

RELATÓRIO ASIST – SPRINT 2

24 NOVEMBRO

Professor: Rui Filipe Nogueira Marques (RFM)

Da autoria de: Grupo 15, 3DC



Índice:

User Story 1..... 2

User Story 2..... 6

User Story 3..... 8

User Story 4..... 11

User Story 5..... 13

User Story 6..... 15

User Story 7..... 18

User Story 8..... 19

User Story 1

Enunciado: 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.

Aluno Responsável: Vasco Sousa (1221700)

Solução: De forma a solucionar esta User Story, começamos por instalar todos os packages necessários, sendo estes o **git**, o **dotnet** e o **cron**. Após esta instalação geramos uma **chave ssh** para conseguirmos aceder ao repositório sem ser necessária a autenticação a cada pedido efetuado. A chave foi gerada através do comando **“ssh-keygen -t rsa -b 4096 -C your_email@example.com”** e após esta ser gerada demos as permissões necessárias ao **root** com o comando **“chmod 600 ~/.ssh/id_rsa”** de maneira a este ser o único com acesso à mesma. Para além disto, foi também necessário adicionar a chave ao **ssh-agent** através do comando **“ssh-add ~/.ssh/id_rsa”**.

Por fim, e já no **Github**, adicionamos esta mesma chave na secção **“Deploy Keys”** do nosso repositório e após isso foi executado o comando **“git clone git@github.com:your-username/your-repository.git”** na pasta **“/root/apps/HospitalManagementAppBE”** que foi criada por nós através do comando **“mkdir”**.

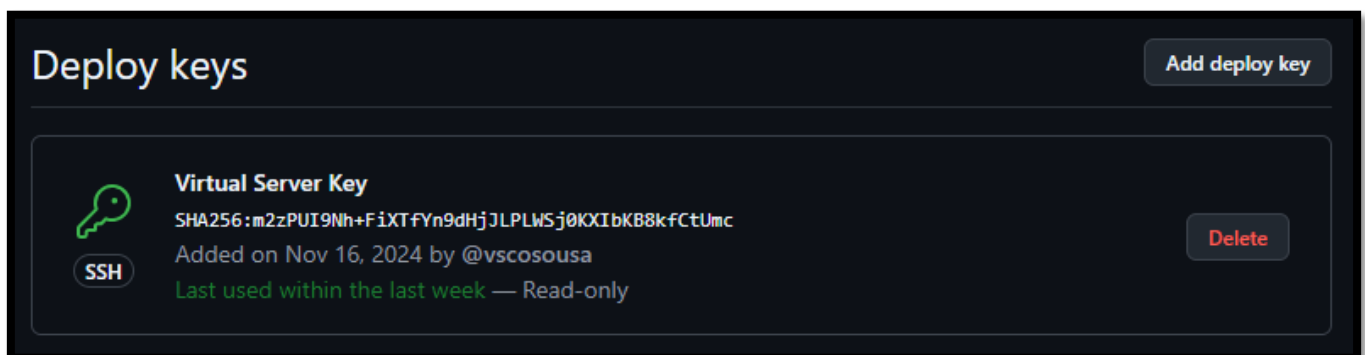


Figura 1 - Deploy Keys Github Repo

Terminado todo este processo de adicionar o projeto ao nosso Virtual Server, e ainda na pasta mencionada anteriormente, executamos o comando **“touch deployBE.sh”**, para criar o nosso ficheiro script, seguido do comando **“nano deployBE.sh”**, que nos permite editar o ficheiro que criamos.

Já dentro do **nano editor** adicionamos vários comandos com o objetivo de verificar a integridade do nosso projeto de **Backend**. O formato final e os comandos foram os seguintes:

```
GNU nano 5.4 /root/apps/HospitalManagementAppBE/deployBE.sh
#!/bin/bash
start_time=$(date +%s)

# Set working directory for HospitalManagementAppBE project
cd /root/apps/HospitalManagementAppBE/LAPR5_3DC_G15 || { echo "Directory not found! Exiting."; exit 1; }

echo "Running in day: $(date)"

echo -e '\n----- Pulling latest code from repository -----'\n'

# Pull the latest changes from the repository
git pull origin main || { echo "Git pull failed! Exiting."; exit 1; }

# Navigate to the Backoffice directory
cd Backoffice || { echo "Backoffice directory not found! Exiting."; exit 1; }

echo -e '\n----- Cleaning previous build outputs -----'\n'

# Clean previous build outputs
dotnet clean || { echo "Dotnet clean failed! Exiting."; exit 1; }

echo -e '\n----- Restoring project dependencies -----'\n'

# Restore project dependencies
dotnet restore || { echo "Dotnet restore failed! Exiting."; exit 1; }

echo -e '\n----- Building project -----'\n'

# Build the project without restoring dependencies again
dotnet build --no-restore || { echo "Dotnet build failed! Exiting."; exit 1; }

echo -e '\n----- Running tests -----'\n'

# Run tests without building the project again
dotnet test --no-build || { echo "Dotnet test failed! Exiting."; exit 1; }

end_time=$(date +%s)
total_time=$((end_time - start_time))
minutes=$((total_time / 60))
seconds=$((total_time % 60))

echo -e "\nTotal time: $minutes minutes and $seconds seconds"

echo -e '\n----- Script completed successfully -----'\n'
```

Figura 2 - Script Deploy Backend

Comando	Função
<code>cd /root/apps/HospitalManagementAppBE/LAPR5_3DC_G15</code>	Altera o diretório de trabalho para onde o projeto está localizado.
<code>\$(date)</code>	Extraí a data e hora atual.
<code>git pull origin main</code>	Extraí todas as alterações efetuadas no repositório.
<code>cd Backoffice</code>	Posiciona o utilizador na pasta desejada.
<code>dotnet clean</code>	Remove os ficheiros criados em builds anteriores.
<code>dotnet restore</code>	Restaura e instala as dependências do projeto.
<code>dotnet build --no-restore</code>	Compila o projeto sem restaurar as dependências
<code>dotnet test --no-build</code>	Executa os testes sem compilar novamente o projeto.

Tabela 1 - Comandos utilizados no Script

Após a configuração deste ficheiro foram dadas as permissões necessárias para este ser executável através do comando “**chmod +x deployBE.sh**”. Por fim definimos uma regra no Crontab através do comando “**crontab -e**” onde adicionamos a seguinte linha:

```
0 3 * * * ./apps/HospitalManagementAppBE/deployBE.sh >> ./logs/logfile_$(date +%Y-%m-%d).log 2>&1
```

Figura 3 - Linha Adicionada ao Crontab

A linha de configuração no **crontab** indica que o script será executado diariamente às **3:00 da manhã**, preservando o output da execução num arquivo de **log** único, gerado após a finalização do script e armazenado na pasta “**/root/logs**” que foi criada por nós. Esse horário foi escolhido por ter menor afluência de utilizadores, reduzindo o impacto no sistema.

```
root@vs781:~/logs# cat logfile_2024-11-18.log
Running in day: Mon 18 Nov 2024 03:00:01 AM WET

----- Pulling latest code from repository -----

hint: Pulling without specifying how to reconcile divergent branches is
hint: discouraged. You can squelch this message by running one of the following
hint: commands sometime before your next pull:
hint:
hint:   git config pull.rebase false  # merge (the default strategy)
hint:   git config pull.rebase true   # rebase
hint:   git config pull.ff only       # fast-forward only
hint:
hint: You can replace "git config" with "git config --global" to set a default
hint: preference for all repositories. You can also pass --rebase, --no-rebase,
hint: or --ff-only on the command line to override the configured default per
hint: invocation.
From github.com:vscosousa/LAPR5_3DC_G15
 * branch          main      -> FETCH_HEAD
    0ce01eb..21c55af main     -> origin/main
Updating 0ce01eb..21c55af
Fast-forward
 Angular-View/package-lock.json      | 19499 ++++++-----
 Angular-View/package.json           | 5 +-
 .../create-staff/create-staff.component.html | 2 +-
 .../create-staff/create-staff.component.scss | 15 +-
 .../create-staff/create-staff.component.ts   | 4 +-
 .../src/app/Components/login/login.component.html | 72 +-
 .../src/app/Components/login/login.component.scss | 8 +-
 .../src/app/Components/login/login.component.ts | 75 +-
 .../Components/register/register.component.html | 4 -
 .../Components/register/register.component.spec.ts | 7 +-
 .../app/Components/register/register.component.ts | 21 +-
 .../search-staffs/search-staffs.component.html | 35 +-
 .../search-staffs/search-staffs.component.scss | 64 +-
 .../search-staffs/search-staffs.component.ts   | 4 +-
 .../update-staff/update-staff.component.html | 2 +-
 .../update-staff/update-staff.component.scss   | 20 +-
 .../update-staff/update-staff.component.ts     | 7 +-
 .../availability-modal.component.html | 23 -
 .../availability-modal.component.scss         | 92 -
 .../availability-modal.component.spec.ts      | 7 -
 .../availability-modal.component.ts           | 59 -
 .../view-availability.component.html          | 79 +-
 .../view-availability.component.scss          | 171 +-
 .../view-availability.component.ts            | 130 +-
 .../src/app/Interceptors/auth.interceptor.ts  | 6 +-
 Angular-View/src/app/Services/login.service.ts | 23 +-
 Angular-View/src/index.html             | 1 +
 Backoffice/Controllers/UserController.cs     | 46 +-
 Backoffice/DDNetCore.csproj               | 1 +
```

Figura 4 - Output guardado no ficheiro de log

Através do output armazenado nos **logs**, conseguimos identificar e resolver com mais facilidade possíveis erros que ocorram no módulo. Estes **logs** permitem consultar informações detalhadas sobre o comportamento do sistema e as execuções do **plano de testes**, ajudando na detecção de falhas específicas, problemas de desempenho e eventos inesperados. Com esta documentação centralizada e acessível, o administrador consegue agir proactivamente na manutenção e otimização do módulo, assegurando que o ambiente de produção opera de forma eficiente e sem interrupções indesejadas.

Por exemplo, através do output armazenado no log do dia **18/11/2024 às 03:00h**, conseguimos detetar uma falha nos testes do módulo de **Backend**, que indicava uma inconsistência na comunicação entre as diferentes componentes existentes no mesmo. A análise detalhada deste **log** permitiu identificar rapidamente o problema, possibilitando a correção antes que afetasse outros módulos ou o ambiente de produção. Desta forma, o uso de **logs** facilita uma resposta rápida e eficaz, mantendo o sistema estável e funcionando conforme o esperado.

```
----- Running tests -----  
  
Test run for /root/apps/HospitalManagementAppBE/LAPR5_3DC_G15/Backoffice/bin/Debug/net8.0/DDNetCore.dll (.NETCoreApp,Version=v8.0)  
VSTest version 17.11.1 (x64)  
  
Starting test execution, please wait...  
A total of 1 test files matched the specified pattern.  
[xUnit.net 00:01:34.35] DDDSample1.Tests.Staffs.UnitTests.StaffTests.AddAvailabilitySlot_ExistingSlot_ShouldThrowException [FAIL]  
[xUnit.net 00:01:35.55] DDDSample1.Tests.Staffs.UnitTests.StaffTests.RemoveAvailabilitySlot_NonExistingSlot_ShouldThrowException [FAIL]  
Failed DDDSample1.Tests.Staffs.UnitTests.StaffTests.AddAvailabilitySlot_ExistingSlot_ShouldThrowException [2 s]  
Error Message:  
Assert.Throws() Failure  
Expected: typeof(DDDSample1.Domain.Shared.BusinessRuleValidationException)  
Actual: (No exception was thrown)  
Stack Trace:  
at DDDSample1.Tests.Staffs.UnitTests.StaffTests.AddAvailabilitySlot_ExistingSlot_ShouldThrowException() in /root/apps/HospitalManagementAppBE/LAPR5_3DC_G15/Backoffice/Tests/StaffTests.cs:line 176  
at System.RuntimeMethodHandle.InvokeMethod(Object target, Void** arguments, Signature sig, Boolean isConstructor)  
at System.Reflection.MethodBaseInvoker.InvokeWithNoArgs(Object obj, BindingFlags invokeAttr)  
Failed DDDSample1.Tests.Staffs.UnitTests.StaffTests.RemoveAvailabilitySlot_NonExistingSlot_ShouldThrowException [60 ms]  
Error Message:  
Assert.Throws() Failure  
Expected: typeof(DDDSample1.Domain.Shared.BusinessRuleValidationException)  
Actual: (No exception was thrown)  
Stack Trace:  
at DDDSample1.Tests.Staffs.UnitTests.StaffTests.RemoveAvailabilitySlot_NonExistingSlot_ShouldThrowException() in /root/apps/HospitalManagementAppBE/LAPR5_3DC_G15/Backoffice/Tests/StaffTests.cs:line 213  
at System.RuntimeMethodHandle.InvokeMethod(Object target, Void** arguments, Signature sig, Boolean isConstructor)  
at System.Reflection.MethodBaseInvoker.InvokeWithNoArgs(Object obj, BindingFlags invokeAttr)  
  
Failed! - Failed: 2, Passed: 173, Skipped: 0, Total: 175, Duration: 2 m 23 s - DDNetCore.dll (net8.0)  
Dotnet test failed! Exiting.  
root@vs781:~/logs# |
```

Figura 5 - Erro nos Testes do Backend

User Story 2

Pedido: Como administrador do sistema, quero apenas clientes na rede interna DEI (cabeada ou via VPN) para poder aceder à solução.

Aluno Responsável: João Pereira (1211503)

Solução: O primeiro passo desta User Story seria entender qual seria o “range” de ips da rede interna DEI. Após uma breve pesquisa concluiu-se que a rede DEI **começa a partir do IP: 10.8.0.0 e termina no IP: 10.8.255.255**

Com esta informação, podemos passar à segunda fase da user story. Para podermos limitar apenas o acesso aos clientes na rede interna DEI, teríamos de criar um script, onde sejam criadas regras nas cadeias da firewall.

Foi criado o script: “*scriptfirewall.sh*” com o seguinte conteúdo:



```
GNU nano 5.4 scriptfirewall.sh
#!/bin/bash

iptables -A INPUT -s 10.8.0.0/16 -j ACCEPT      #permite clientes na rede do DEI (10.8.0.0)
iptables -A INPUT -p tcp --dport 22 -j ACCEPT  #permite conexões ssh
iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT #permite conexões estabelecidas
iptables -P INPUT DROP #Bloqueia o resto do trafego para esta rede
```

Figura 6 - script onde foram criadas as regras

Vamos passar à explicação de cada uma das regras definidas:

iptables -A INPUT -s 10.8.0.0/16 -j ACCEPT

Esta regra apenas permite tráfego num intervalo de ips

Comando	Definição
iptables	Comando principal usado para configurar e gerenciar regras de firewall.
-A	Adiciona (append) uma nova regra ao final da cadeia especificada
INPUT -	Nome da cadeia que trata o tráfego destinado aos endereços de nó.
-s	Define o IP de origem.
10.8.0.0/16	Intervalo de ip's que podem aceder ao sistema: 10.8.0.0 – 10.8.255.255
-j ACCEPT	Define a ação que será tomada para pacotes que correspondem à regra. Neste caso deixa passar o pacote.

iptables -A INPUT -p tcp --dport 22 -j ACCEPT

Esta regra permite conexões SSH na porta padrão.

Comando	Definição
-p tcp	Especifica que a regra se aplica ao protocolo TCP.
-dport 22	Filtra pacotes destinados à porta 22 (usada por padrão para conexões SSH).

iptables -A INPUT -m conntrack --ctstate ESTABLISHED, RELATED -j ACCEPT

Esta regra aceita pacotes de entrada que fazem parte de conexões existentes ou relacionadas.

Comando	Definição
-m conntrack	Utiliza o módulo de rastreamento de conexões para analisar o estado das conexões.
-ctstate	Cria uma regra baseada num estado de conexão
ESTABLISHED	Permite pacotes pertencentes a conexões já estabelecidas.
RELATED	Permite pacotes relacionados a conexões já existentes.

iptables -P INPUT DROP

Define a política padrão da cadeia INPUT para o resto do tráfego. Neste caso bloqueia.

Comando	Definição
-P	Define a política padrão de uma cadeia.
INPUT	Refere-se ao tráfego de entrada.
DROP	Bloqueia todos os pacotes que não correspondam a regras explícitas.

Por fim é necessário dar permissões para executar ao script. Foi utilizado o comando:

```
root@vs781:~# chmod +x scriptfirewall.sh
```

Figura 7- Comando utilizado para dar permissão de executar ao script

De seguida corremos o script e podemos verificar as alterações no iptables com o seguinte comando:

```
root@vs781:~# iptables -L -v -n
```

Figura 8- comando iptables -L -v -n

```
root@vs781:~# iptables -L -v -n
Chain INPUT (policy DROP 16 packets, 1675 bytes)
  pkts bytes target    prot opt in     out     source               destination
   10   688 ACCEPT    all  --  *      *       10.8.0.0/16          0.0.0.0/0
    0     0 ACCEPT    tcp  --  *      *       0.0.0.0/0            0.0.0.0/0          tcp dpt:22
    0     0 ACCEPT    all  --  *      *       0.0.0.0/0            0.0.0.0/0          ctstate RELATED,ESTABLISHED

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
  pkts bytes target    prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 10 packets, 920 bytes)
  pkts bytes target    prot opt in     out     source               destination
```

Figura 9 – verificação das regras definidas no script no iptables

Podemos verificar que a cadeia INPUT só aceita tráfego vindo da rede DEI.

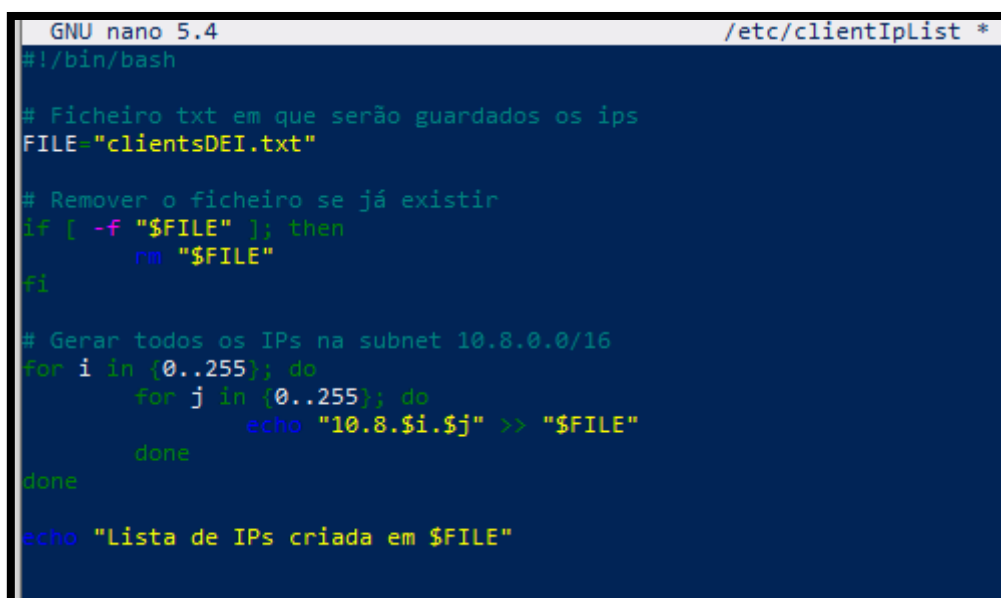
User Story 3

Pedido: Como administrador do sistema quero que os clientes indicados na User Story 2 possam ser definidos pela simples alteração de um ficheiro de texto

Aluno Responsável: Jack Pinheiro (1120419)

Solução: Para implementarmos esta User Story, temos de criar um script que preencha um ficheiro de texto para que contenha todos os clientes definidos na User Story 2 (IP range: 10.8.0.0/16). A segunda parte da tarefa é configurar o **iptables** para que só permita o acesso desses mesmos clientes da rede interna do DEI (contidos no ficheiro de texto), cablada e via VPN.

Iniciamos a tarefa com o comando **nano /etc/clientIpList** onde iremos criar o script.

A screenshot of a terminal window with a dark blue background. The title bar at the top shows 'GNU nano 5.4' on the left and '/etc/clientIpList *' on the right. The script content is as follows:

```
#!/bin/bash

# Ficheiro txt em que serão guardados os ips
FILE="clientsDEI.txt"

# Remover o ficheiro se já existir
if [ -f "$FILE" ]; then
    rm "$FILE"
fi

# Gerar todos os IPs na subnet 10.8.0.0/16
for i in {0..255}; do
    for j in {0..255}; do
        echo "10.8.$i.$j" >> "$FILE"
    done
done

echo "Lista de IPs criada em $FILE"
```

Figura 10 – Criação do script para adicionar os clientes da rede interna do DEI

Após a criação do script, é necessário torná-lo executável com o comando **chmod +x /etc/clientIpList** e correr o script. O comando **chmod** permite modificar permissões e **+x** adiciona permissões para executar ao script. Verificar se o ficheiro .txt inclui todos os IPs a começar em **10.8.0.0** até ao **10.8.255.255**.

```

root@vs781:~# /etc/clientIpList
Lista de IPs criada em clientsDEI.txt
root@vs781:~# head -10 clientsDEI.txt
10.8.0.0
10.8.0.1
10.8.0.2
10.8.0.3
10.8.0.4
10.8.0.5
10.8.0.6
10.8.0.7
10.8.0.8
10.8.0.9
root@vs781:~#

```

Figura 11 1112 – Execução do script e visualização dos primeiros 10 IPs do ficheiro clientsDEI.

Entramos na última parte da User Story que é configurar o **iptables** para que só permita o acesso aos clientes contidos nesse ficheiro .txt criado anteriormente.

```

GNU nano 5.4 /etc/iptables_config *
#!/bin/bash

# Caminho do ficheiro .txt
LIST_IP="clientsDEI.txt"

# Começar com uma tabela de regras vazia
iptables -F

# Defaults
iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -P OUTPUT ACCEPT

# Permitir tráfego na interface loopback, usada para comunicação interna
iptables -A INPUT -i lo -j ACCEPT

# Permitir tráfego pela interface primária de Ethernet (eth0)
iptables -A INPUT -i eth0 -j ACCEPT

# Permitir SSH (Port 22)
iptables -A INPUT -p tcp --dport 22 -j ACCEPT

# Permitir outgoing requests
iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT

# Permitir tráfego pelos endereços de IP do ficheiro .txt
if [[ -f "$LIST_IP" ]]; then
    while IFS= read -r ip; do
        iptables -A INPUT -s "$ip" -j ACCEPT
    done < "$LIST_IP"
else
    echo "Error: List of IPs file not found!"
    exit 1
fi

# Negar o restante tráfego
iptables -A INPUT -j DROP

# Guardar as regras iptables
iptables-save > /etc/iptables/rules.v4

echo "Regras iptables aplicadas com sucesso"

```

Figura 1132 – Script para a criar as regras do iptables

Por fim, tornamos o script novamente executável com **chmod +x /etc/iptables_config** e corremos o script para darmos como terminada a User Story.

User Story 4

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

Aluno Responsável: Jack Pinheiro (1120419)

Solução: A identificação e quantificação dos riscos associados à solução proposta foram realizadas com base numa análise detalhada dos possíveis cenários adversos. Este processo considerou fatores como a complexidade técnica, dependências externas, conformidade com regulamentações e potenciais desafios operacionais. Esta abordagem permite priorizar as áreas críticas e definir estratégias de mitigação apropriadas, assegurando maior robustez e eficiência na implementação da solução.

Possibilidade	Probabilidade	Impacto	Risco
Leak/ Data Breach que compromete os dados sensíveis dos pacientes/staff	2	4	8
Falha de energia que afete o sistema e agendamento de cirurgias	2	3	6
Interdição de acesso ao sistema devido a um ciberataque	2	4	8
Acesso não autorizado ao sistema (dados sensíveis)	2	4	8
Perda de dados de pacientes devido a falha de backup	3	4	12
Impedimento do acesso autorizado devido a falhas do serviço VPN	2	3	6
Acesso não autorizado na pasta pública devido a permissões incorretas	2	3	6
Erros/Atrasos nas atualizações, deixando o sistema exposto possíveis vulnerabilidades de segurança	2	3	6

Tabela 1415 - Matriz de risco referente a situações/possibilidades encontradas

A escala da probabilidade utilizada foi a seguinte:

- Improvável (1)
- Remoto (2)
- Ocasional (3)
- Frequente (4)

Já a escala do Impacto foi a seguinte:

- Marginal (1)
- Moderado (2)
- Crítico (3)
- Catastrófico (4)

A junção destas duas escalas (Probabilidade x Impacto) permitem determinar o risco, sendo este representado numa escala que pode ir de 1 a 12, sendo 12 considerado o risco máximo e 1 o risco mínimo.

Podemos então concluir que a perda de dados de pacientes devido a falhas de backup tem o maior risco perante todas as outras situações pois poderá ter impacto direto na saúde do paciente. *Leaks* ou *Data Breaches*, acessos não autorizados ao sistema e perda de acesso ao sistema também têm um risco elevado pois têm o mesmo nível de impacto, mas com menor probabilidade de acontecer.

User Story 5

Pedido: Como administrador do sistema quero que seja definido o MBCO (Minimum Business Continuity Objective) a propor aos stakeholders.

Aluno Responsável: Vasco Sousa (1221700)

Solução: O MBCO (Minimum Business Continuity Objective) estabelece o nível mínimo de operação que deve ser mantido durante uma possível interrupção na infraestrutura. Por outras palavras, define os serviços essenciais que precisam de ser garantidos para que a organização continue a alcançar os seus objetivos, mesmo em situações de desastre.

Para definir o MBCO, é necessário ter uma base sólida, que se apoie nos objetivos da organização e nos serviços essenciais para os atingir. Além disso, é fundamental avaliar o impacto potencial sobre esses serviços e o tempo necessário para a sua recuperação, seja ela total ou parcial.

No âmbito da nossa solução, cujo objetivo é fornecer um sistema de gestão de recursos e de marcação de cirurgias, a aplicação encontra-se dividida em diferentes módulos, sendo eles os seguintes:

- Planeamento e otimização de cirurgias;
- Visualização 3D;
- GDPR;
- Backoffice Web App;
- Business Continuity Plan (BCP).

Com o desenvolvimento desta aplicação pretende-se que os utilizadores, dependendo das suas permissões, tenham a possibilidade de gerir cirurgias, contas e utilizadores, obter o melhor planeamento de cirurgias possível e navegar num modelo 3D do hospital. Devido à natureza do sistema informático, a nossa aplicação está suscetível a falhas, tais como problemas de rede, falhas de energia, avarias de componentes físicos e perdas de dados (ataque ou falha).

Perante estes desafios e uma vez que estamos a conceber uma solução para uma área que se envolve diretamente com vidas, qualquer erro pode ser fatal. Assim, cabe a nós ultrapassar os desafios e garantir que os serviços críticos e o nível mínimo de funcionamento (conforme o MBCO) estejam assegurados.

Para assegurar a continuidade mínima, os módulos essenciais, como o Planeamento e Otimização de Cirurgias e o GDPR, devem ser priorizados em situações de interrupção, pois afetam diretamente a segurança e privacidade dos dados dos pacientes e a continuidade dos serviços cirúrgicos.

Em caso de falhas nos serviços de base de dados, é necessário um plano preventivo, como backups diários e redundância de dados, garantindo uma recuperação rápida e a minimização da perda de dados. Além disso, é essencial contar com uma equipa de especialistas capacitada para agir imediatamente e restaurar os serviços críticos conforme o RTO (Recovery Time Objective) e o RPO (Recovery Point Objective) definidos. Esses tempos devem ser estabelecidos em alinhamento com o MBCO, para assegurar que os

serviços críticos sejam restaurados dentro de prazos que minimizem o impacto na operação e nos pacientes.

Por fim, a monitorização constante e a manutenção dos componentes físicos e da infraestrutura de rede são essenciais para antecipar e mitigar potenciais falhas, garantindo que o nível mínimo de operação definido pelo MBCO seja mantido de forma confiável.

User Story 6

Pedido: 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).

Aluno Responsável: João Pereira (1211503)

Solução: A estratégia que foi implementada para diminuir o **RPO** (Recovery Point Objective) e o **WRT** (Work Recovery Time), foi **fazer um backup da base de dados** todos os dias às 2 da manhã. A escolha do horário para realizar o backup foi feita com base na baixa atividade de utilizadores às 2 da manhã. Este horário minimiza o impacto no desempenho do sistema, já que há menos carga de tráfego e transações na base de dados. Além disso, o deploy do backend ocorre às 3 da manhã, e o backup realizado uma hora antes oferece uma janela de segurança caso o deploy apresente falhas.

Detalhes da Estratégia:

- **Backup diário às 2 AM:** O backup será executado automaticamente todas as noites às 2 AM. O backup é guardado localmente, assim diminuindo o WRT.
- **Verificação de Integridade:** Após o backup, uma verificação de integridade será realizada para garantir que os dados podem ser restaurados sem problemas, minimizando o risco de corrupção de dados durante o processo.
- **Automatização de Recuperação:** Em caso de erro durante o deploy, a base de dados poderá ser restaurada automaticamente para o ponto de backup anterior. Isto reduz significativamente o WRT.
- **Monitoramento e Alertas:** O processo de backup será monitorado e, caso ocorra algum erro durante o backup ou restauração, um alerta será gerado para a equipa para tomar as ações necessárias.

Conclusão: Esta estratégia foi pensada para garantir a segurança da base de dados e minimizar o impacto de possíveis falhas. A escolha do horário e a automatização do processo reduzem ao máximo o tempo de WRT e o RPO, assegurando que o sistema possa ser restaurado rapidamente em caso de falhas, sem prejudicar a experiência dos utilizadores. Considerando que a aplicação lida com dados sensíveis de utilizadores, esta solução oferece a melhor proteção contra a perda de dados, garantindo a continuidade e segurança do serviço.


```
#!/bin/bash

# dados da nossa base de dados
SQL_SERVER="vs380.dei.isep.ipp.pt"
SQL_USER="sa"
SQL_PASSWORD="hWmCG+RMJQ==Xa5"
SQL_DB="SqlDb"
BACKUP_DIR="/var/opt/mssql/data"
LOG_FILE="/var/log/sql_backup.log"

# gerar o backup com a data
BACKUP_FILE="$BACKUP_DIR/HospitalDBBackup_$(date +%Y-%m-%d_%H-%M-%S).bak"

# correr o comando para fazer backup
echo "$(date +%Y-%m-%d %H:%M:%S)" - Starting backup for database: $SQL_DB" | tee -a $LOG_FILE
/opt/mssql-tools/bin/sqlcmd -S $SQL_SERVER -U $SQL_USER -P $SQL_PASSWORD -Q "BACKUP DATABASE [$SQL_DB] TO DISK = '$BACKUP_FILE';"

# log
echo "$(date +%Y-%m-%d %H:%M:%S)" - Backup script completed" | tee -a $LOG_FILE
```

Figura 13 - script onde o backup vai ser realizado

Explicação script:

1. Configuração Inicial:

- São definidos os dados necessários para que o servidor Linux possa conectar-se ao SQL Server, incluindo:
 - **Endereço do Servidor SQL** (SQL_SERVER),
 - **Utilizador e Senha** (SQL_USER e SQL_PASSWORD),
 - **Nome da Base de Dados** (SQL_DB).
- É configurado o diretório onde os ficheiros de backup serão armazenados (BACKUP_DIR) e o ficheiro de log (LOG_FILE), que registará as atividades do script.

2. Criação do Nome do Backup:

- O nome do ficheiro de backup é gerado dinamicamente com base na **data e hora atuais**, utilizando o comando date.
- Essa abordagem facilita a organização dos backups e a identificação de versões específicas no caso de uma recuperação.

3. Execução do Backup:

- O comando **sqlcmd** é utilizado para se conectar ao **SQL Server** e executar a instrução SQL **BACKUP DATABASE**. Esta instrução faz o backup da base de dados especificada para o ficheiro no caminho definido (**\$BACKUP_FILE**).
- Qualquer mensagem ou erro gerado pelo **sqlcmd** será exibido no terminal e registado no ficheiro de log.

4. Registo em Log:

- Cada etapa do processo é registada no ficheiro de log (**LOG_FILE**), incluindo o início e a conclusão do backup, bem como possíveis erros.
- Isto garante um histórico detalhado das operações realizadas.

```
GNU nano 5.4 /tmp/crontab.Zkf0dy/crontab *
#
# 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 3 * * * ./apps/HospitalManagementAppBE/deployBE.sh >> ./logs/logfile_$(date +%Y-%m-%d).log 2>&1
0 2 * * * ./deploy.sh >> ./logs/backup_log_$(date +%Y-%m-%d).log 2>&1
```

Figura 1164 - Adição do `deploy.sh` ao `crontab` para o script ser executado periodicamente

A seguir, configuramos o `crontab` para executar o script criado anteriormente de forma periódica. Adicionamos uma linha no `crontab` para que o backup seja realizado diariamente às 2 da manhã, garantindo que ocorra antes do `deploy do backend`. Esta configuração segue as diretrizes mencionadas na teoria, permitindo que o backup seja realizado num momento estratégico para minimizar o **RPO** (*Recovery Point Objective*) e preparar o sistema para um eventual processo de recuperação antes das atualizações no backend.

```
root@vs781:~/logs# cat backup_log_2024-11-23.log
2024-11-23 02:30:01 - Starting backup for database: SqlDb
Processed 800 pages for database 'SqlDb', file 'SqlDb' on file 1.
Processed 2 pages for database 'SqlDb', file 'SqlDb_log' on file 1.
BACKUP DATABASE successfully processed 802 pages in 0.307 seconds (20.396 MB/sec).
2024-11-23 02:30:03 - Backup script completed
```

Figura 15 - Exemplo de um backup realizado no dia 23/11/2024

User Story 7

Pedido: 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.

Aluno Responsável: Jack Pinheiro (1120419)

Solução: Primeiro passo é a criação da pasta que irá ser pública (partilhada) e depois é necessário as suas permissões para que os utilizadores possam ler o que lá for colocado.

Para a criação da pasta utilizamos o comando **mkdir**.

```
root@vs781:~# mkdir /shared_folder
```

Figura 16 - Criação da pasta partilhada

Mudamos as permissões da pasta para que o **owner** tenha permissões para ler, escrever e executar, o **group** tenha permissões para ler e executar e o mesmo para os **users**.

```
root@vs781:/# chmod 755 /shared_folder
```

Figura 17 – Mudar as permissões da pasta partilhada

Criamos um ficheiro de texto dentro da pasta partilhada para testarmos as permissões.

```
root@vs781:/# nano /shared_folder/test.txt
```

Figura 18 - Criação do ficheiro de teste

Executamos **ls-l /shared_folder** para verificarmos as permissões, conseguimos observar que o ficheiro tem permissões “-rw-r--r--” que significa que o **owner** tem permissões de escrita e leitura, e que o **grupo** e **users** têm só permissões de leitura.

```
root@vs781:/# ls -l /shared_folder
total 4
-rw-r--r-- 1 root root 32 Nov 24 19:26 test.txt
```

Figura 19 - Verificação das permissões do ficheiro dentro da pasta

User Story 8

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

Aluno Responsável: Vasco Sousa (1221700)

Solução: O primeiro passo para atingir os objetivos desta User Story foi a instalação do **rsyslog**. O **rsyslog** é uma ferramenta de **logging** utilizada em sistemas operativos **Unix** e **Linux** para recolher, processar e armazenar mensagens de **log**. Trata-se de uma versão mais avançada do tradicional **syslog**, oferecendo funcionalidades adicionais como segurança, filtragem, envio remoto de **logs** e desempenho superior.

Após esta instalação, foi necessário inicializar e ativar os serviços do **rsyslog** através dos comandos “**systemctl start rsyslog**” e “**systemctl enable rsyslog**”.

Passando agora para a criação do script, começamos por nos dirigir ao diretório “**/usr/local/bin**” através do comando “**cd /usr/local/bin**” e, já dentro da pasta **bin**, executamos o comando “**nano failedLogins.sh**”, responsável por criar e editar o ficheiro do script. O conteúdo definido no script foi o seguinte:

```
GNU nano 5.4 /usr/local/bin/check_failed_logins.sh *
#!/bin/bash

# Get a list of users with three or more failed logins
failed_users=$(lastb | awk '{print $1}' | sort | uniq -c | awk '$1 >= 3 {print $2, "->", $1, "failed attempts"}')

# Check if there are any failed users and display them
if [ -n "$failed_users" ]; then
    echo -e '\nUsers with 3 or more failed logins:\n'
    echo "$failed_users"
else
    echo "No users found"
fi

^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 20 - Script utilizadores com mais de 3 tentativas falhadas

Os comandos definidos neste script foram os seguintes:

Comando	Função
<code>lastb</code>	Exibe o registo de tentativas de login falhadas. Este registo encontra-se no ficheiro “ <code>/var/log/btmp</code> ”.
<code>awk '{print \$1}'</code>	Filtra e imprime a primeira coluna da saída de <code>lastb</code> (normalmente o nome de utilizador).
<code>sort</code>	Ordena a lista de utilizadores.
<code>uniq -c</code>	Conta a ocorrência de cada utilizador único na lista ordenada.
<code>awk '\$1 >= 3 {print \$2, "->", \$1, "failed attempts"}'</code>	Filtra os utilizadores com 3 ou mais tentativas falhadas e formata a saída.
<code>if [-n "\$failed_users"]</code>	Verifica se a variável “ <code>failed_users</code> ” não está vazia (ou seja, se há utilizadores com tentativas falhadas).
<code>echo "\$failed_users"</code>	Exibe a lista de utilizadores com 3 ou mais tentativas falhadas.

Alguns exemplos de output podem ser os seguintes:

- Caso sem utilizadores:

```
root@vs781:~# /usr/local/bin/check_failed_logins.sh
No users found
```

Figura 21 - Exemplo sem users

- Caso com utilizadores:

```
root@vs781:~# /usr/local/bin/check_failed_logins.sh

Users with 3 or more failed logins:

root -> 7 failed attempts
vasco -> 8 failed attempts
```

Figura 22 - Exemplo com users