do porto de destino e -j a ação a tomar.

```
root@asist:/etc/sem5pi# iptables -F
root@asist:/etc/sem5pi# iptables -S
-P INPUT ACCEPT
-P FORWARD ACCEPT
-P OUTPUT ACCEPT
root@asist:/etc/sem5pi# iptables -A INPUT -s 10.8.0.0/255.255.0.0 -p tcp --dport 4000 -j ACCEPT
root@asist:/etc/sem5pi# iptables -A INPUT -s 172.18.0.0/255.255.0.0 -p tcp --dport 4000 -j ACCEPT
root@asist:/etc/sem5pi# iptables -A INPUT -p tcp --dport 4000 -j DROP
root@asist:/etc/sem5pi# iptables -S
-P INPUT ACCEPT
-P FORWARD ACCEPT
-P OUTPUT ACCEPT
-A INPUT -s 10.8.0.0/16 -p tcp -m tcp --dport 4000 -j ACCEPT
-A INPUT -s 172.18.0.0/16 -p tcp -m tcp --dport 4000 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 4000 -j DROP
```

Figura 3 – Comandos a ser executados no terminal necessário para limitar o trafego de pacotes de rede

Caso queiramos guardar a configuração para, quando a máquina der reboot, a configuração se manter, usamos o comando *iptables-save*.

```
root@asist:/etc/sem5pi# iptables-save
# Generated by iptables-save v1.8.7 on Fri Nov 24 06:44:06 2023
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -s 10.8.0.0/16 -p tcp -m tcp --dport 4000 -j ACCEPT
-A INPUT -s 172.18.0.0/16 -p tcp -m tcp --dport 4000 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 4000 -j DROP
COMMIT
# Completed on Fri Nov 24 06:44:06 2023
root@asist:/etc/sem5pi# _
```

Figura 4 - Comando para verificação das alterações efetuadas nas ip-tables

### User story 660

*“Como administrador do sistema quero que os clientes indicados na user story anterior possam ser definidos pela simples alteração de um ficheiro de texto”*

O objetivo desta US é permitir que os administradores do sistema possam definir os clientes através de um ficheiro de texto para facilitar a gestão e controle dos mesmos. Para isto primeiro criamos um ficheiro de texto mediante o uso do comando “nano” para adicionarmos os IPs dos clientes no path “/etc/sem5pi/clientes.txt” como podemos visualizar na seguinte figura:

```
GNU nano 5.4 clientes.txt
10.8.0.0/255.255.0.0
```

Figura 5 - Ficheiro de texto onde devem ser contidos os ips e as máscaras de rede dos clientes

A seguir iremos escrever um script para atualizar as iptables no path “/etc/sem5pi/atualizar\_iptables.txt” também mediante o uso do comando nano, na figura a seguir mostramos como se deveria ver:

```

GNU nano 5.4                                atualizar_iptables.sh
#!/bin/bash

# Localização do arquivo com os IPs dos clientes
CLIENTES="clientes.txt"

# Limpar as regras existentes
iptables -F

# Adicionar regras para cada IP com máscara no ficheiro dos clientes.txt
while IFS= read -r ip
do
    iptables -A INPUT -s "$ip" -p tcp --dport 4000 -j ACCEPT
done < "$CLIENTES"

# Denegar qualquer outro ip que não esteja dentro do ficheiro
iptables -A INPUT -p tcp --dport 4000 -j DROP

# Salvar as regras
iptables-save

```

Figura 6 - Script para inserir regras nas iptables automaticamente utilizando o ficheiro clientes.txt

Neste script definimos uma variável chamada CLIENTES que armazena o caminho para o ficheiro que contém os endereços IP dos clientes com as suas respectivas máscaras de rede e a seguir limpamos todas as regras existentes nas iptables com o commando “iptables -F”. Após isto entramos num ciclo while onde irá se ler o ficheiro e atribuir-se à variavel ip cada linha do ficheiro em cada iteração até chegar ao fim do ficheiro. Dentro desse ciclo iremos executar o comando que adiciona uma nova regra ao iptables na chain INPUT para aceitar conexões TCP vindas do endereço IP obtido do ficheiro especificado na variável \$ip para a porta 4000. Após o loop, usamos outro comando que adiciona uma regra ao iptables para descartar todas as outras conexões TCP que tentam acessar a porta 4000. Isso significa que qualquer IP que não foi aceito explicitamente pelo loop anterior será bloqueado. No fim usa-se o comando iptables-save para guardar e exibir todas as regras do iptables no console.

Por último adiciona-se uma entrada no cron para executar este script regularmente todos os dias às 23h. Executando o comando sudo crontab -e e adicionando uma linha como aparece na figura a seguir conseguiríamos atingir esse objetivo:

```
GNU nano 5.4 /tmp/crontab.20zxo5/crontab * M
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m. every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow   command
0 23 * * * /etc/sem5pi/deploy.sh
0 23 * * * /etc/sem5pi/atualizar_iptables.sh_
```

GNU nano 5.4 interface showing crontab file editing. The file path is /tmp/crontab.20zxo5/crontab \*. The content includes comments about cron task syntax and two tasks: a daily backup at 5 a.m. and a script execution at 23:00. The bottom status bar shows various keyboard shortcuts like ^G Help, ^O Write Out, ^W Where Is, ^K Cut, ^T Execute, ^C Location, M-U Undo, ^X Exit, ^R Read File, ^\_ Replace, ^U Paste, ^J Justify, ^\_ Go To Line, M-E Redo.

Figura 7 - Adição do script ao crontab para execução diária do script



## User Story 670

*“Como administrador do sistema quero identificar e quantificar os riscos envolvidos na solução preconizada.”*

Para podermos identificar os riscos na nossa solução, começamos por identificar possíveis pontes de comunicação entre os diversos componentes da aplicação e onde estes estavam alocados. Também pensamos em possíveis causas da falha desses componentes. Assim chegamos aos seguintes riscos:

- Corte de energia nos servidores do ISEP
- Corte de internet no ISEP
- Avaria no Servidor do ISEP
- Desastre natural no ISEP
- Ataque informático
- Perder ligação à base de dados

Após termos identificados os riscos da nossa solução colocámo-los numa matriz de riscos. Sendo esta matriz composta por gravidade e probabilidade do risco. Neste caso, a gravidade está na horizontal e a probabilidade está na vertical.

	Catastrófico 4	Alto 3	Moderado 2	Insignificante 1
Frequente 4				
Provável 3		Corte de energia nos servidores do ISEP Corte de internet no ISEP		
Pouco provável 2		Perder ligação à base de dados Avaria no servidor do ISEP	Ataque informático	

Raro 1	Desastre natural no ISEP			
-----------	-----------------------------	--	--	--

Após colocados os riscos na matriz podemos calcular os valores de risco. Para calcular o risco basta multiplicar a probabilidade e a gravidade no qual esse risco está inserido, neste caso o valor máximo é de 16 e o mínimo é de 1.

Valores de cada risco:

- Corte de energia nos servidores do ISEP =  $3 \times 3 = 9$
- Corte de internet no ISEP =  $3 \times 3 = 9$
- Avaria no Servidor do ISEP =  $2 \times 3 = 6$
- Desastre natural no ISEP =  $1 \times 4 = 4$
- Ataque informático =  $2 \times 2 = 4$
- Perder ligação à base de dados =  $2 \times 3 = 6$

Por fim, podemos concluir que os maiores riscos estão onde os nossos componentes principais estão alocados, que neste caso é no ISEP e com isso devemos alocar uma parte substancial dos recursos possível nesta zona para manter os componentes livres de constrangimentos.

### User Story 800

A nossa aplicação MDRoboISEP funciona como back-end do nosso sistema e é basicamente o motor que impulsiona todas as aplicações front-end e interfaces com que os utilizadores interagem. Sem ele, esses pontos de comunicação não funcionam, o que significa que os utilizadores finais não podem realizar transações, acessar informações ou utilizar serviços. A disponibilidade contínua da aplicação MDRoboISEP é fundamental para preservar a consistência dos dados, o processamento de dados, a lógica de negócio e todas as integrações com as restantes APIs.

## User Story 810

*“Como administrador do sistema quero que seja proposta, justificada e implementada uma estratégia de cópia de segurança que minimize o RPO (Recovery Point Objective) e o WRT (Work Recovery Time)”*

Antes de falar sobre a estratégia, primeiro iremos explicar o conceito **RPO (Recovery Point Objective)** e **WRT (Work Recovery Time)** cujo está relacionado com o **RTO (Recovery Time Objective)**.

O RPO, é um conceito que representa o ponto de tempo para qual uma organização deve ser capaz de recuperar os dados após um incidente ou interrupção. Por outras palavras o RPO determina a quantidade máxima de dados que a organização esta disposta a perder me caso de falha.

O RTO, é uma métrica para definir o tempo máximo que o sistema levaria para que volte a sua operacionalidade normal após um desastre.

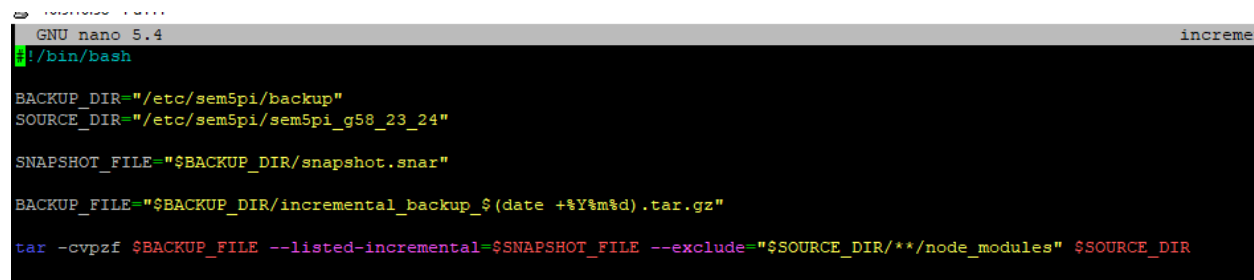
O WRT, é o tempo estimado que uma organização leva para que se testes e verifique todos os sistemas, podendo ser eles: sistemas, aplicações, base de dados entre outros.

Com base nesta informação a proposta que fazemos é que as copias de segurança sejam feitas com uma estratégia incremental, sendo realizado o backup quando existe menor fluxo de utilizadores, por exemplo de madrugada.

Tendo sido implementada esta estratégia a organização terá um RPO de no máximo 24 horas, equivalendo assim a 1 dia de trabalho.

Com isso o WRT também seria otimizado, pois as cópias de segurança seriam incrementais o que facilita a verificação de alterações especificas e a validação de que o processo de recuperação está funcionado conforme esperado, levando assim a um melhor RTO.

Para realizar o backup criamos um script que faz backups incrementais, e fizemos com que esse corresse todos os dias às 2 da manhã.



```
GNU nano 5.4                                     incremer
#!/bin/bash

BACKUP_DIR="/etc/sem5pi/backup"
SOURCE_DIR="/etc/sem5pi/sem5pi_g58_23_24"

SNAPSHOT_FILE="$BACKUP_DIR/snapshot.snar"

BACKUP_FILE="$BACKUP_DIR/incremental_backup_$(date +%Y%m%d).tar.gz"

tar -cvpzf $BACKUP_FILE --listed-incremental=$SNAPSHOT_FILE --exclude="$SOURCE_DIR/**/*.node_modules" $SOURCE_DIR
```

Figura 8 - Script incremental para realização dos backups

```
0 2 * * * /etc/sem5pi/incremental_backup.sh
```

Figura 9 - Adição do script ao crontab para execução diária às 2h

## User Story 820

*“Como administrador do sistema quero definir uma pasta pública para todos os utilizadores registados no sistema.”*

Para poder criar uma pasta pública acessível de outras virtual machines foi necessária a utilização do NFS. Para tal instalei o nfs para servidor e alterei o ficheiro em /etc/exports para o seguinte:

```
GNU nano 5.4 /etc/exports
/etc/exports: the access control list for filesystems which may be exported
#           to NFS clients.  See exports(5).
#
# Example for NFSv2 and NFSv3:
# /srv/homes hostname1(rw,sync,no_subtree_check) hostname2(ro,sync,no_subtree_check)
#
# Example for NFSv4:
# /srv/nfs4 gss/krb5i(rw,sync,fsid=0,crossmnt,no_subtree_check)
# /srv/nfs4/homes gss/krb5i(rw,sync,no_subtree_check)
#
/home/asist/pastaPublica *(rw,sync)
```

Figura 10 - Ficheiro exports

Esta linha indica o caminho da pasta a ser partilhada e com quem vai ser partilhada, neste caso todos da rede, para tal utilizei o \*, é são necessárias as permissões, que neste caso read e write e colocar como sync para as alterações serem de forma instantânea.

Para testar a funcionalidade utilizei outra máquina Linux com o nfs cliente instalado. Criar um mounting point onde para a pasta destino e utilizar o comando:

```
root@asist:/mnt/testing# mount -t nfs 10.9.10.58:/home/asist/pastaPublica /mnt/testing
```

Figura 11- Comando do lado do cliente

Para este caso, foi utilizado o ip da máquina servidor e a pasta de origem, e depois a pasta destino onde existem vão ser colocados os documentos presentes na pasta origem.

Para podermos testar se os ficheiros na pasta pública realmente estavam a ser partilhados foi criado um ficheiro no servidor cliente com o nome de agua.

```
[asist@asist:~/pastaPublica$ ls
agua
asist@asist:~/pastaPublica$
```

Figura 12 - Listagem dos ficheiros do lado do servidor

E podemos verificar que no lado do cliente também aparece o mesmo ficheiro:

```

root@asist:/# cd /mnt/testing
root@asist:/mnt/testing# ls
agua
root@asist:/mnt/testing# _

```

Figura 13 - Listagem dos ficheiros do lado do cliente

### User Story 830

*“Como administrador do sistema quero obter os utilizadores com mais do que 3 acessos incorretos”*

Para esta US é necessário analisar o ficheiro que contém os logs do sistema e fazer uma filtragem para determinar quais utilizadores têm mais de 3 acessos incorretos.

Para isso, foi criado o script `usersWith3IncorrectAccesses.sh` descrito na imagem abaixo:

```

GNU nano 5.4      usersWith3IncorrectAccesses.sh
#!/bin/bash

# Path para o ficheiro de logs
LOGS="/var/log/auth.log"

# Obter o nome de utilizador dos users com mais de 3 acessos incorretos
# grep todas as linhas com "Failed password"
# awk selecciona apenas os nome dos users (field na posição NF-5)
# sort organiza as linhas por ordem alfabetica
# uniq -c agrupa as linhas na forma n(nº de ocorrencias) user
# 2ª awk se n>3 print user

USERS=$(grep "Failed password" $LOGS | awk '{print $(NF-5)}' | sort | uniq -c | awk '$1>3 {print $2}')

if [ -z "$USERS" ]; then
    echo "Não existem users com mais de 3 acessos incorretos"
else
    echo "Users:"
    echo $USERS
fi

```

Figura 14 - Script para filtragem dos logs

O script vai buscar os logs do sistema, presentes no ficheiro `/var/log/auth.log`, filtra pelos que contêm “Failed Password”, agrupa por utilizador e imprime os nomes de utilizador com mais de 3 ocorrências.

```

root@asist:/etc/sem5pi# ./usersWith3IncorrectAccesses.sh
Users:
asist luser1
root@asist:/etc/sem5pi# _

```

Figura 15 - Execução do script