Instituto Superior de Engenharia do Porto

Relatório ASIST

Sprint B

3DJ G059

Feito por:

André Ferreira, 1190378

David Marques, 1221276

Diogo Cunha, 1221071

João Monteiro, 1221023

Professor:

Bruno Alexandre Moreira e Silva (BAS)

15/11/2024

DEPARTAMENTO DE ENGENHARIA INFORMÁTICA
Instituto Superior de Engenharia do Porto

Índice

Distribuição de *User Stories*

USER STORY	MEMBRO RESPONSÁVEL
1	1221023
2	1190378
3	1190378
4	1221276
5	1221023
6	1221071
7	1221276
8	1221071

Ou seja, o membro:

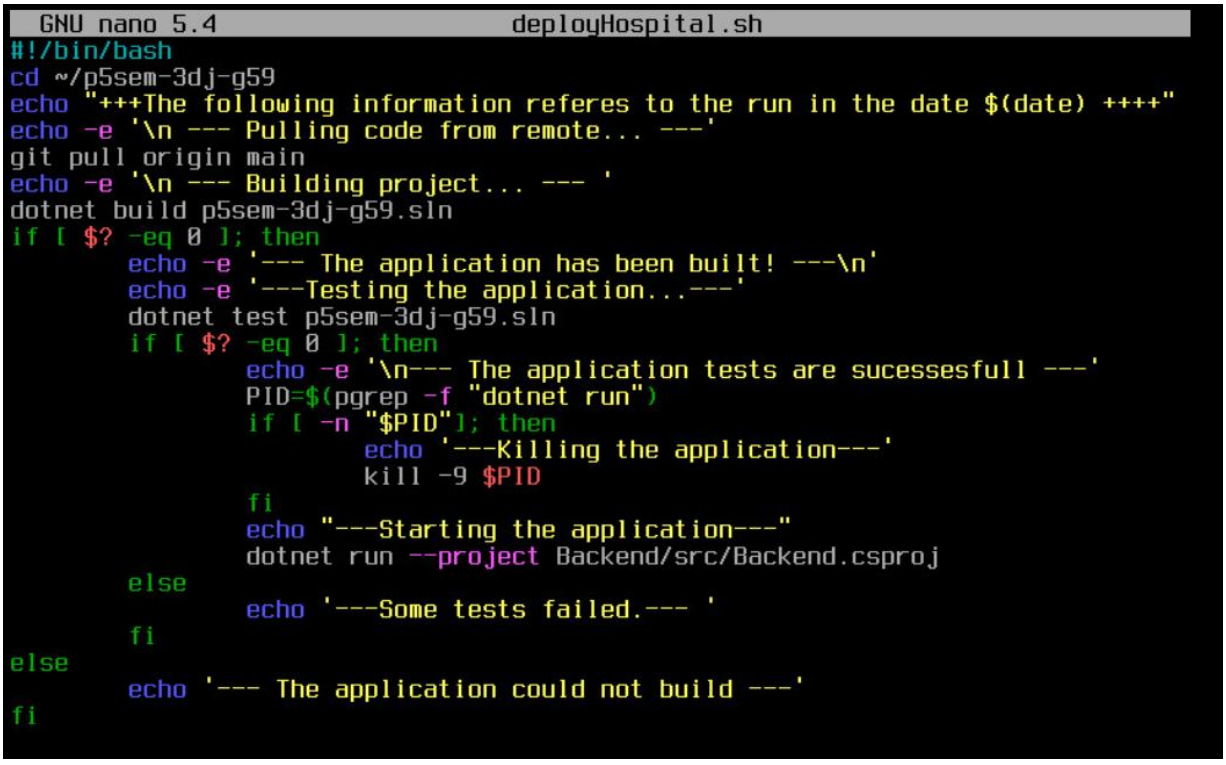
- 1190378 (André Ferreira) realizou as ***User Stories*** 2 e 3.
- 1221276 (David Marques) realizou as ***User Stories*** 4 e 7.
- 1221071 (Diogo Cunha) realizou as ***User Stories*** 6 e 8.
- 1221023 (João Monteiro) realizou as ***User Stories*** 1 e 5.

User Story 1

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.

O script *deployHospital.sh*, desenvolvido em Bash, foi projetado para:

1. **Atualizar o código fonte**: Utiliza o comando `git pull` para sincronizar a aplicação com o repositório remoto.
2. **Compilar o projeto**: Com o comando `dotnet build`, o script verifica a integridade do código e compila os arquivos.
3. **Executar os testes automatizados**: Após a compilação, o comando `dotnet test` executa os testes definidos no plano de testes, garantindo a validação funcional antes do deployment.
4. **Parar e reiniciar a aplicação, se necessário**: Caso uma instância da aplicação já esteja em execução, ela é identificada pelo seu processo PID e encerrada utilizando `kill -9`. Em seguida, o script reinicia a aplicação com o comando `dotnet run`.



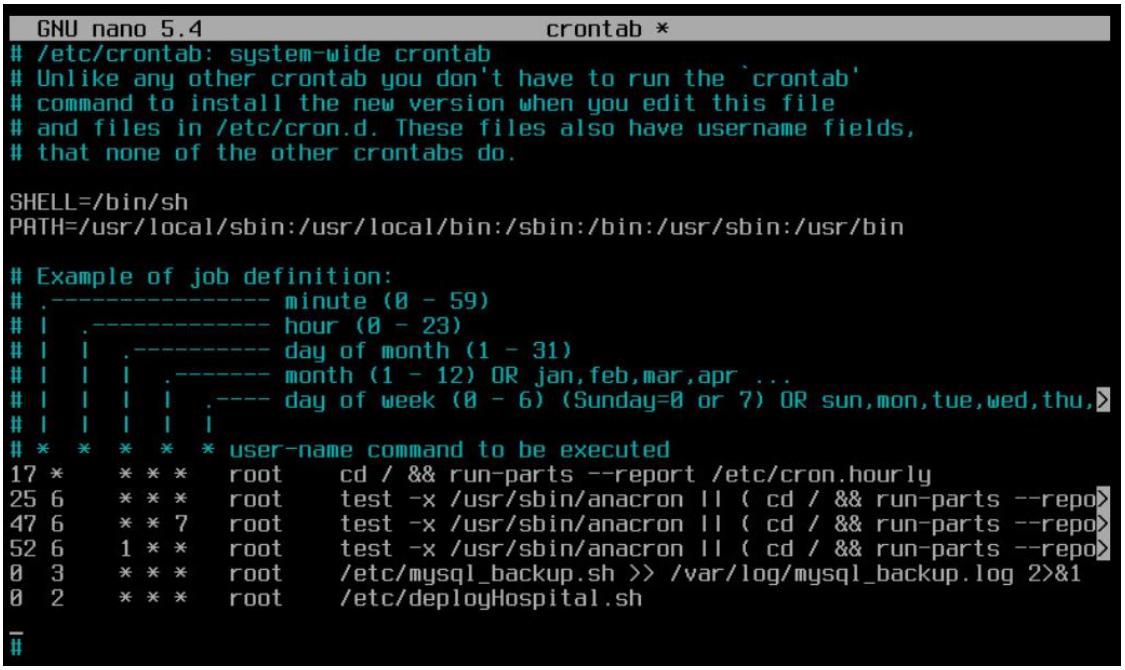
Este fluxo assegura que o módulo do RFP seja implantado apenas se os testes forem bem-sucedidos, protegendo a consistência e a confiabilidade do sistema.

Fluxo Condicional

- **Caso a compilação falhe**, o script termina imediatamente, informando que a aplicação não pôde ser construída.
- **Caso os testes falhem**, a execução do script é interrompida e uma mensagem de erro é exibida.

Configuração de Agendamento com o Crontab

Para garantir a execução automática e regular do script, configurei uma tarefa no *crontab* do sistema. A linha adicionada ao ficheiro de configuração do *crontab* é a mais abaixo na figura:



Explicação dos Parâmetros

- **Minuto (0)**: Define que a tarefa será iniciada no início do minuto.
- **Hora (2)**: Especifica que a execução ocorrerá às 2:00 da manhã.
- **Dia do mês (*):** Permite que a tarefa seja executada diariamente.
- **Mês (*):** Não restringe a execução a meses específicos.
- **Dia da semana (*):** Permite a execução em qualquer dia da semana.
- **Comando:** Define o script a ser executado, que está localizado no diretório `/etc`.

Com essa configuração, o sistema automaticamente executa o script todas as madrugadas, sem intervenção manual, promovendo um ciclo contínuo de integração e deployment.

User Story 2

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

Para ser possível restringir os utilizadores apenas à rede interna do DEI será necessário configurar a firewall da máquina de modo a que permita apenas acesso apenas à gama de ip's associados ao isep DEI.

Para manter as configurações da firewall sempre que a máquina é desligada optou-se por recorrer ao pacote de utilidade: iptables-persistent.

Em geral os passos passam por configurar a firewall com iptables e depois gravar as configurações com: iptables-save > /etc/iptables/rules.v4

Interface da conexão vpn

```
204: tun0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UNKNOWN g
roup default qlen 500
    link/none
    inet 10.8.205.37 peer 10.8.205.38/32 scope global noprefixroute tun0
       valid_lft forever preferred_lft forever
    inet6 fe80::91f7:c788:b1b1:2431/64 scope link stable-privacy
       valid_lft forever preferred_lft forever
```

Criar um script para facilitar as configuracoes ip tables:

```
sudo iptables -A INPUT -i tun0 -j ACCEPT    # Permite tráfego de entrada pela interface tun0 (VPN)
sudo iptables -A FORWARD -i tun0 -j ACCEPT  # Permite tráfego de encaminhamento pela interface tun0 (VPN)

# Permitir tráfego para conexões relacionadas com a VPN
# Aqui estamos permitindo o tráfego que faz parte de uma conexão já estabelecida ou relacionada
# "ESTABLISHED" significa conexões que já foram iniciadas e "RELATED" significa conexões relacionadas a essas, como respostas de ICMP
sudo iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
sudo iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT

# Permitir tráfego na porta UDP da OpenVPN (port 1194)
# Caso esteja usando a OpenVPN com a porta padrão 1194, é necessário permitir tráfego UDP nessa porta
sudo iptables -A INPUT -p udp --dport 1194 -j ACCEPT  # Permite tráfego UDP na porta 1194 (típica para OpenVPN)

# Permitir tráfego dentro da rede VPN
# Asseguramos que qualquer tráfego proveniente ou destinado à rede VPN seja permitido
# Este exemplo assume que a rede VPN usa o intervalo de IPs 10.8.205.0/24.
sudo iptables -A INPUT -s 10.8.205.0/24 -j ACCEPT   # Permite tráfego de entrada proveniente da rede VPN (10.8.205.0/24)
sudo iptables -A FORWARD -s 10.8.205.0/24 -j ACCEPT # Permite tráfego de encaminhamento proveniente da rede VPN (10.8.205.0/24)
sudo iptables -A OUTPUT -d 10.8.205.0/24 -j ACCEPT  # Permite tráfego de saída destinado à rede VPN (10.8.205.0/24)
sudo iptables -A FORWARD -d 10.8.205.0/24 -j ACCEPT # Permite tráfego de encaminhamento destinado à rede VPN (10.8.205.0/24)

sudo iptables-save > /etc/iptables/rules.v4
```

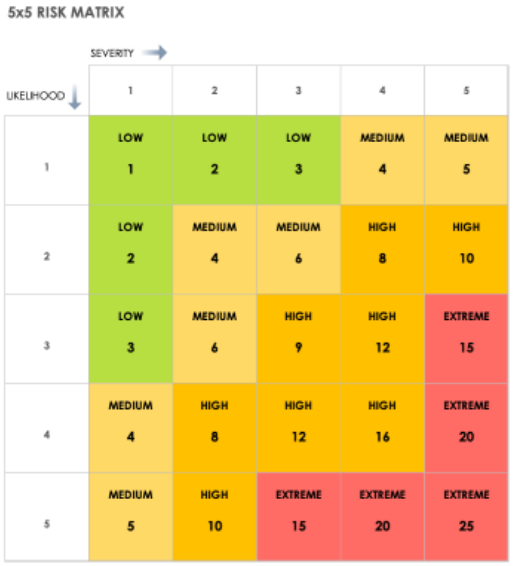
Como administrador do sistema quero que os clientes indicados na user story 2 possam ser definidos pela simples alteração de um ficheiro de texto.

Esta user story já foi efetuada na implementação da user story dois. Para cumprir os requisitos da mesma é possível alterar o ficheiro /etc/iptables/rules.v4 para decidir quais os clientes que podem comunicar com a máquina.

User Story 4

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

Para identificar os riscos da nossa solução iremos utilizar uma Matriz de Risco. Esta ferramenta permite gerir e identificar visualmente os riscos do sistema a que devemos prestar atenção. Olhando para a matriz o risco é calculado multiplicando a probabilidade pela consequência.



Atendendo à matriz apresentado podemos tirar as seguintes conclusões:

1. Interrupção do funcionamento dos servidores
    - Consequência: 4
    - Probabilidade: 2
    - Risco: 8 (Alto) – Estes problemas podem ser causados por fatores externos ou até uma avaria do servidor. A interrupção do funcionamento dos servidores causa um interrupção do serviço impedindo a continuidade do negócio.
2. Perda da Base de Dados
    - Consequência: 5
    - Probabilidade: 1
    - Risco: 5 (Médio) – Toda a informação relevante ao funcionamento da solução seria afetada independentemente se houve uma perda parcial ou total da base de dados.

3. Acesso Indevido
    - Consequência: 5
    - Probabilidade: 2
    - Risco: 10 (Alto) – O acesso indevido pode acontecer devido a falhas de configuração no controlo de autenticação e autorização, configuração inadequada de permissões e acessos, ou credenciais comprometidas. Isto pode levar a perda de confidencialidade e ao comprometimento da integridade dos dados.
4. Vulnerabilidades de Software
    - Consequência: 5
    - Probabilidade: 2
    - Risco: 10 (Alto) – O software pode ter vulnerabilidades desconhecidas, levando a que exploits, ataques de força bruta, ataques DOS ou injeções SQL, ponham em causa o bom funcionamento da solução e que sejam expostos dados sensíveis.

User Story 5

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

Objetivo mínimo de continuidade de negócios:

Garantir o agendamento e a gestão de recursos para cirurgias em pelo menos 30% da capacidade operacional, mantendo a integridade dos dados dos pacientes e a conformidade com o GDPR, enquanto possibilita o acesso básico ao sistema para profissionais de saúde e administradores.

1. **Serviços Críticos Identificados**:
    a. **Gestão de agendamentos:** Manter a capacidade de agendar cirurgias, priorizando casos urgentes e recursos críticos.
    b. **Acesso de profissionais de saúde:** Permitir que médicos, enfermeiros e administradores acessem informações relevantes para continuar os procedimentos cirúrgicos.
    c. **Conformidade com GDPR:** Garantir que os dados pessoais sejam protegidos e acessíveis apenas conforme permitido pelas leis de privacidade.
2. **Capacidades Necessárias**:
    a. Infraestrutura Técnica:
        i. Manter o funcionamento do **módulo de backoffice** para gestão de dados de pacientes, profissionais de saúde e salas cirúrgicas.
        ii. Operar o **módulo de planejamento**, garantindo a geração de horários simplificados.
        iii. Prover **acesso básico ao módulo GDPR**, para atender a requisitos de auditoria e notificações de conformidade.
    b. Recursos Humanos:
        i. Uma equipa técnica mínima para suporte e recuperação de sistemas.
    c. Backup e Recuperação:
        i. Garantir backups frequentes para minimizar o **RPO (Recovery Point Objective)** e recuperação rápida para atender ao **WRT (Work Recovery Time)**.
3. **Impacto e Benefícios**:
    a. **Redução de riscos clínicos:** Manter a capacidade de gerenciar casos críticos reduz atrasos que podem impactar a saúde dos pacientes.
    b. **Preservação da reputação:** Evitar interrupções totais que poderiam comprometer a confiança dos usuários no sistema.

c. **Conformidade legal:** Evitar penalidades relacionadas a violações de privacidade.

4. **Estratégias de Implementação**:
    a. **Plano de Failover:** Implementar um ambiente alternativo (por exemplo, numa VM do DEI) para manter a disponibilidade mínima.
    b. **Monitoramento Contínuo:** Detectar e responder rapidamente a falhas, priorizando a recuperação dos módulos mais críticos.
    c. **Comunicação com Stakeholders:** Informar stakeholders sobre a capacidade mínima operacional durante interrupções.

User Story 6

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*).

Visto que o código das várias componentes está contido no ***Github***, a equipa propõe uma estratégia de cópia de segurança da base de dados.

Para isto, foi elaborado um *script* chamado ***/etc/mysql_backup.sh***. As próximas prints e explicações fazem parte deste mesmo *script*.

```
# MySQL config
HOSTNAME="vsgate-s1.dei.isep.ipp.pt"
PORT="10694"
USERNAME="root"
PASSWORD="password"
DATABASE="projeto5sem"

# Backup output folders
BACKUP_DIR="/var/backups/mysql"
FULL_BACKUP_DIR="$BACKUP_DIR/full"
INCREMENTAL_BACKUP_DIR="$BACKUP_DIR/incremental"

DATE=$(date +%F)
DAY_OF_WEEK=$(date +%u)

# If the folders don't exist, create them
mkdir -p "$FULL_BACKUP_DIR" "$INCREMENTAL_BACKUP_DIR"
chmod 700 "$FULL_BACKUP_DIR" "$INCREMENTAL_BACKUP_DIR"
```

Figura 1. Setup ***/etc/mysql_backup.sh***

Explicação do código:

- ***# MySQL config***
  - Nesta secção, são declaradas as variáveis de configuração do servidor ***MySql***.
- ***# Backup output folders***
  - Nesta secção, são declarados os nomes das pastas onde serão colocados os *backups*.

- ***# If the folders don't exist, create them***
    - Como o comentário diz, nesta secção criamos as pastas onde serão colocados os *backups* e damos as permissões necessárias só para o *owner*.

```
# Methods
complete_backup() {
    BACKUP_FILE="$FULL_BACKUP_DIR/full_backup_$DATE.sql.gz"
    echo "[$(date)] Starting complete backup on schema $DATABASE from host $HOSTNAME:$PORT"
    mysqldump -h "$HOSTNAME" -P "$PORT" -u "$USERNAME" -p"$PASSWORD" "$DATABASE" | gzip > "$BACKUP_FILE"

    if [ $? -eq 0 ]; then
        echo "[$(date)] Complete backup successfully completed: $BACKUP_FILE"
    else
        echo "[$(date)] ERRO: Failure to start complete backup." >&2
        exit 1
    fi
}
```

Figura 2. ***complete_backup()*** *de* ***/etc/mysql_backup.sh***

Explicação do código:

- **mysqldump -h "$HOSTNAME" -P "$PORT" -u "$USERNAME" -p"$PASSWORD" "$DATABASE" | gzip > "$BACKUP_FILE"**
    - Conecta-se ao servidor com as credenciais previamente estabelecidas e cria um *dump* do *schema* especificado.
    - Comprime o arquivo para otimizar espaço e coloca o resultado na pasta de *output*.
- ***Bloco if*:**
    - Se ocorrer um erro (valor de retorno não é **0**), o administrador é avisado e é retornado o valor **1**.

```
partial_backup() {
    LAST_FULL_BACKUP=$(find "$FULL_BACKUP_DIR" -type f -name "full_backup_*.sql.gz" | sort | tail -n 1)
    if [ -z "$LAST_FULL_BACKUP" ]; then
        echo "[$(date)] No complete backup was found. Starting complete backup to compensate."
        complete_backup
        return
    fi

    BACKUP_FILE="$INCREMENTAL_BACKUP_DIR/incremental_backup_$DATE.sql.gz"
    echo "[$(date)] Starting partial backup on schema $DATABASE from host $HOSTNAME:$PORT"
    mysqldump -h "$HOSTNAME" -P "$PORT" -u "$USERNAME" -p"$PASSWORD" --single-transaction --quick --databases "$DATABASE" \
        --master-data=2 | gzip > "$BACKUP_FILE"

    if [ $? -eq 0 ]; then
        echo "[$(date)] Partial backup successfully completed: $BACKUP_FILE"
    else
        echo "[$(date)] ERROR: Failure to start partial backup" >&2
        exit 1
    fi
}
```

Figura 3. ***partial_backup()*** *de* ***/etc/mysql_backup.sh***

Explicação do código:

- ***Primeiras cinco linhas:***
  - Verifica que existe um *backup* completo no sistema. Se não existir, começa a criação dum *backup* completo, por segurança.
- **mysqldump -h "$HOSTNAME" -P "$PORT" -u "$USERNAME" -p"$PASSWORD" --single-transaction --quick --databases "$DATABASE" \ --master-data=2 | gzip > "$BACKUP_FILE"**
  - A mesma coisa feita no método ***complete_backup()***, mas com novas opções, a mais relevante sendo a "**--master-data=2**", que permite identificar se foram feitas alterações desde o último *backup* completo.
- ***Último bloco if:***
  - Faz exatamente o mesmo que o bloco *if* do método anterior.

```
clean_old_backups() {
    echo "[$(date)] Cleaning old backups..."
    find "$FULL_BACKUP_DIR" "$INCREMENTAL_BACKUP_DIR" -type f -mtime +7 -name "*.sql.gz" -exec rm {} \;
}
```

Figura 4. ***clean_old_backups()*** *de* ***/etc/mysql_backup.sh***

Este método foi criado para remover *backups* com mais de 7 dias.

```
if [ "$DAY_OF_WEEK" -eq 7 ]; then
    complete_backup
else
    partial_backup
fi

# Cleaning
clean_old_backups

echo "[$(date)] Backup complete."
exit 0
```

Figura 5. Fim de ***/etc/mysql_backup.sh***

Se for domingo, o programa roda o método ***complete_backup()***, caso contrário, roda o ***partial_backup()***. De qualquer forma, verifica se há *backups* antigos para remover, e acaba a sua execução com o valor 0.

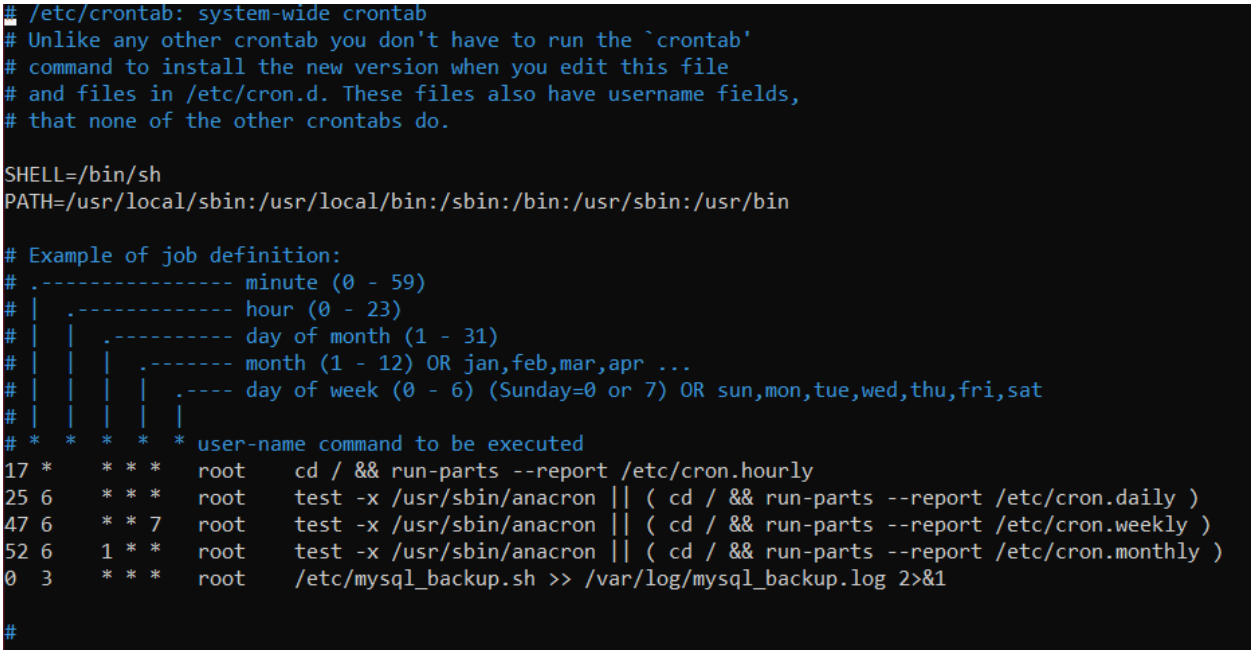


*Figura 6. Escalonamento no **crontab***

Esta linha foi colocada no ficheiro ***/etc/crontab*** para agendar a execução do *script* explicado anteriormente, todos os dias às 03:00, a linha "**/etc/mysql_backup.sh >> /var/log/mysql_backup.log 2>&1**" é executada pelo utilizador *root*.

- O *output* do *script* é colocado no ficheiro **/var/log/mysql_backup.log**, se este já existir, isto é escrito no seu final, sem apagar o que já está lá.
- Tanto o **stdout** como o **stderr** são registados no ficheiro.

Com esta estratégia, obtemos:

- **RPO = 24 horas**, visto que o *backup* é feito/atualizado diariamente.
- **WRT = Baixo**, visto que a cópia completa só é feita aos domingos.
    - Uma falha imediatamente antes da criação duma cópia completa (domingo às 03:00) resultará num **WRT** maior, visto que o programa terá de repor 1 cópia completa e 6 parciais. **Este é o pior caso**.

User Story 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.

Para criar uma pasta pública, no perfil root foram executados os seguintes passos:

1. Criou-se uma pasta public com o seguinte comando:

```
root@vm059:~# mkdir public
```

2. Atribuíram-se permissões de leitura a todos os utilizadores do sistema e de execução ao root, com os seguintes comandos:

```
root@vm059:/public# sudo chmod 755 /public
```

- `7` (rwx) para o proprietário (root) — permite leitura, escrita e execução.

- `5` (r-x) para o grupo e outros — permite apenas leitura e execução (o que é necessário para aceder os arquivos no diretório).

3. Verificou-se se as permissões foram aplicadas corretamente:

```
root@vm059:/# ls -ld /public
drwxr-xr-x 2 root root 4096 Nov 23 12:30 /public
```

User Story 8

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

Para obter os utilizadores com mais do que 3 tentativas de acesso incorretas, foi preparado um *script* que analisa os registos em **/var/log/auth.log** e organiza as linhas que contêm falhas na password e organizando-as por utilizador, ***get_users_with_3plus_failed_login_attempts.sh***. O resultado é impresso no ecrã para o administrador visualizar.

```
#!/bin/bash

limit=3

grep 'Failed password' /var/log/auth.log | grep -v 'invalid' | \

awk '{print $9}' | \

sort | uniq -c | \

awk -v limit="$limit" '$1 > limit {print $2, "had", $1, "failed login attempts"}'
```

Figure 1. get_users_with_3plus_failed_login_attempts.sh

Explicação por linha:

- **limit=3**
  - O número de acessos até o utilizador aparecer no *output* deste *script*, colocado numa variável para cumprir com boas práticas.
- **grep 'Failed password' /var/log/auth.log | grep -v 'invalid' | **
  - Obtém todas as linhas com **'Failed Password'** de **/var/log/auth.log**.
  - Exclui linhas com **'invalid'**, ignorando tentativas de autenticação inválidas.
- **awk '{print $9} ' | **
  - Extrai a 9ª coluna de cada linha (nome do utilizador).
- **sort | uniq -c | **
  - Junta os resultados através do seu nome.
  - Conta o número de instâncias de cada nome.
- **awk -v limit="$limit" '$1 > limit {print $2, "had", $1, "failed login attempts"}'**
  - Exclui utilizadores com um número de tentativas de acesso incorretas menor ou igual a **limit**.
  - Imprime os vários utilizadores.

Aqui, vemos o *script* em execução:

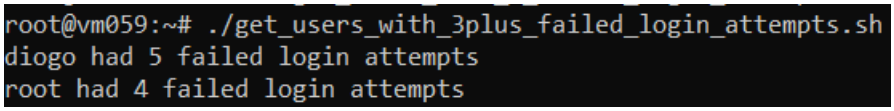


Figure 2. Exemplo de execução do script