Relatório de ASIST
Sprint 2

Rute Ferreira, 1220994

Mateus Cabral, 1220704

Rui Soares, 1221283

Sandro Luís, 1221121

ISEP INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO

Índice

Divisão de User Stories

User Stories 6.4.1 e 6.4.5 – Mateus Cabral, 1220704

User Stories 6.4.2 e 6.4.6 – Rui Soares, 1221283

User Stories 6.4.3 e 6.4.7 – Sandro Luís, 1221121

User Stories 6.4.4 e 6.4.8 – Rute Ferreira, 1220994

User Story #6.4.1

Procedimento

Para implementar esta *User Story*, será necessário criar um script capaz de realizar de forma sistemática as seguintes tarefas: obter o módulo escolhido, validar sua funcionalidade executando os planos de teste e, por fim, fazer o *deploy* do módulo.

A solução foi implementada por meio de um script que realiza as etapas de obtenção, teste e *deploy* do módulo. Para garantir a execução sistemática do script, utilizou-se o *crontab*.

Execução

1. Criar um script com permissões adequadas de execução;
2. Exportar o path para garantir que o script tenha acesso às aplicações e adicionar variáveis e funções locais úteis durante a execução do programa.

Como o ambiente do *crontab* é limitado, é necessário exportar o *path* para que o script tenha acesso às aplicações e ferramentas necessárias. Sem essa linha, o script não conseguiria executar comandos dependentes de pacotes externos.

```
export PATH=$PATH:/root/.nvm/versions/node/v22.11.0/bin:/usr/local/bin

REPO_DIR="/home/asist/sem5pi"
MODULE_DIR="$REPO_DIR/3D-Visualization-Module"
LOG_FILE="/var/log/deploy_module.log"

log() {
    echo "$(date '+%Y-%m-%d %H:%M:%S') - $1" | tee -a "$LOG_FILE"
}
```

3. Criar a lógica responsável por obter a versão mais recente do módulo;

O script deve verificar o repositório ou o local adequado para obter a versão mais recente do módulo. Isso garante que o processo sempre utilize a versão mais atualizada do módulo.

```
log "Pulling updates..."
git pull origin Dev || { log "Error: Failed to pull updates"; exit 1; }
```

4. Realizar o *build* do módulo para validar seu funcionamento e registar o resultado;

O script deve fazer a compilação do módulo (build) para verificar se esta está funcional. O resultado do processo de compilação deve ser registado num log.

```
log "Attempting to build the program..."
BUILD_OUTPUT=$(ng build --optimization=false 2>&1)

if echo "$BUILD_OUTPUT" | grep -q "Application bundle generation complete"; then
    log "Build successful"
else
    log "Build failed"
    echo "$BUILD_OUTPUT" >> "$LOG_FILE"
    exit 1
fi
```

5. Interromper a execução de uma versão desatualizada do módulo e iniciar a versão mais recente;

Caso o módulo já esteja em execução, o script deve interromper a versão anterior e iniciar a nova versão do módulo. Esse processo também deve ser registado no log, incluindo detalhes sobre o *deploy* da nova versão.

```
log "Stopping any existing instance of the program..."
pkill -f "ng serve --host 10.9.10.8" || log "No running instance found to stop."

log "Starting the program..."
nohup ng serve --host 10.9.10.8 > /dev/null 2>&1 &

if [ $? -eq 0 ]; then
    log "Program started successfully"
    echo "The program has been updated, built, and is now running at http://10.9.10.8:4201." >> "$LOG_FILE"
else
    log "Failed to start the program"
    echo "The build was successful, but the program failed to start. Check the log at $LOG_FILE for details." >> "$LOG_FILE"
    exit 1
fi

log "Deployment completed successfully."
```

6. Adicionar o script ao crontab (utilizando crontab -e, a parte > dev/null serve para ignorar o stdout)

```
* * * * * /scripts/editIssue.sh
*/15 * * * * /scripts/deploy_3dModule.sh > /dev/null 2>&1
```

Resultado

Caso sucesso:

```
root@debian:/home/asist# cat /var/log/deploy_module.log
2024-11-20 21:51:02 - Starting deployment script
2024-11-20 21:51:02 - Pulling updates...
2024-11-20 21:51:03 - Installing dependencies...
2024-11-20 21:51:14 - Attempting to build the program...
2024-11-20 21:51:32 - Build successful
2024-11-20 21:51:32 - Stopping any existing instance of the program...
2024-11-20 21:51:32 - Starting the program...
2024-11-20 21:51:32 - Program started successfully
The program has been updated, built, and is now running at http://10.9.10.8:4201.
2024-11-20 21:51:32 - Deployment completed successfully.
```

Caso insucesso:

Para criar um caso de insucesso irei criar um erro num dos ficheiros do modulo escolhido e validar se a log regista esse erro.

```
root@debian:/home/asist# cat /var/log/deploy_module.log
2024-11-20 21:59:01 - Starting deployment script
2024-11-20 21:59:01 - Pulling updates...
2024-11-20 21:59:03 - Installing dependencies...
2024-11-20 21:59:13 - Attempting to build the program...
2024-11-20 21:59:28 - Build failed
› Building...
✔Building...
Application bundle generation failed. [12.629 seconds]
```

(O erro continua para baixo dando mais informação acerca do erro)

User Story #6.4.2

Procedimento:

Para a realização desta ***User Story*** será necessário configurar as regras de ***firewall***. As regras de ***firewall*** vão permitir ou bloquear o tráfego de rede. Neste caso, vou permitir que a rede interna do ***DEI*** aceda à solução, especificando a porta, que é a porta onde a solução será disponibilizada. Todo o tráfego originado de redes não autorizadas para essa porta será bloqueado. Estas regras serão configuradas a nível da ***chain INPUT***, uma vez que se pretende limitar o acesso, ou seja, o tráfego de entrada.

Execução:

1- Primeiramente é necessário obter a sequência de ***IPs*** que terá acesso ao sistema. Para isso conectei-me à rede interna do ***DEI*** através de um cabo ***Ethernet***. Em seguida acedi à Linha de Comandos do ***Windows*** e usei o comando ***ipconfig*** para obter as configurações de rede:

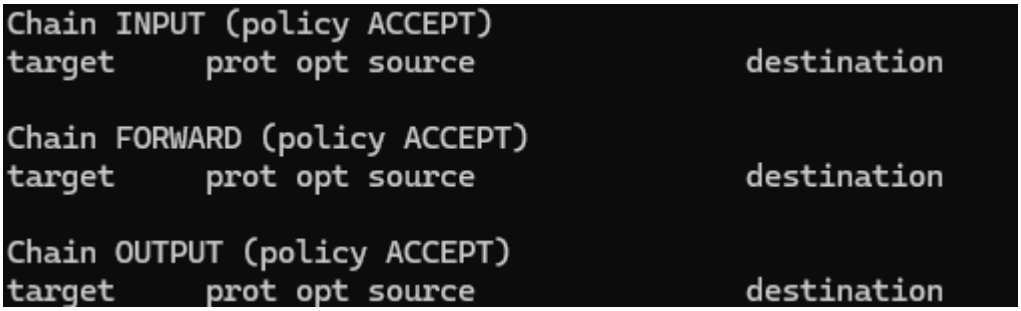
```
Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . : dei.isep.ipp.pt
   IPv6 Address. . . . . . . . . . . : fd1e:2bae:c6fd:1008:d85f:c9fa:63e:1293
   Temporary IPv6 Address. . . . . . : fd1e:2bae:c6fd:1008:dc39:1145:8ce:2a7b
   Link-local IPv6 Address . . . . . : fe80::c921:bd0:8fdf:1a03%10
   IPv4 Address. . . . . . . . . . . : 10.8.86.103
   Subnet Mask . . . . . . . . . . . : 255.255.0.0
   Default Gateway . . . . . . . . . : fe80::ceef:48ff:fe98:efa2%10
                                       10.8.0.1
```

#US6.4.2 - Figura 1: Configurações de Rede

Analisando as configurações, é possível concluir que o IP atualmente configurado pertence à rede **10.8.0.0/16,** que é rede interna do DEI.

2- Em seguida, já na máquina virtual ***Debian***, vou começar por verificar as regras de firewall atualmente ativas, através do comando ***iptables -L***.



#US6.4.2 - Figura 2: Tabela de Regras de Firewall Antes das Configurações

3- Depois vou configurar as regras de ***firewall*** pretendidas:

Esta primeira regra é a responsável por **permitir** que tráfego originado da rede

```
root@debian:/home/asist# iptables -A INPUT -p tcp --dport 4201 -s 10.8.0.0/16 -j ACCEPT
```

#US6.4.2 – Figura 3: Comando Para Regra Firewall

10.8.0.0/16 consiga aceder à porta ***TCP* 4201**.

Esta segunda regra tem como função **bloquear** todo o tráfego para a porta ***TCP* 4201**.

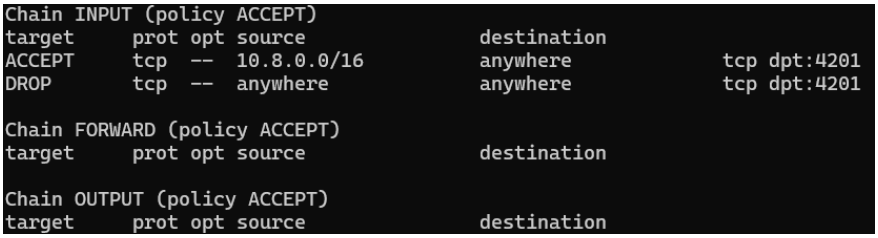
```
root@debian:/home/asist# iptables -A INPUT -p tcp --dport 4201 -j DROP
```

#US6.4.2 - Figura 4: Comando para Regra Firewall

Desta forma, através da combinação destas duas regras, o tráfego originado da rede **10.8.0.0/16** para a porta **4201** será permitido e todo restante tráfego com destino a essa porta será bloqueado.

4- Após a configuração das regras de ***firewall***, utilizo o comando ***iptables-save > /etc/iptables/rules.v4***, que vai guardar as configurações no ficheiro ***rules.v4*** de modo a persistir as configurações, ou seja, em caso de ***reboot*** estas não serão perdidas. O ficheiro ***rules.v4*** é importante porque contém as regras em um formato que pode ser carregado automaticamente durante a inicialização do sistema.

5- Depois, seguido de um **reboot**, através do comando ***iptables -L***, verifica-se que as regras foram guardadas corretamente.



#US6.4.2 - Figura 5: Tabela de Regras de Firewall Depois das Configurações

De notar a ordem com que as regras aparecem, caso estivessem trocadas nenhum tráfego seria permitido, visto que as regras são avaliadas de forma sequencial. Assim que uma regra é correspondida, as regras subsequentes não são verificadas para aquele tráfego.

Resultado:

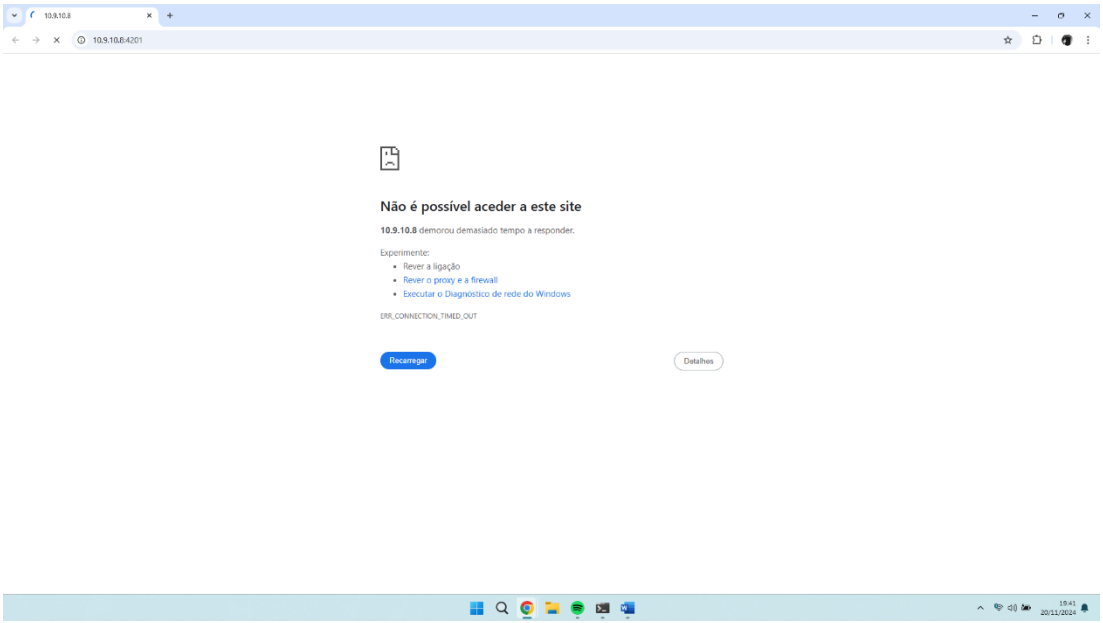
Uma vez que é necessário estar conectado à rede interna do ***DEI*** para conseguir aceder à máquina virtual, para fazer o teste às configurações vou bloquear o ***IP*** da minha conexão ***VPN*** de aceder à porta onde a solução estará disponibilizada, de modo a simular um ***IP*** que não pertence à rede **10.8.0.0/16**. De seguida vou estabelecer outra conexão ***VPN*** de modo a obter um ***IP*** diferente, send este pertencente à rede **10.8.0.0/16**. Neste caso o ***IP*** a ser bloqueado é **10.8.211.3/32**, e então a tabela de regras ***firewall*** (***iptables -L***) ficará assim:

```
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       tcp  --  10.8.211.3           anywhere             tcp dpt:4201
ACCEPT     tcp  --  10.8.0.0/16          anywhere             tcp dpt:4201
DROP       tcp  --  anywhere             anywhere             tcp dpt:4201
```

#US6.4.2 - Figura 6: Tabela de Regras Firewall Para o Teste de Configurações

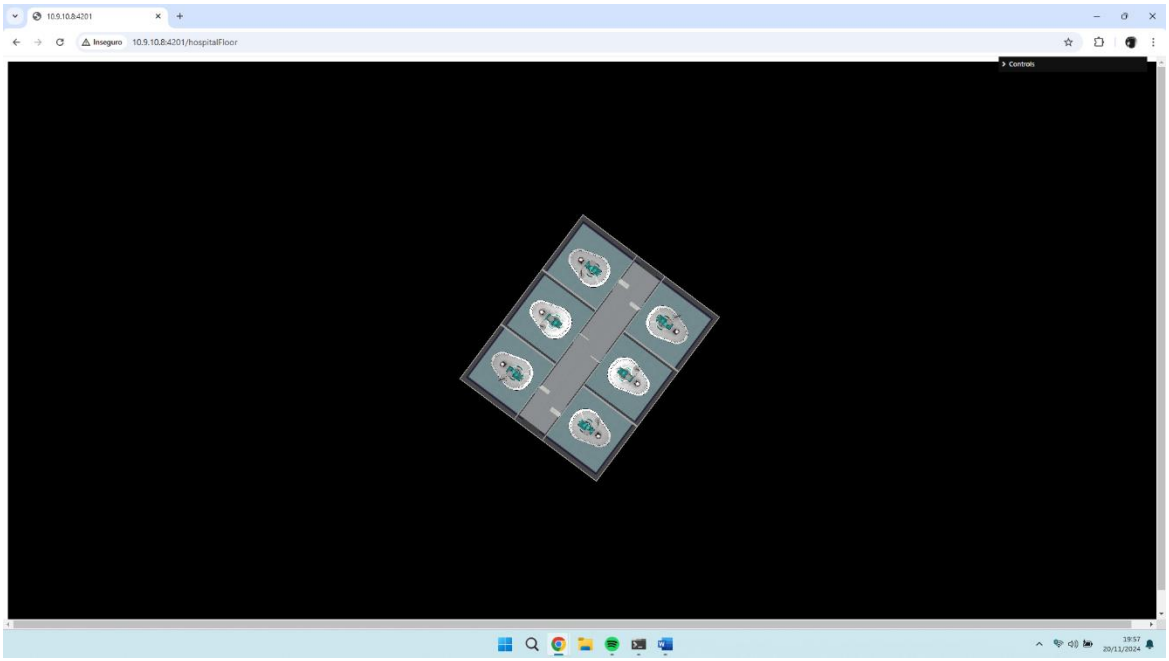
O previsto é que com o ***IP*** bloqueado não consiga aceder à solução e que ao conectar-me novamente à ***VPN*** consiga aceder.

Com a conexão via ***VPN*** utilizando o ***IP*** bloqueado **10.8.211.3/32,** simulando que não pertence à rede **10.8.0.0/16**:



#US6.4.2 - Figura 7: Resultado Obtido Para Um IP Não Pertencente à Rede Interna do DEI

Com a conexão via ***VPN*** utilizando um ***IP*** não bloqueado pertencente à rede **10.8.0.0/16**:



#US6.4.2 - Figura 8: Resultado Obtido Para Um IP Pertencente à Rede Interna do DEI

Desta forma é possível concluir que as configurações de ***firewall*** estão então a funcionar.

User Story #6.4.3

Como administrador do sistema quero que os clientes indicados na *user story* 2 possam ser definidos pela simples alteração de um ficheiro de texto.

Primeiro começamos por cria um ficheiro de texto que conterá os ip´s dos clientes permitidos.

```
PPP adapter deinet.isep.ipp.pt:

   Connection-specific DNS Suffix  . : dei.isep.ipp.pt
   IPv4 Address. . . . . . . . . . . : 10.8.224.127
   Subnet Mask . . . . . . . . . . . : 255.255.255.255
   Default Gateway . . . . . . . . . : 0.0.0.0
```

#US6.4.3 - Figura1 ip da vpn do isep

```
 GNU nano 5.4                                /etc/authorized-clients
10.8.224.47/32
```

#US6.4.3 - Figura2 ficheiro com utilizadores permitidos /etc/authorized-clients

Agora passo a criação do script que é uma solução simples e eficaz para gerenciar permissões de acesso a um serviço específico em um servidor Linux. Ele utiliza o iptables para permitir conexões à porta TCP 4201,22,80 e 443 apenas para endereços IP listados no ficheiro /etc/authorized-clients. A automação reduz a possibilidade de erros e torna o gerenciamento de regras de firewall mais eficiente.

```
  GNU nano 5.4                          /etc/allow_clients.sh *
#!/bin/bash

while IFS= read -r ip; do

    iptables -A INPUT -p tcp --dport 4201 -s $ip -j ACCEPT
 iptables -A INPUT -p tcp --dport 22 -s $ip -j ACCEPT
 iptables -A INPUT -p tcp --dport 80 -s $ip -j ACCEPT
 iptables -A INPUT -p tcp --dport 443 -s $ip -j ACCEPT
        echo "client wuth this $ip was allowed"

done < /etc/authorized-clients
```

#US6.4.3 - Figura3 script /etc/allow_clients.sh

- Porta 22: Permite conexões SSH.
- Porta 80: Permite conexões HTTP.
- Porta 443: Permite conexões HTTPS.
- Porta 4201: Permite o tráfego na porta 4201 usada na user story 2.

```
root@debian:/home/asist# iptables -L --line-numbers
Chain INPUT (policy ACCEPT)
num  target     prot opt source               destination
1    ACCEPT     tcp  --  10.8.0.0/16          anywhere             tcp dpt:4201
2    DROP       tcp  --  anywhere             anywhere             tcp dpt:4201
```

#US6.4.3 - Figura3 iptables antes de Executar o script

```
root@debian:/# ./etc/allow_clients.sh
client wuth this 10.8.224.47/32 was allowed
root@debian:/# iptables -L --line-numbers
Chain INPUT (policy ACCEPT)
num  target     prot opt source               destination
1    ACCEPT     tcp  --  10.8.0.0/16          anywhere             tcp dpt:4201
2    DROP       tcp  --  anywhere             anywhere             tcp dpt:4201
3    ACCEPT     tcp  --  10.8.224.47          anywhere             tcp dpt:4201
4    ACCEPT     tcp  --  10.8.224.47          anywhere             tcp dpt:ssh
5    ACCEPT     tcp  --  10.8.224.47          anywhere             tcp dpt:http
6    ACCEPT     tcp  --  10.8.224.47          anywhere             tcp dpt:https
```

#US6.4.3 - Figura4 iptables depois de executar ./etc/allow_clients.sh

O objetivo do script é garantir que apenas os usuários cujos endereços IP estão listados no arquivo possam aceder a aplicação.

User Story #6.4.4

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

Procedimento: Para realizar esta *user story*, comecei por identificar os riscos que possam comprometer a solução preconizada. Para quantificá-los elaborei uma matriz de risco que permite o cálculo do valor de cada risco através do produto entre a probabilidade e impacto.

Execução:

1º Passo: Identificar dos riscos

- Falhas no desempenho da VPN que podem levar à indisponibilidade de acesso à aplicação;
- Falhas de segurança na VPN que podem causar acessos não autorizados;
- Ataques informáticos;
- Falta de monitoramento adequado;
- Perdas financeiras associadas a falhas, tais como interrupções prolongadas da aplicação;
- Erros dos utilizadores como a inserção incorreta de dados;
- Escassez de pessoas para resolver problemas técnicos rapidamente;
- Divulgação indevida de dados sensíveis dos utilizadores.

2º Passo: Classificar e avaliar os riscos

Riscos	Probabilidade	Impacto	Descrição
Indisponibilidade da VPN	Frequente	Catastrófico	Interrompe agendamento de operações hospitalares
Falhas de segurança	Frequente	Catastrófico	Pode causar exposição a dados sensíveis
Ataques informáticos	Provável	Catastrófico	Pode comprometer a aplicação
Faltas de monitoramento	Remota	Moderado	Dificulta deteção de atividades suspeitas
Perdas financeiras	Provável	Catastrófico	Interrupções críticas podem levar a custos elevados para restaurar a operação
Erros dos utilizadores	Provável	Moderado	Atinge a precisão de dados e agendamento
Escassez de pessoas	Provável	Moderado	Aumenta o tempo de resposta a falhas
Divulgação de dados sensíveis	Frequente	Catastrófico	Afeta a privacidade e pode resultar em sanções legais

#US 6.4.4 – Tabela de riscos, probabilidades de ocorrência e impacto

3º Passo: Elaborar a matriz de risco

Através da matriz de risco, é possível visualizar com mais clareza e organização a probabilidade e o impacto de cada risco.

Probabilidade	Impacto				
		Catastrófico: 4	**Crítico: 3**	**Moderado: 2**	**Marginal: 1**
	Frequente: 5	Indisponibilidade da VPN; Falhas na segurança; Divulgação de dados sensíveis.	-	-	-
	Provável: 4	Ataques informáticos; Perdas financeiras.	-	Erros dos utilizadores; Falta de monitoramento.	-
	Ocasional: 3	-	-	-	-
	Remoto: 2	-	-	Escassez de pessoas.	-
	Improvável:1	-	-	-	-

#US 6.4.4 – Matriz de riscos

4º Passo: Quantificar os riscos

Após realizar uma análise da matriz, é possível quantificar os riscos, ou seja, atribuir-lhes um valor que representa a gravidade de cada risco, permitindo a sua comparação e atribuição de prioridade. O valor do risco é calculado através do produto entre a probabilidade e o impacto.

- High – 20 (4 x 5):
  - A **indisponibilidade da VPN, falhas da segurança, divulgação de dados sensíveis** possuem uma prioridade máxima para mitigação.
- High – 16 (4 x 4):
  - Os **ataques informáticos e as perdas financeiras** representam riscos graves que podem causar danos operacionais e financeiros.
- Serious – 8 (4 x 4):
  - Os **erros de utilizadores e a falta de monitoramento** têm um efeito considerável, mas não menos grave do que os anteriores.
- Medium – 4 (2 x 4):
  - A **escassez de pessoas** é uma situação que pode ser monitorada com o objetivo que seja evitada.

Resultado:

Quantificação do Risco	Identificação do Risco
High (20)	Indisponibilidade da VPN; Falhas de segurança; Divulgação dos dados sensíveis;
High (16)	Ataques informáticos; Perdas financeiras;
Serious (8)	Erros de utilizadores; Falta de monitoramento;
Medium (4)	Escassez de pessoas;

#US 6.4.4 – Tabela da quantificação e identificação dos riscos

User Story #6.4.5

O *Minimum Business Continuity Objective* (MBCO) corresponde ao nível mínimo de serviços ou produtos que é aceitável para a empresa durante uma interrupção das operações, garantindo que os seus objetivos essenciais possam ser atingidos.

A nossa aplicação tem como objetivo gerir a utilização de um bloco operatório de um hospital e possui os seguintes módulos:

SPA (Single Page Application): Este módulo estabelece a ligação entre o utilizador e os restantes módulos;

3D-Visualization-Module: Responsável pela criação de representações gráficas 3D do bloco operatório do hospital;

Backend: Atua como intermediário entre a aplicação e a base de dados;

Planning Module: Gere o agendamento de horários para a utilização das salas de operações, assim como a alocação de funcionários;

Database: Armazena todos os dados da aplicação.

Identificação de módulos essenciais

Entre os cinco módulos descritos, apenas o 3D-Visualization-Module não é considerado essencial, pois a sua funcionalidade centra-se exclusivamente na representação gráfica, que não é crítica para a continuidade das operações da aplicação. No entanto, os outros quatro módulos são extremamente importantes para o funcionamento do sistema, sendo que base de dados assume uma importância ligeiramente superior. A razão para esta distinção deve-se ao facto de que todos os dados críticos da organização estão armazenados nesta, tornando-a assim no núcleo do sistema.

Estratégias para continuidade e recuperação

Base de Dados

A base de dados é o modulo mais importante e, por isso, requer um plano robusto de continuidade. As seguintes estratégias serão implementadas:

- **Backups regulares**: A base de dados será sujeita a cópias de segurança frequentes, garantindo que os dados estão sempre protegidos e disponíveis para recuperação em caso de falha.
- **Testes periódicos**: Realizar testes de recuperação para assegurar que os backups são funcionais e podem ser restaurados rapidamente.
- **Replicação de dados**: Os backups serão enviados para um local externo ao qual tanto a base de dados principal como uma base de dados alternativa tenham acesso, permitindo uma rápida transição para o sistema alternativo em caso de falha grave.
- **Substituição contingencial**: Caso a base de dados principal deixe de estar operacional, uma base de dados alternativa poderá ser incorporada desde que esteja devidamente sincronizada e atualizada com a principal.

A recuperação da base de dados deve ser feita o mais depressa possível, pois todo o sistema depende do seu funcionamento para operar.

Restantes Módulos Essenciais

Os módulos **SPA**, **Backend** e **Planning Module** são também críticos para a continuidade da aplicação. Em caso de falha de qualquer um destes módulos, o plano de recuperação incluirá:

- **Restauro do estado da máquina**: Tentativa inicial de restaurar a funcionalidade do módulo na máquina original.
- **Substituição por máquina alternativa**: Caso o restauro não seja viável, uma máquina alternativa será utilizada para executar o módulo. Neste cenário, será necessário notificar os restantes módulos que comunicam com este sobre a mudança, para que possam redirecionar as interações corretamente.

É também importante notar que a recuperação destes módulos deve também ser feita o mais depressa possível pois o bom funcionamento do hospital não é possível sem eles.

User Story #6.4.6

Procedimento:

Como administrador de sistemas responsável pela implementação de uma estratégia de backup para uma aplicação de um hospital, a necessidade de minimizar o ***RPO (Recovery Point Objective)*** e o ***RTO (Recovery Time Objective)*** é essencial para garantir a continuidade dos serviços, a integridade dos dados dos pacientes e a conformidade com as exigências de disponibilidade dos sistemas médicos. Desta forma, a estratégia que considerei mais eficaz é realizar ***Backups* Totais** todos os domingos pelas 00:00 horas, onde existirá menos atividade na aplicação, uma vez que o tempo necessário para completar o ***Backup* Total** pode ser longo e pode afetar a performance durante o processo. Nos restantes dias da semana vão ser realizados ***Backups* Incrementais** de hora em hora. Estes ***Backups*** de hora em hora permitem reduzir o ***RPO*** para uma hora, o que significa que em caso de falha a perda de dados seria limitada à última hora de alterações. Para um hospital, onde a precisão dos dados e a continuidade do tratamento são vitais, esse ***RPO*** mais baixo é crucial.

A estratégia de ***Backups* Totais semanais** com **Incrementais horários** reduz o ***RTO*,** pois a cada semana é iniciado um novo **Ciclo de *Backups***, garantindo que o ***Backup* Total**, a partir do qual os ***Backups* Incrementais** vão ser feitos, esteja sempre relativamente recente. Caso contrário, se fosse feito apenas um único ***Backup* Total** no início e ***Backups* Incrementais** fossem realizados por um longo período (como um mês ou mais, por exemplo), a recuperação exigiria a restauração do ***Backup* Total** seguido por todos os **Incrementais** acumulados durante esse período, o que aumentaria significativamente o volume de dados a ser restaurado e, consequentemente, o tempo de recuperação necessário (***RTO***).

Execução:

1- Primeiro comecei por criar um ***Script*** responsável por fazer o ***Backup* Total** da Base de Dados, ***Backup*** este que ficará compactado no formato ***.tar.gz***:

```
#!/bin/bash

DB_HOST="vsgate-s1.dei.isep.ipp.pt"
DB_PORT="10712"
DB_USER="root"
DB_PASSWORD="qyDWqxuLLhI6"
DB_NAME="Rui"

date=$(date '+%Y-%V')
dir_name="/backup/$(date '+%Y-%V')"

mkdir -p $dir_name

mysqldump -h $DB_HOST -P $DB_PORT -u $DB_USER -p$DB_PASSWORD $DB_NAME > $dir_name/backup.${date}.sql

tar -czf $dir_name/backup.${date}.total.tar.gz $dir_name/backup.${date}.sql

rm -f $dir_name/backup.${date}.sql

exit 0
```

2- De seguida criei um Script que tem como função realizar o ***Backup* Incremental**:

```bash
#!/bin/bash

DB_HOST="vsgate-s1.dei.isep.ipp.pt"
DB_PORT="10712"
DB_USER="root"
DB_PASSWORD="qyDWqxuLLhI6"
DB_NAME="Rui"
BACKUP_DIR="/backup/total"

date=$(date '+%Y-%m-%d_%H-%M-%S')
dir_name="/backup/$(date '+%Y-%V')"

mysqldump -h $DB_HOST -P $DB_PORT -u $DB_USER -p$DB_PASSWORD --single-transaction --flush-logs $DB_NAME --routines > $dir_name/backup.${date}.sql

tar -czf $dir_name/backup.${date}.incremental.tar.gz -g $dir_name/backup.snap -p $dir_name/backup.${date}.sql

rm -f $dir_name/backup.${date}.sql

exit 0
```

Este ***Script*** exporta o conteúdo da Base de Dados e através do comando ***tar*** compacta-o no formato ***.tar.gz***. O parâmetro ***-g $dir_name/backup.snap*** especifica o uso do arquivo ***snapshot*** para rastrear as mudanças desde o último ***Backup***. No primeiro ***Backup*** é criado o arquivo ***backup.snap*** que contém uma lista de dados incluídos no ***Backup***. Nos ***Backups*** subsequentes o comando ***tar*** verifica o arquivo ***backup.snap*** e só adiciona ao ***Backup*** os arquivos que foram mudados ou criados desde o último ***Backup***. O ***backup.snap*** é essencial para o ***Backup* Incremental**, pois permite que o ***Script*** rastreie apenas as alterações nos dados.

3- Depois, editei o ficheiro ***crontab*** através do comando ***crontab -e*** de forma a executar o Script de ***Backup* Total** à meia-noite de domingo para segunda-feira e o ***Script*** de ***Backup* Incremental** de hora em hora, começando às 00:30 horas.

```
# 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

* * * * * /scripts/editIssue.sh
*/15 * * * * /scripts/deploy_3dModule.sh > /dev/null 2>&1
0 0 * * 0 /scripts/backupTotal.sh
30 * * * * /scripts/backupIncremental.sh
```

Resultado*:*

Os ***Backups*** vão ficar organizados por semana, ou seja, sempre que é corrido o ***Script*** de ***Backup*** **Total** é criada uma pasta relativa à semana correspondente, onde esse mesmo ***Backup*** **Total** e os ***Backups*** **Incrementais** associados vão ficar armazenados. O nome dos ***Backups*** **Totais** será relativo à semana em questão, o nome dos ***Backups*** **Incrementais** será relativo à data e hora em que forem executados.

```
root@debian:/home/asist# ls /backup/2024-47/
backup.2024-11-23_01-34-37.incremental.tar.gz  backup.2024-11-23_05-30-01.incremental.tar.gz  backup.2024-11-23_09-30-01.incremental.tar.gz  backup.2024-47.total.tar.gz
backup.2024-11-23_02-30-01.incremental.tar.gz  backup.2024-11-23_06-30-01.incremental.tar.gz  backup.2024-11-23_10-30-01.incremental.tar.gz  backup.snap
backup.2024-11-23_03-30-02.incremental.tar.gz  backup.2024-11-23_07-30-02.incremental.tar.gz  backup.2024-11-23_11-30-01.incremental.tar.gz
backup.2024-11-23_04-30-01.incremental.tar.gz  backup.2024-11-23_08-30-01.incremental.tar.gz  backup.2024-11-23_12-30-01.incremental.tar.gz
```

Criei também um ***Script*** responsável por restaurar os ***Backups*** automaticamente. Este ***Script*** começa por verificar qual a semana mais recente e de seguida restaura os ***Backups*** pela ordem correta. Começa pelo ***Backup*** **Total**, de seguida organiza os incrementais por ordem crescente da data e depois restaura-os seguindo essa mesma ordem.

```bash
#!/bin/bash

BASE_BACKUP_DIR="/backup"
DB_HOST="vsgate-s1.dei.isep.ipp.pt"
DB_PORT="10712"
DB_USER="root"
DB_PASSWORD="qyDWqxuLLhI6"
DB_NAME="Rui"
REMOTE_HOST="vs427.dei.isep.ipp.pt"
REMOTE_DIR="/backup/total"
SSH_USER="root"
RESTORE_DIR="/backup/2024-47/"


BACKUP_DIR=$(ls -d $BASE_BACKUP_DIR/* | sort | tail -n 1)

TOTAL_BACKUP=$(ls $BACKUP_DIR | grep '.total.tar.gz' | head -n 1)
backup_name=$(basename "$TOTAL_BACKUP" .tar.gz)
sql_file="$BACKUP_DIR/$backup_name.sql"

tar --strip-components=2 -xOzf "$BACKUP_DIR/$TOTAL_BACKUP" > "$sql_file"

mysql -h $DB_HOST -P $DB_PORT -u $DB_USER -p$DB_PASSWORD $DB_NAME < $sql_file

rm $sql_file

INCREMENTAL_BACKUPS=$(ls $BACKUP_DIR | grep '.incremental.tar.gz' | sort)
if [ -n "$INCREMENTAL_BACKUPS" ]; then
  for backup in $INCREMENTAL_BACKUPS; do
    backup_name=$(basename "$backup" .tar.gz)
    sql_file="$BACKUP_DIR/$backup_name.sql"
    tar --strip-components=2 -xOzf "$BACKUP_DIR/$backup" > "$sql_file"

    mysql -h $DB_HOST -P $DB_PORT -u $DB_USER -p$DB_PASSWORD $DB_NAME < $sql_file

    rm $sql_file
  done
fi
```

User Story #6.4.7

Como administrador do sistema quero definir uma pasta pública para todos os utilizadores registados no sistema, onde podem ler tudo o que lá for colocado.

Começamos pela criação de um novo diretório ***/etc/public_folder*** que será a pasta partilhada e passo por dar apenas permissão de leitura aos utilizadores registados no sistema.

```
root@debian:/# mkdir /etc/public_folder
root@debian:/home/asist# chmod 755 /etc/public_folder/
root@debian:/home/asist# ls -ld /etc/public_folder/
drwxr-xr-x 2 root root 4096 Nov 24 18:09 /etc/public_folder/
```

#US6.4.7 - Figura 1: Criação do diretório partilhado e definição de permissões

Uso **chmod 750** para dar as seguintes permissões:

- 7 para o proprietário: Permissões de leitura (r), escrita (w) e execução (x).
- 5 para o grupo: Permissões de leitura (r) e execução (x), mas sem permissão de escrita.
- 5 para outros usuários: Permissões de leitura (r) e execução (x), mas sem permissão de escrita.

```
  GNU nano 5.4                    /etc/public_folder/teste_view *
Pasta compartilhada
users apenas podem ler.
```

#US6.4.7 - Figura 2 : criação do ficheiro teste_view como owner

Testando a ligação como um utilizador do sistema:

Para este caso usamos o user teto após o login tento aceder o ficheiro ***teste_view***. Como resultado consigo aceder a pasta e ler o conteúdo do ficheiro:

```
sandr@DESKTOP-JFIRAKG:~$ ssh teto@10.9.10.8
Machine: debian
Logged-in users: 6
Current time: 18:29
teto@10.9.10.8's password:

Last login: Sun Nov 24 18:28:17 2024 from 10.8.224.114
Could not chdir to home directory /home/teto: No such file or directory
-e
##########################

Bem-vindo, teto!
Data: Sunday, 24 de November de 2024
Hora: 18:30
-e
##########################

$ cd /etc/public_folder
$ pwd
/etc/public_folder
$ cat teste_view
Pasta compartilhada
users apenas podem ler.
```

#US6.4.7 - Figura 4: Leitura de um ficheiro na pasta partilhado como um utilizador do sistema

```
$ touch tetoarquivo
touch: cannot touch 'tetoarquivo': Permission denied
$ pwd
/etc/public_folder
```

#US6.4.7 - Figura 3: Tentativa de criação de um ficheiro na pasta partilhada

User Story #6.4.8

Como administrador do sistema quero obter os utilizadores com mais de 3 tentativas de acesso incorretas.

Procedimento: Para resolver esta *user story*, criei um ficheiro que guardasse apenas as mensagens *logs* vindas do *sshd* executei um comando *grep* para filtrar esse ficheiro e obter os utilizadores com mais de 3 tentativas falhadas.

Execução:

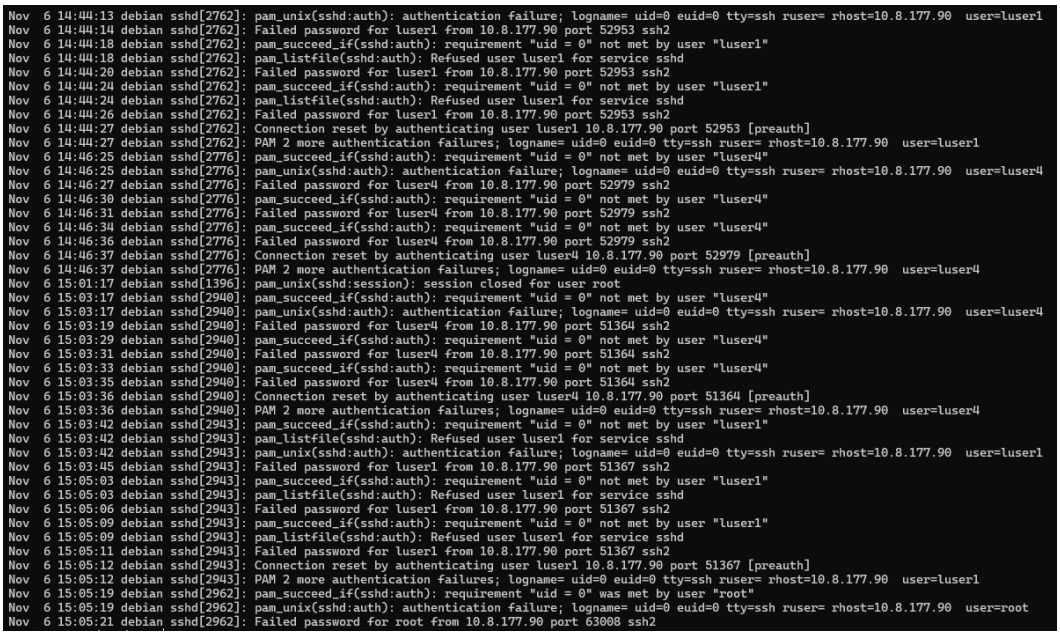
1º Passo: Criar o ficheiro */etc/rsyslog.d/sshd.conf* com o seguinte conteúdo

```
  GNU nano 5.4
if $programname == 'sshd' then /var/log/sshd.log
```

#US6.4.8- Conteúdo do ficheiro /etc/rsyslog.d/sshd.conf

As mensagens de *logs* vindas do *ssh* ficarão guardadas no ficheiro */var/log/sshd.log*.

2º Passo: Executar o comando *sudo service rsyslog restart* para reiniciar o *rsyslog*.



#US6.4.8- Conteúdo do ficheiro ssh.log

3º Passo: Executar o comando ***grep "Failed password" /var/log/sshd.log | awk '{print $(NF-5)}' | uniq -c | awk '$1 > 3 {print $2}'*** de modo a obter os utilizadores com mais de 3 tentativas de acesso falhadas.

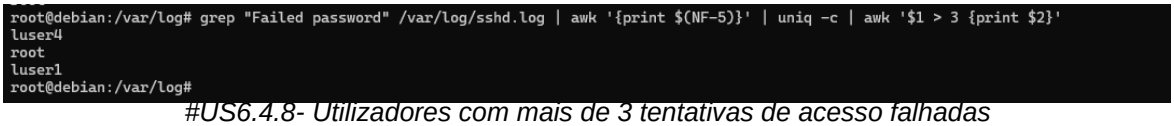
grep "Failed password" /var/log/sshd.log : filtra o ficheiro para encontrar as linhas com o texto “Failed password” que representam tentativas de acesso falhadas.

awk '{print $(NF-5)}' : extrai a quinta palavra a partir do final (NF-5) de cada linha filtrada anteriormente. Geralmente, essa posição corresponde ao endereço *IP* de quem tentou a autenticação falhada.

uniq -c : conta o número de ocorrências de cada endereço *IP*, ou seja, cada tentativa falhada por cada endereço de IP.

awk '$1 > 3 {print $2}' : filtra apenas os *IPs* com mais de 3 falhas de login.

Resultado:



#US6.4.8- Utilizadores com mais de 3 tentativas de acesso falhadas