

RELATÓRIO ASIST – SPRINT 3

5 JANEIRO

Professor: Rui Filipe Nogueira Marques (RFM)

Da autoria de: Grupo 15, 3DC



Índice:

Índice:..... 1

User Story 1..... 2

User Story 2..... 4

User Story 3..... 6

User Story 4..... 10

User Story 5..... 16

User Story 6..... 20

User Story 7..... 22

User Story 8..... 27

User Story 9..... 29

User Story 10..... 32

User Story 11..... 35

User Story 12..... 40

User Story 1

Enunciado: Como administrador da organização quero um plano de recuperação de desastre que satisfaça o MBCO definido no sprint B.

Aluno Responsável: Vasco Sousa (1221700)

Solução: O objetivo deste plano é garantir a recuperação eficaz do sistema de gestão de recursos e marcação de cirurgias após um desastre. O plano visa minimizar a interrupção das operações e a perda de dados, em conformidade com o **MBCO (Minimum Business Continuity Objective)** definido no Sprint B. Para tal, o plano considera os riscos identificados no Sprint B, as análises de impacto nos negócios e a estratégia de backup já implementada. O plano detalha as etapas para a recuperação dos serviços essenciais, com foco na minimização do **RPO (Recovery Point Objective)** e do **WRT (Work Recovery Time)**.

A equipa responsável pela resposta a desastres será composta por: um **Líder da Equipa**, responsável pela coordenação geral da resposta; um **Administrador de Sistemas**, responsável pelos servidores, rede, backups e firewall; um **Administrador de Bases de Dados**, responsável pela recuperação das bases de dados; um **Especialista em Segurança**, responsável pela gestão de acessos e resposta a incidentes de segurança; e uma **Equipa de Comunicação**, responsável pela comunicação interna e externa.

O plano define os seguintes **RTOs (Recovery Time Objective)**, **RPOs (Recovery Point Objective)** e **MTD (Maximum Tolerable Downtime)**:

- **RTO:** O tempo máximo aceitável para a recuperação de cada serviço.
 - **Planeamento e Otimização de Cirurgias:** Tempo mínimo possível.
- **RPO:** A quantidade máxima de dados aceitável a perder.
 - **Todos os serviços:** Minimizado através de backups diários às 2h da manhã.
- **MTD:** Tempo máximo de inatividade tolerável do sistema.
 - **20 minutos**

A documentação detalhada da infraestrutura é essencial para a recuperação do sistema. Esta documentação deve incluir **diagramas de rede**, **configurações de hardware e software**, **dependências entre sistemas** e **informações de contacto da equipa**. A implementação de **ferramentas de monitorização** para detetar falhas e enviar alertas é crucial, bem como a utilização do **sistema de logs do Linux (rsyslog)** para registar eventos e notificar o administrador sobre falhas graves.

A **estratégia de backup** e redundância do sistema inclui a realização de **backups diários** das bases de dados às **2h da manhã**, armazenados localmente para minimizar o **WRT**. Backups adicionais na **cloud (AWS S3)** e num **servidor remoto**, seguindo a **regra 3-2-1**, também são recomendados. É fundamental **verificar a integridade** de cada backup e implementar um script para **automatizar a restauração e validação dos backups**. O plano também prevê a **redundância de rede**, servidores redundantes com balanceamento de carga e failover automático via **HAProxy**, e a disponibilização de **peças de reposição** para hardware crítico.

Os critérios para a ativação do plano incluem: **falha total ou parcial dos serviços essenciais**, como o Planeamento e Otimização de Cirurgias, a Gestão de dados de pacientes (GDPR), a gestão de utilizadores e a submissão e gestão de pedidos de operação; **desastres naturais**, como incêndios, inundações e terremotos; falhas de hardware/software, como falha de servidores, falha de rede e corrupção de dados; **ataques de segurança**, como ataques de ransomware e negação de serviço; e **tentativas de acesso indevido** ao sistema.

O processo de recuperação de desastres envolve as seguintes etapas:

1. **Identificação e Avaliação do Desastre:** Determinar a causa, impacto e extensão do desastre.
2. **Comunicação:** Notificar a equipa de resposta a desastres, stakeholders e, se necessário, as autoridades competentes.
3. **Contenção:** Isolar sistemas afetados para evitar propagação do problema.
4. **Ativação do Plano:** Iniciar os procedimentos de recuperação de acordo com o tipo de desastre.
5. **Restauração de Backups:** Restaurar os backups mais recentes da base de dados, priorizando os backups locais para minimizar o WRT.
6. **Recuperação de Hardware/Software:** Substituir ou reparar componentes defeituosos, utilizando peças de reposição disponíveis.
7. **Testes e Validação:** Assegurar que os sistemas restaurados funcionam corretamente, executando testes e queries de validação.
8. **Comunicação de Status:** Manter as partes interessadas informadas sobre o progresso da recuperação.
9. **Documentação:** Documentar as ações tomadas, os tempos de recuperação e as lições aprendidas.

É importante **testar o plano de recuperação de desastres** regularmente, realizando **simulações de desastres** e **testes de recuperação de backups**. O plano deve ser **revisto** e **atualizado** pelo menos **anualmente**, e **após cada incidente**, para se **adaptar a mudanças na infraestrutura, novos riscos e requisitos de negócio**.

Este plano de recuperação de desastres visa **minimizar** o impacto de eventos inesperados no sistema de gestão de recursos e marcação de cirurgias, garantindo a **continuidade das operações e a proteção dos dados**, com foco na **rápida recuperação dos serviços essenciais** e na **minimização da perda de dados**. É crucial que a equipa de resposta a desastres esteja familiarizada com o plano e que o mesmo seja **testado** e **atualizado** regularmente para garantir a sua **eficácia**.

User Story 2

Pedido: Como administrador da organização quero que me seja apresentada de forma justificada a ou as alterações a realizar na infraestrutura por forma a assegurar um MTD (Maximum Tolerable Downtime) de 20 minutos.

Aluno Responsável: Vasco Sousa (1221700)

Solução: Para garantir um **MTD (Maximum Tolerable Downtime)** de **20 minutos** para o sistema de gestão de recursos e marcação de cirurgias, é crucial implementar uma série de **alterações** na infraestrutura, com foco na **redundância**, **monitorização** e **rápida recuperação de serviços críticos**. O objetivo é **minimizar o impacto** de falhas e interrupções, assegurando a **continuidade** das operações e a disponibilidade dos dados.

Para alcançar este objetivo, é crucial adotar uma abordagem **holística** que englobe redundância de **hardware** e **software**, monitorização **constante** e **eficaz**, **gestão robusta de backups** e **recuperação de desastres** e, por fim, **segurança reforçada e controlo de acessos**.

Redundância em todos os níveis é a chave para **minimizar** o tempo de inatividade. A implementação de **servidores redundantes** para os módulos críticos, como o **Planeamento e Otimização de Cirurgias** e o módulo **GDPR**, garante a **continuidade** dos serviços mesmo em caso de **falha** de um servidor. O **balanceamento de carga e failover**, através de ferramentas como o **HAProxy**, permite **distribuir** o tráfego entre os servidores e garantir a comutação automática em caso de **falha**. A **replicação** das bases de dados, é outra medida crucial para **assegurar** a disponibilidade dos dados, mantendo uma **cópia sincronizada** num **servidor secundário**. A redundância de rede, com **múltiplos links de internet** e **dispositivos de rede redundantes**, garante a **conectividade** mesmo perante falhas.

Monitorização contínua e eficaz é fundamental para **detetar** e **responder** a problemas **rapidamente**. A implementação de ferramentas de monitorização para verificar o estado dos **servidores**, da **rede**, das **bases de dados** e dos **serviços críticos**, com alertas em caso de **falhas** ou **problemas de desempenho**, é essencial. Um **sistema de logs centralizado**, como o **rsyslog**, facilita a identificação de problemas e permite configurar **alertas** para notificar o administrador sobre **falhas graves**. A realização **regular** de **testes de carga e stress**, permite verificar a **capacidade** do sistema em lidar com **picos de tráfego** e **identificar potenciais problemas**.

A segurança e o controlo de acessos são aspetos críticos para garantir a integridade e a disponibilidade do sistema. Uma **gestão de acessos rigorosa**, com **autenticação forte**, **controlo de acessos baseado em funções** e **políticas de segurança**, garante que apenas **utilizadores autorizados** têm **acesso a dados sensíveis**. Um **firewall robusto** e **sistemas de deteção de intrusão** protegem a infraestrutura contra **ataques externos**. A implementação de um **VPN** para acesso remoto garante a **segurança das conexões**.

A realização de uma **Análise de Impacto nos Negócios (BIA)** e a **gestão de riscos** são essenciais para **definir as prioridades de recuperação** e os **RTOs** aceitáveis para cada serviço. A **BIA** identifica os **processos críticos do negócio** e os **impactos potenciais de interrupções**, permitindo a implementação de **medidas de mitigação adequadas**.

Ao implementar estas alterações, o sistema estará **mais preparado** para lidar com **falhas e interrupções**, minimizando o **tempo de inatividade** e garantindo o **MTD** de **20 minutos**. A **redundância**, a **monitorização**, a **gestão de backups** e a **segurança** são essenciais para a **continuidade das operações** e a **disponibilidade dos dados**. Tal como mencionado na User Story 1, o **plano de recuperação de desastres** deve ser **testado** e **atualizado** regularmente de forma a garantir a sua **eficácia**.

User Story 3

Pedido: Como administrador de sistemas quero que seja realizada uma cópia de segurança da(s) DB(s) para um ambiente de Cloud através de um script que a renomeie para o formato `_yyyymmdd` sendo o nome da base de dados, `yyyy` o ano de realização da cópia, `mm` o mês de realização da cópia e `dd` o dia da realização da cópia.

Aluno Responsável: João Pereira (1211503)

Solução: Para garantir a integridade e a segurança dos dados, foi definida a necessidade de realizar cópias de segurança diárias da nossa base de dados. A tarefa de backup envolve a criação de cópias completas da base de dados, que serão armazenadas em formato comprimido para otimizar o uso de espaço. Foi estabelecido que o formato das cópias de segurança seguiria o padrão `_yyyymmdd`, onde:

- `yyyy` refere-se ao ano de realização do backup,
- `mm` é o mês,
- `dd` é o dia em que a cópia de segurança foi realizada.

Antes de realizar o backup, foi necessário garantir que todos os pacotes e ferramentas necessárias para o MongoDB estivessem devidamente instalados no sistema Linux. Embora a instalação dos pacotes não faça parte deste relatório, é importante destacar que a ferramenta `mongodump`, fornecida pelo MongoDB, foi configurada para permitir a criação de backups da base de dados.

O comando utilizado para realizar o backup da nossa base de dados do MongoDB é o seguinte:

```
"mongodump -- uri='mongodb://mongoadmin:fb2760583afec14f39c8aa64@vs884.dei.isep.ipp.pt:27017/admin'"
```

Este comando conecta-se ao servidor MongoDB e cria uma cópia de segurança da base de dados.

Após a execução do comando `mongodump`, os dados da base de dados são armazenados num diretório.

Para otimizar o espaço de armazenamento e facilitar o envio ou arquivamento dos backups, a cópia de segurança é compactada utilizando o comando `tar`.

O comando `tar` é usado para compactar o ficheiro que contem o backup num arquivo `.tar.gz`, o que reduz significativamente o tamanho do arquivo, tornando o processo mais eficiente.

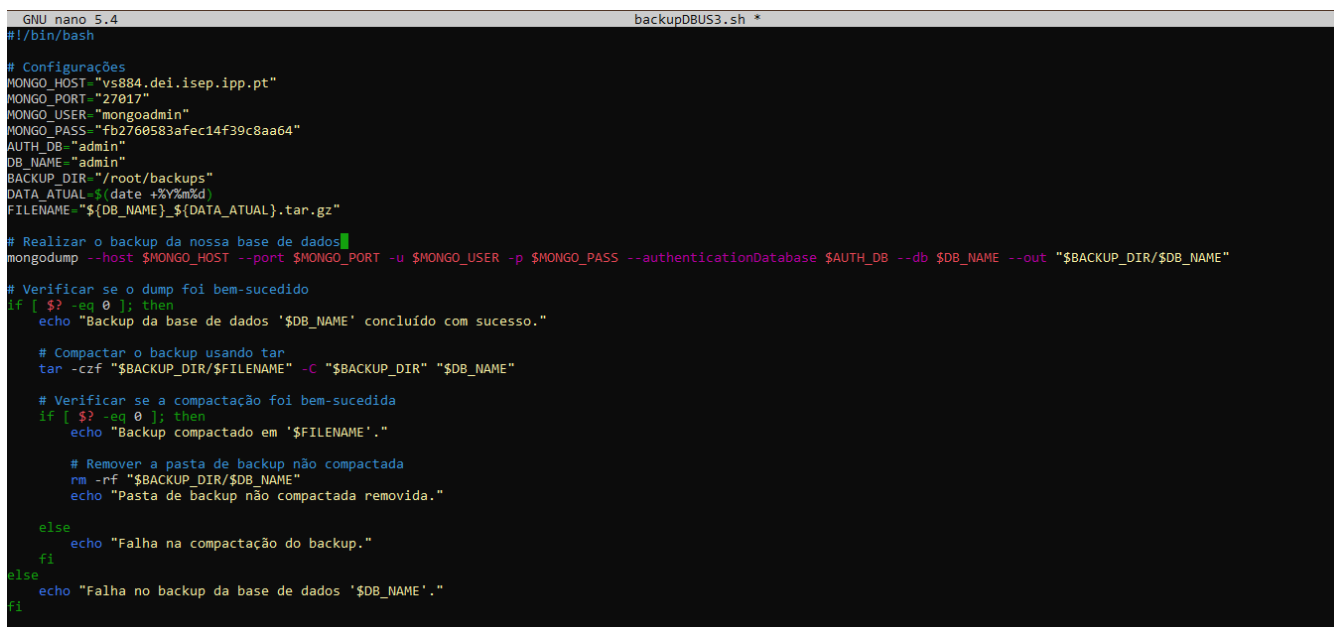
Esta combinação dos comandos **`mongodump`** e **`tar`** permite que o backup da base de dados seja realizado de forma automatizada, eficiente e compactada.

Criação Script

Seguindo esta linha de pensamento podemos então começar a criar o script, com o seguinte comando:

```
root@vs405:~# nano backupDBUS3.sh
```

E por seguinte podemos desenvolver o script:



```
GNU nano 5.4 backupDBUS3.sh *
#!/bin/bash

# Configurações
MONGO_HOST="vs884.dei.isep.ipp.pt"
MONGO_PORT="27017"
MONGO_USER="mongoadmin"
MONGO_PASS="fb2760583afec14f39c8aa64"
AUTH_DB="admin"
DB_NAME="admin"
BACKUP_DIR="/root/backups"
DATA_ATUAL=$(date +%Y%m%d)
FILENAME="${DB_NAME}_${DATA_ATUAL}.tar.gz"

# Realizar o backup da nossa base de dados
mongodump --host $MONGO_HOST --port $MONGO_PORT -u $MONGO_USER -p $MONGO_PASS --authenticationDatabase $AUTH_DB --db $DB_NAME --out "$BACKUP_DIR/$DB_NAME"

# Verificar se o dump foi bem-sucedido
if [ $? -eq 0 ]; then
    echo "Backup da base de dados '$DB_NAME' concluído com sucesso."

    # Compactar o backup usando tar
    tar -czf "$BACKUP_DIR/$FILENAME" -C "$BACKUP_DIR" "$DB_NAME"

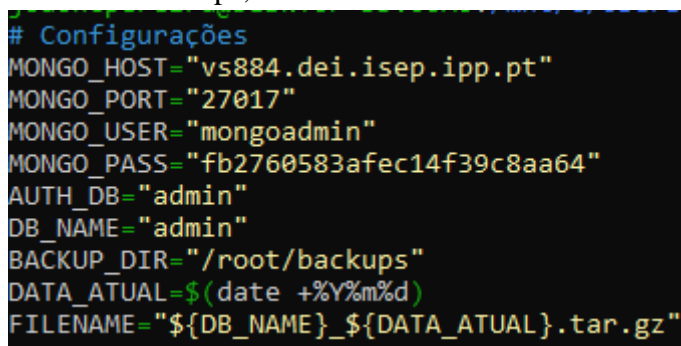
    # Verificar se a compactação foi bem-sucedida
    if [ $? -eq 0 ]; then
        echo "Backup compactado em '$FILENAME'."

        # Remover a pasta de backup não compactada
        rm -rf "$BACKUP_DIR/$DB_NAME"
        echo "Pasta de backup não compactada removida."
    else
        echo "Falha na compactação do backup."
    fi
else
    echo "Falha no backup da base de dados '$DB_NAME'."
fi
```

Figura 1- Script criado para realizar o backup da base de dados

Explicação script:

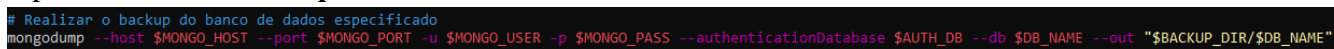
Inicialmente definiu-se os dados da base de dados onde se irá realizar a cópia de segurança, bem como o diretório onde vão ser colocados os backups, e ainda o nome do ficheiro como foi pedido na US.



```
# Configurações
MONGO_HOST="vs884.dei.isep.ipp.pt"
MONGO_PORT="27017"
MONGO_USER="mongoadmin"
MONGO_PASS="fb2760583afec14f39c8aa64"
AUTH_DB="admin"
DB_NAME="admin"
BACKUP_DIR="/root/backups"
DATA_ATUAL=$(date +%Y%m%d)
FILENAME="${DB_NAME}_${DATA_ATUAL}.tar.gz"
```

Figura 2 - Parte do script com a definição das variáveis

De seguida realiza-se o backup da base de dados definida, através do comando **“mongodump”** para o repositório **“/root/backups”**.



```
# Realizar o backup do banco de dados especificado
mongodump --host $MONGO_HOST --port $MONGO_PORT -u $MONGO_USER -p $MONGO_PASS --authenticationDatabase $AUTH_DB --db $DB_NAME --out "$BACKUP_DIR/$DB_NAME"
```

Figura 3 - Comando utilizado para a realização do backup

De seguida realiza-se a compactação do backup através do comando “**tar**” e adiciona-se o backup compactado a “**/root/backups**”. Por fim, apaga-se o primeiro backup do mongodump, para ficarmos só com o ficheiro compactado.

```
# Verificar se o dump foi bem-sucedido
if [ $? -eq 0 ]; then
    echo "Backup da base de dados '$DB_NAME' concluído com sucesso."

    # Compactar o backup usando tar
    tar -czf "$BACKUP_DIR/$FILENAME" -C "$BACKUP_DIR" "$DB_NAME"

    # Verificar se a compactação foi bem-sucedida
    if [ $? -eq 0 ]; then
        echo "Backup compactado em '$FILENAME'."


        # Remover a pasta de backup não compactada
        rm -rf "$BACKUP_DIR/$DB_NAME"
        echo "Pasta de backup não compactada removida."

    else
        echo "Falha na compactação do backup."
    fi
else
    echo "Falha no backup da base de dados '$DB_NAME'."
fi
```

Figura 4 - Parte do Script com as verificações pós-backup

Automatização da execução do script

Para concluir esta User Story, bastava adicionar este script no “**crontab**”, para que este seja executado automaticamente todos os dias às 2h da manhã.



```
GNU nano 5.4 /tmp/crontab.mOHAbW/crontab *
# 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
* * * * * /usr/local/bin/generate-ssh-banner.sh
0 2 * * * /bin/bash /root/backupDBUS3.sh
```

Figura 5 - Adição do script no crontab para execução automática do backup

Demonstração do funcionamento do script

Execução do script:

```
root@vs405:~# ./backupDBUS3.sh
2024-12-18T15:58:55.315+0000 writing admin.system.users to /root/backups/admin/admin/system.users.bson
2024-12-18T15:58:55.317+0000 done dumping admin.system.users (1 document)
2024-12-18T15:58:55.317+0000 writing admin.system.version to /root/backups/admin/admin/system.version.bson
2024-12-18T15:58:55.318+0000 done dumping admin.system.version (2 documents)
2024-12-18T15:58:55.319+0000 writing admin.appointments to /root/backups/admin/admin/appointments.bson
2024-12-18T15:58:55.321+0000 writing admin.specializations to /root/backups/admin/admin/specializations.bson
2024-12-18T15:58:55.322+0000 writing admin.allergies to /root/backups/admin/admin/allergies.bson
2024-12-18T15:58:55.323+0000 writing admin.medicalconditions to /root/backups/admin/admin/medicalconditions.bson
2024-12-18T15:58:55.372+0000 done dumping admin.appointments (17 documents)
2024-12-18T15:58:55.374+0000 done dumping admin.specializations (6 documents)
2024-12-18T15:58:55.376+0000 done dumping admin.medicalconditions (3 documents)
2024-12-18T15:58:55.377+0000 done dumping admin.allergies (3 documents)
2024-12-18T15:58:55.377+0000 writing admin.patientmedicalhistories to /root/backups/admin/admin/patientmedicalhistories.bson
2024-12-18T15:58:55.380+0000 done dumping admin.patientmedicalhistories (1 document)
Backup da base de dados 'admin' concluído com sucesso.
Backup compactado em 'admin_20241218.tar.gz'.
Pasta de backup não compactada removida.
```

Figura 6 - Execução do script

Backup da base de dados compactado no diretório definido.

```
root@vs405:~/backups# ls
admin_20241218.tar.gz
```

Figura 7 - Demonstração do script criado pós execução

User Story 4

Pedido: Como administrador de sistemas quero que utilizando o Backup elaborado na US C3, seja criado um script que faça a gestão dos ficheiros resultantes desse backup, no seguinte calendário. 1 Backup por mês no último ano, 1 backup por semana no último mês, 1 backup por dia na última semana.

Aluno Responsável: João Pereira (1211503)

Solução: Para realizar esta User Story, o pensamento foi ter dois repositórios distintos. Neste repositório iremos guardar sempre 1 backup por dia na última semana, 1 backup por semana no último mês, e 1 backup por mês no último ano.

Para isso foi criado um repositório novo chamado: “backups_saved”. Dentro deste repositório temos 3 pastas:

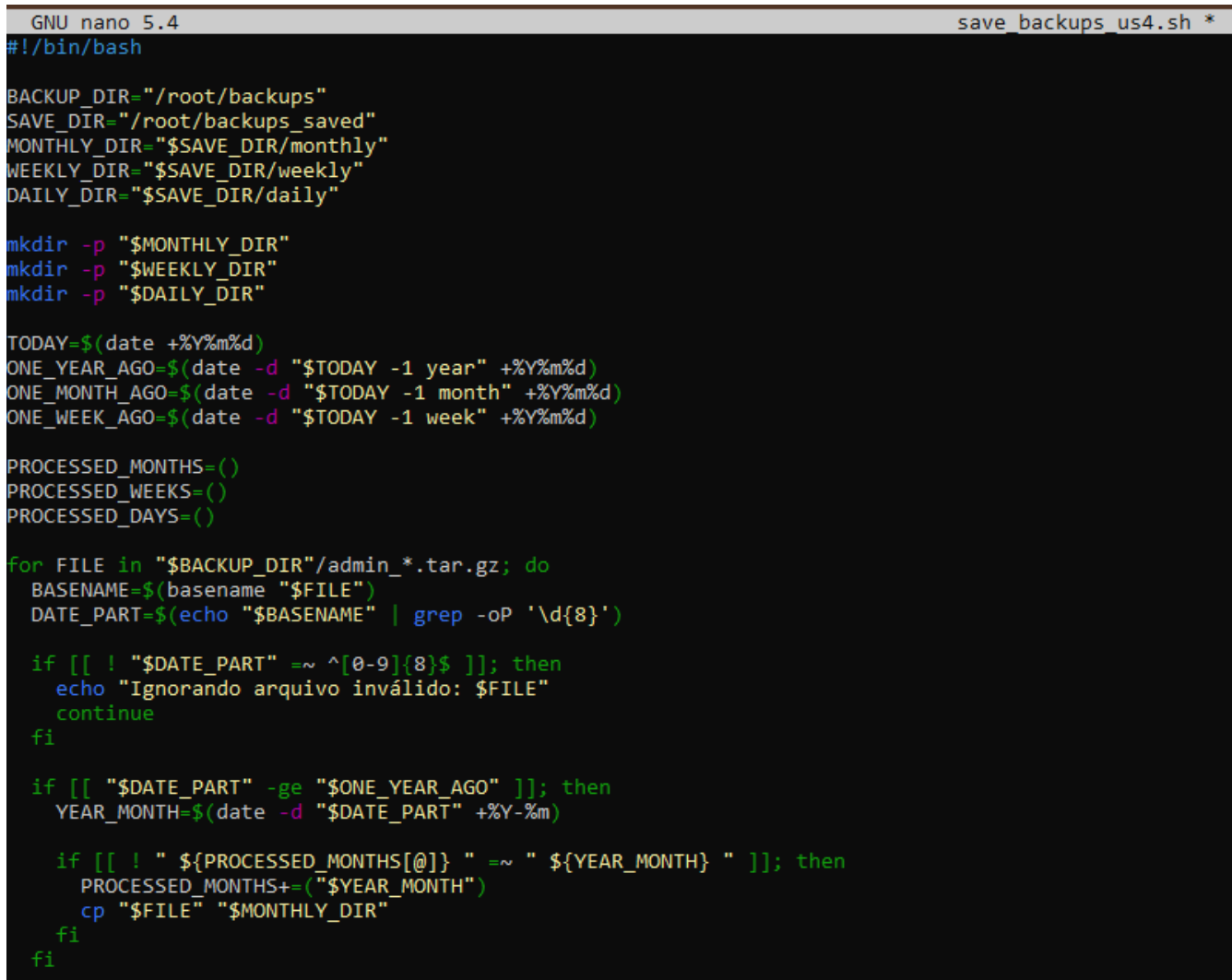
“Daily” – Onde será guardado **1 backup por dia na última semana.**

“Weekly” – Onde será guardado **1 backup por semana no último mês.**

“Monthly” – Onde será guardado **1 backup por mês no último ano.**

Script:

Para isso, foi criado o script: “save_backups_us4.sh”:



```
GNU nano 5.4 save_backups_us4.sh *
#!/bin/bash

BACKUP_DIR="/root/backups"
SAVE_DIR="/root/backups_saved"
MONTHLY_DIR="$SAVE_DIR/monthly"
WEEKLY_DIR="$SAVE_DIR/weekly"
DAILY_DIR="$SAVE_DIR/daily"

mkdir -p "$MONTHLY_DIR"
mkdir -p "$WEEKLY_DIR"
mkdir -p "$DAILY_DIR"

TODAY=$(date +%Y%m%d)
ONE_YEAR_AGO=$(date -d "$TODAY -1 year" +%Y%m%d)
ONE_MONTH_AGO=$(date -d "$TODAY -1 month" +%Y%m%d)
ONE_WEEK_AGO=$(date -d "$TODAY -1 week" +%Y%m%d)

PROCESSED_MONTHS=()
PROCESSED_WEEKS=()
PROCESSED_DAYS=()

for FILE in "$BACKUP_DIR"/admin_*.tar.gz; do
    BASENAME=$(basename "$FILE")
    DATE_PART=$(echo "$BASENAME" | grep -oP '\d{8}')

    if [[ ! "$DATE_PART" =~ ^[0-9]{8}$ ]]; then
        echo "Ignorando arquivo inválido: $FILE"
        continue
    fi

    if [[ "$DATE_PART" -ge "$ONE_YEAR_AGO" ]]; then
        YEAR_MONTH=$(date -d "$DATE_PART" +%Y-%m)

        if [[ ! "${PROCESSED_MONTHS[@]}" =~ "$YEAR_MONTH" ]]; then
            PROCESSED_MONTHS+=("$YEAR_MONTH")
            cp "$FILE" "$MONTHLY_DIR"
        fi
    fi
fi
```

Figura 8 – Parte 1 do Script criado para guardar backups

```

if [[ "$DATE_PART" -ge "$ONE_MONTH_AGO" ]]; then
    WEEK_NUMBER=$(date -d "$DATE_PART" +%U)

    if [[ ! " ${PROCESSED_WEEKS[@]} " =~ " ${WEEK_NUMBER} " ]]; then
        PROCESSED_WEEKS+=("$WEEK_NUMBER")
        cp "$FILE" "$WEEKLY_DIR"
    fi
fi

if [[ "$DATE_PART" -ge "$ONE_WEEK_AGO" ]]; then
    DAY=$(date -d "$DATE_PART" +%Y-%m-%d)

    if [[ ! " ${PROCESSED_DAYS[@]} " =~ " ${DAY} " ]]; then
        PROCESSED_DAYS+=("$DAY")
        cp "$FILE" "$DAILY_DIR"
    fi
fi
done
echo "Gestão de backups concluída!"

```

Figura 9 - Parte 2 do Script criado para guardar backups

Explicação do script:

```

BACKUP_DIR="/root/backups"
SAVE_DIR="/root/backups_saved"
MONTHLY_DIR="$SAVE_DIR/monthly"
WEEKLY_DIR="$SAVE_DIR/weekly"
DAILY_DIR="$SAVE_DIR/daily"

mkdir -p "$MONTHLY_DIR"
mkdir -p "$WEEKLY_DIR"
mkdir -p "$DAILY_DIR"

```

Figura 10 - Parte do script onde se define as variáveis

Na primeira parte do script, definimos o diretório onde se encontram os backups armazenados – **BACKUP_DIR**, e definimos onde vão ser armazenados os backups gerenciados – **SAVE_DIR**. De seguida criamos 3 diretórios (caso estes não existam), para os backups mensais – **MONTHLY_DIR** semanais – **WEEKLY_DIR** e diários – **DAILY_DIR**, respetivamente.

```

TODAY=$(date +%Y%m%d)
ONE_YEAR_AGO=$(date -d "$TODAY -1 year" +%Y%m%d)
ONE_MONTH_AGO=$(date -d "$TODAY -1 month" +%Y%m%d)
ONE_WEEK_AGO=$(date -d "$TODAY -1 week" +%Y%m%d)

```

Figura 11 - Parte do script onde se criam as datas de referência

De seguida criamos as datas de referência que serão uteis na lógica do script.

TODAY=\$(date +%Y%m%d): Obtém a data de hoje no formato YYYYMMDD.

ONE_YEAR_AGO, ONE_MONTH_AGO, ONE_WEEK_AGO: Calcula as datas de um ano, um mês e uma semana atrás, respetivamente.

```
PROCESSED_MONTHS=()
PROCESSED_WEEKS=()
PROCESSED_DAYS=()
```

Figura 12 - Parte do script onde se criam as variáveis para evitar backups duplicados

Foram implementadas estas variáveis porque o que estava a acontecer era que o script estava a guardar backups duplicados no mesmo mês. E esta foi a solução implementada para resolver o problema.

```
for FILE in "$BACKUP_DIR"/admin_*.tar.gz; do
    BASENAME=$(basename "$FILE")
    DATE_PART=$(echo "$BASENAME" | grep -oP '\d{8}')

    if [[ ! "$DATE_PART" =~ ^[0-9]{8}$ ]]; then
        echo "Ignorando arquivo inválido: $FILE"
        continue
    fi
done
```

Figura 13 - Parte do script onde se percorre todos os ficheiros que vão ser processados

De seguida criamos o loop que percorre todos os ficheiros que se encontrem no formato dos backups: “admin_YYYYmmdd.tar.gz” o “*” é uma wildcard.

Para cada arquivo, o nome do arquivo (BASENAME) é extraído e a data (DATE_PART) é extraída do nome usando a expressão regular: ‘grep -oP '\d{8}’.

O script verifica ainda se a data extraída do nome do arquivo é válida, ou seja, se possui o formato YYYYMMDD. Caso contrário, o arquivo é ignorado.

```
if [[ "$DATE_PART" -ge "$ONE_YEAR_AGO" ]]; then
    YEAR_MONTH=$(date -d "$DATE_PART" +%Y-%m)

    if [[ ! "${PROCESSED_MONTHS[@]}" =~ "$YEAR_MONTH" ]]; then
        PROCESSED_MONTHS+=("$YEAR_MONTH")
        cp "$FILE" "$MONTHLY_DIR"
    fi
fi
```

Figura 14 - Parte do Script onde se verifica os ficheiros mensais

Nesta parte do script realiza-se o processamento dos backups mensais:

Se o arquivo de backup for de um ano atrás ou mais recente (**DATE_PART -ge ONE_YEAR_AGO**), o script verifica se já existe um backup mensal para o mês correspondente.

Caso não exista, o arquivo é copiado para o diretório de backups mensais. Verificamos ainda se o mês já foi processado para evitar backups duplicados.

```
if [[ "$DATE_PART" -ge "$ONE_MONTH_AGO" ]]; then
    WEEK_NUMBER=$(date -d "$DATE_PART" +%U)

    if [[ ! "${PROCESSED_WEEKS[@]}" =~ "$WEEK_NUMBER" ]]; then
        PROCESSED_WEEKS+=("$WEEK_NUMBER")
        cp "$FILE" "$WEEKLY_DIR"
    fi
fi
```

Figura 15 - Parte do Script onde se realiza a verificação de ficheiros semanais

Nesta parte do script realiza-se o processamento dos backups semanais:

Se o arquivo de backup for do último mês ou mais recente (**DATE_PART -ge ONE_MONTH_AGO**), o script verifica se já existe um backup semanal para a semana correspondente.

Caso não exista, o arquivo é copiado para o diretório de backups semanais. Verificamos ainda se a semana já foi processada, para evitar backups duplicados.

```
if [[ "$DATE_PART" -ge "$ONE_WEEK_AGO" ]]; then
    DAY=$(date -d "$DATE_PART" +%Y-%m-%d)

    if [[ ! "${PROCESSED_DAYS[@]}" =~ "$DAY" ]]; then
        PROCESSED_DAYS+=("$DAY")
        cp "$FILE" "$DAILY_DIR"
    fi
fi
done

echo "Gestão de backups concluída!"
```

Figura 16 - Parte do Script onde se realiza a verificação de ficheiros diários

Por fim o script realiza o processamento dos backups diários na última semana:

Se o arquivo de backup for da última semana ou mais recente (**DATE_PART -ge ONE_WEEK_AGO**), o arquivo é copiado para o diretório de backups diários. Verificamos ainda se o backup já foi processado para evitar backups duplicados.

Após o processamento de todos os arquivos, o script imprime a mensagem "Gestão de backups concluída!".

Para concluir bastava apenas executar o script com o seguinte comando: **./save_backups_us4.sh**

Resultado Após Execução: 04/01/2025

```
root@vs405:~# ./save_backups_us4.sh
Gestão de backups concluída!
```

Figura 17 - Execução do script

Verificamos o diretório com os backups que vão ser geridos com o comando: **'cd backups'**

```
root@vs405:~# cd backups
root@vs405:~/backups# ls
admin_20241229.tar.gz  admin_20241230.tar.gz  admin_20241231.tar.gz  admin_20250101.tar.gz  admin_20250102.tar.gz  admin_20250103.tar.gz  admin_20250104.tar.gz
```

Figura 18 - Ficheiros que irão ser processados

Verificamos agora o diretório onde serão copiados os backups em conformidade com os requisitos desta user stor, com o comando: **'cd backups_saved'**.

Daily:

```
root@vs405:~/backups_saved/daily# ls
admin_20241229.tar.gz  admin_20241230.tar.gz  admin_20241231.tar.gz  admin_20250101.tar.gz  admin_20250102.tar.gz  admin_20250103.tar.gz  admin_20250104.tar.gz
```

Figura 19 - Ficheiros pós processamento no diretório Daily

Aqui estão guardados 1 backup por dia na última semana.

Weekly:

```
root@vs405:~/backups_saved/weekly# ls
admin_20241229.tar.gz  admin_20250101.tar.gz
```

Figura 20 - Ficheiros pós processamento no diretório Weekly

Aqui estão guardados 1 backup por semana no último mês.

Monthly:


```
root@vs405:~/backups_saved/monthly# ls
admin_20241229.tar.gz  admin_20250101.tar.gz
```

Figura 21 - Ficheiros pós processamento no diretório Monthly

Aqui estão guardados 1 backup por mês no último ano.

Execução do Script Automaticamente:

Para a execução automática deste script foi adicionado uma entrada na cron table:



```
GNU nano 5.4 /tmp/crontab.alU0Xw/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
* * * * * /usr/local/bin/generate-ssh-banner.sh
0 2 * * * /bin/bash /root/backupDBUS3.sh
1 2 * * * /bin/bash /root/remove_old_backups.sh
0 3 * * * /root/verifyBackups.sh > /dev/null 2>&1
1 2 * * * /root/save_backups_us4.sh
```

Figura 22 - Adição de uma entrada no crontab para execução do script automaticamente

Este script irá ser executado todos os dias às 02:01h da manhã, imediatamente a seguir da execução do backup diário.

Teste da Funcionalidade – 04/01/2025:

Para realizar o teste à funcionalidade, foi criado um script de teste igual ao anterior, porém só foi alterado o repositório onde estão a ser guardados os backups e o repositório onde irão ser guardados os backups pós processamento.

Foi então criado o script: **save_backups_teste.sh**, com os seguintes diretórios:

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

# Diretórios de origem e destino
BACKUP_DIR="/tmp/test_backups/backups"
SAVE_DIR="/tmp/test_backups/backups_saved"
```

Figura 23 - Criação de um script de teste com alteração dos diretórios para diretórios de teste

No diretório: **/tmp/test_backups/backups**, foram criados os seguintes backups de teste:

```
root@vs405:/tmp/test_backups/backups# ls
admin_backups_20240101.tar.gz  admin_backups_20240201.tar.gz  admin_backups_20241225.tar.gz  admin_backups_20241226.tar.gz  admin_backups_20250104.tar.gz
```

Figura 24 - Criação de backups de teste

De seguida executamos o script de teste:

```
root@vs405:~# ./save_backups_teste.sh
Gestão de backups concluída!
```

Figura 25 - Execução do script de teste

E verificamos o diretório de armazenamento dos backups pós processamento:

```
root@vs405:~# cd /tmp/test_backups/backups_saved/
root@vs405:/tmp/test_backups/backups_saved# ls
daily  monthly  weekly
```

Figura 26 - Verificação do diretório de teste pós execução

Verificamos o diretório: Daily (1 Backup por dia na última semana).

```
root@vs405:/tmp/test_backups/backups_saved/daily# ls
admin_backups_20250104.tar.gz
```

Figura 27 - Backups pós processamento no diretório de teste Daily

Verificamos o diretório: Weekly (1 Backup por semana no último mês).

```
root@vs405:/tmp/test_backups/backups_saved/weekly# ls
admin_backups_20241225.tar.gz  admin_backups_20250104.tar.gz
```

Figura 28 - Backups pós processamento no diretório de teste Weekly

Verificamos o diretório: Monthly (1 Backup por mês no último ano).

```
root@vs405:/tmp/test_backups/backups_saved/monthly# ls
admin_backups_20240201.tar.gz  admin_backups_20241225.tar.gz  admin_backups_20250104.tar.gz
```

Figura 29 - Backups pós processamento no diretório de teste Monthly

User Story 5

Pedido: Como administrador de sistemas quero que o processo da US C3 seja mantido no log do Linux, num contexto adequado, e alertado o administrador no acesso à consola se ocorrer uma falha grave neste processo.

Aluno Responsável: João Pereira (1211503)

Solução: Para realizar esta User Story, foi necessário alterar o script já criado na US3 – backupDBUS3.sh.

```
GNU nano 5.4 backupDBUS3.sh *
#!/bin/bash

# Configurações
MONGO_HOST="vs884.dei.isep.ipp.pt"
MONGO_PORT="27017"
MONGO_USER="mongoadmin"
MONGO_PASS="fb2760583afec14f39c8aa64"
AUTH_DB="admin"
DB_NAME="admin"
BACKUP_DIR="/root/backups"
DATA_ATUAL=$(date +%Y%m%d)
FILENAME="${DB_NAME}_${DATA_ATUAL}.tar.gz"
LOG_FILE="/var/log/mongo_backup.log"
FAILURES_FILE="/var/log/backup_failures.log"

# Função para registar no log
log_message() {
    local MESSAGE=$1
    echo "$(date +%Y-%m-%d %H:%M:%S) - $MESSAGE" >> $LOG_FILE
    logger "$MESSAGE" # Adiciona no log do sistema
}

# Função para registar falhas críticas
register_failure() {
    local MESSAGE=$1
    echo "$(date +%Y-%m-%d %H:%M:%S) - $MESSAGE" >> $FAILURES_FILE
}

# Realizar o backup da nossa base de dados
log_message "A iniciar o backup da base de dados '$DB_NAME'."
mongodump --host $MONGO_HOST --port $MONGO_PORT -u $MONGO_USER -p $MONGO_PASS --authenticationDatabase $AUTH_DB --db $DB_NAME --out "$BACKUP_DIR/$DB_NAME"

# Verificar se o dump foi bem-sucedido
if [ $? -eq 0 ]; then
    log_message "Backup da base de dados '$DB_NAME' concluído com sucesso."

    # Compactar o backup usando tar
    log_message "Iniciar a compactação do backup."
    tar -czf "$BACKUP_DIR/$FILENAME" -C "$BACKUP_DIR" "$DB_NAME"
fi
```

Figura 30 - Parte 1 do script criado na US3 modificado

```
# Verificar se a compactação foi bem-sucedida
if [ $? -eq 0 ]; then
    log_message "Backup compactado com sucesso em '$FILENAME'."

    # Remover a pasta de backup não compactada
    rm -rf "$BACKUP_DIR/$DB_NAME"
    log_message "Pasta de backup não compactada removida."
else
    FAILURE_MESSAGE="Falha na compactação do backup da base de dados '$DB_NAME'."
    log_message "$FAILURE_MESSAGE"
    register_failure "$FAILURE_MESSAGE"
fi
else
    FAILURE_MESSAGE="Falha no backup da base de dados '$DB_NAME'."
    log_message "$FAILURE_MESSAGE"
    register_failure "$FAILURE_MESSAGE"
fi
```

Figura 31 - Parte 2 do script criado na US3 modificado

Alteração do ficheiro

Primeiramente foram adicionados os diretórios onde vão ser registados os logs e as falhas

```
LOG_FILE="/var/log/mongo_backup.log"
FAILURES_FILE="/var/log/backup_failures.log"
```

Figura 32 - Primeira alteração: Criação de Logs files e Failures Files

Foi adicionado uma função para registrar mensagens no log:

```
# Função para registrar no log
log_message() {
    local MESSAGE=$1
    echo "$(date +%Y-%m-%d %H:%M:%S') - $MESSAGE" >> $LOG_FILE
    logger "$MESSAGE" # Adiciona no log do sistema
}
```

Figura 33 - Segunda alteração: Criação de uma função para registrar mensagens de log

E outra função para registrar as falhas críticas:

```
# Função para registrar falhas críticas
register_failure() {
    local MESSAGE=$1
    echo "$(date +%Y-%m-%d %H:%M:%S') - $MESSAGE" >> $FAILURES_FILE
}
```

Figura 34 - Terceira alteração: Criação de uma função para registrar falhas durante a execução do script

O script vai registrando nos logs o nível de execução e em caso de falha chama a função “register_failure()”

```
# Verificar se o dump foi bem-sucedido
if [ $? -eq 0 ]; then
    log_message "Backup da base de dados '$DB_NAME' concluído com sucesso."

    # Compactar o backup usando tar
    log_message "Iniciar a compactação do backup."
    tar -czf "$BACKUP_DIR/$FILENAME" -C "$BACKUP_DIR" "$DB_NAME"

    # Verificar se a compactação foi bem-sucedida
    if [ $? -eq 0 ]; then
        log_message "Backup compactado com sucesso em '$FILENAME'."

        # Remover a pasta de backup não compactada
        rm -rf "$BACKUP_DIR/$DB_NAME"
        log_message "Pasta de backup não compactada removida."
    else
        FAILURE_MESSAGE="Falha na compactação do backup da base de dados '$DB_NAME'"
        log_message "$FAILURE_MESSAGE"
        register_failure "$FAILURE_MESSAGE"
    fi
else
    FAILURE_MESSAGE="Falha no backup da base de dados '$DB_NAME'."
    log_message "$FAILURE_MESSAGE"
    register_failure "$FAILURE_MESSAGE"
fi
```

Figura 35 - Demonstração da utilização das funções ao longo do script

Informar o administrador:

Caso exista alguma falha crítica durante a execução do backup uma mensagem será exibida para o administrador quando ele fizer login. Para que isto seja implementado, foi criado um script no diretório “etc/profile.d”.

```
root@vs405:~# nano backup_failures.sh
```

Figura 36 - Criação de um script para alertar o administrador caso haja alguma falha durante a execução do script

Script criado que irá alertar o administrador caso alguma falha tenha acontecido no backup da base de dados

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

FAILURES_FILE="/var/log/backup_failures.log"

# Verificar se o arquivo de falhas existe e não está vazio
if [ -s "$FAILURES_FILE" ]; then
    echo -e "\033[31m=== ALERTA: FALHAS NOS BACKUPS ===\033[0m"
    cat "$FAILURES_FILE"
    echo -e "\033[31m===== \033[0m"

    # Apagar o conteúdo do arquivo após exibição
    > "$FAILURES_FILE"
else
    echo "Nenhuma falha nos backups registrada."
fi
```

Figura 37 - Criação do script para alertar o administrador

Demonstração da Funcionalidade:

Para demonstrar o funcionamento desta US, foi modificado o nome do servidor da base de dados para “serverErrado” para que este não realize o backup corretamente.

```
# Configurações
MONGO_HOST="serverErrado"
MONGO_PORT="27017"
```

Figura 38 - Modificação no script para gerar uma falha propositalmente

Após a execução do script deu erro, como previsto:

```
root@vs405:~# ./backupDBUS3.sh
2024-12-18T16:38:46.945+0000 Failed: can't create session: could not connect to server: server selection error: server selection timeout, current topology: { Type: Single, Servers: [{ Addr: servererrado:27017, Type: Unknown, Last error: connection() error occurred during connection handshake: dial tcp: lookup servererrado on 192.168.62.32:53: no such host }, ] }
```

Figura 39 - Erro na execução do script, como previsto

Após o administrador se autenticar, aparece uma mensagem com as falhas dos backups:

```

joahcpereira@DESKTOP-DDV0UMO:/mnt/c/Users/Utilizador$ ssh root@vs405.dei.isep.ipp.pt
Welcome to the server!
Current Date: 2024-12-18 16:39:01
Users Logged In: 0

root@vs405.dei.isep.ipp.pt's password:
Linux vs405 5.4.0-173-generic #191-Ubuntu SMP Fri Feb 2 13:55:07 UTC 2024 x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.

Last login: Wed Dec 18 16:37:04 2024 from 10.8.254.61
Hello root, today's date is Wed 18 Dec 2024 04:39:06 PM UTC
=== ALERTA: FALHAS NOS BACKUPS ===
2024-12-18 16:38:46 - Falha no backup da base de dados 'admin'.
=====
root@vs405:~#

```

Figura 40 - Mensagem de erro apresentada ao administrador após a falha na criação do backup

Após o administrador se autenticar e vir esta mensagem, o histórico de falhas é apagado. Desta forma, o administrador será informado de falhas e o sistema manterá o log limpo após cada login.

Este é o caso de uma autenticação do administrador, caso não haja nenhum problema com os backups:

```

joahcpereira@DESKTOP-DDV0UMO:/mnt/c/Users/Utilizador$ ssh root@vs405.dei.isep.ipp.pt
Welcome to the server!
Current Date: 2024-12-18 17:32:01
Users Logged In: 1

root@vs405.dei.isep.ipp.pt's password:
Linux vs405 5.4.0-173-generic #191-Ubuntu SMP Fri Feb 2 13:55:07 UTC 2024 x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.

Last login: Wed Dec 18 17:11:13 2024 from 10.8.254.61
Hello root, today's date is Wed 18 Dec 2024 05:32:26 PM UTC
Nenhuma falha nos backups registada.
root@vs405:~#

```

Figura 41 - Autenticação normal, caso não haja erros durante a execução dos backups

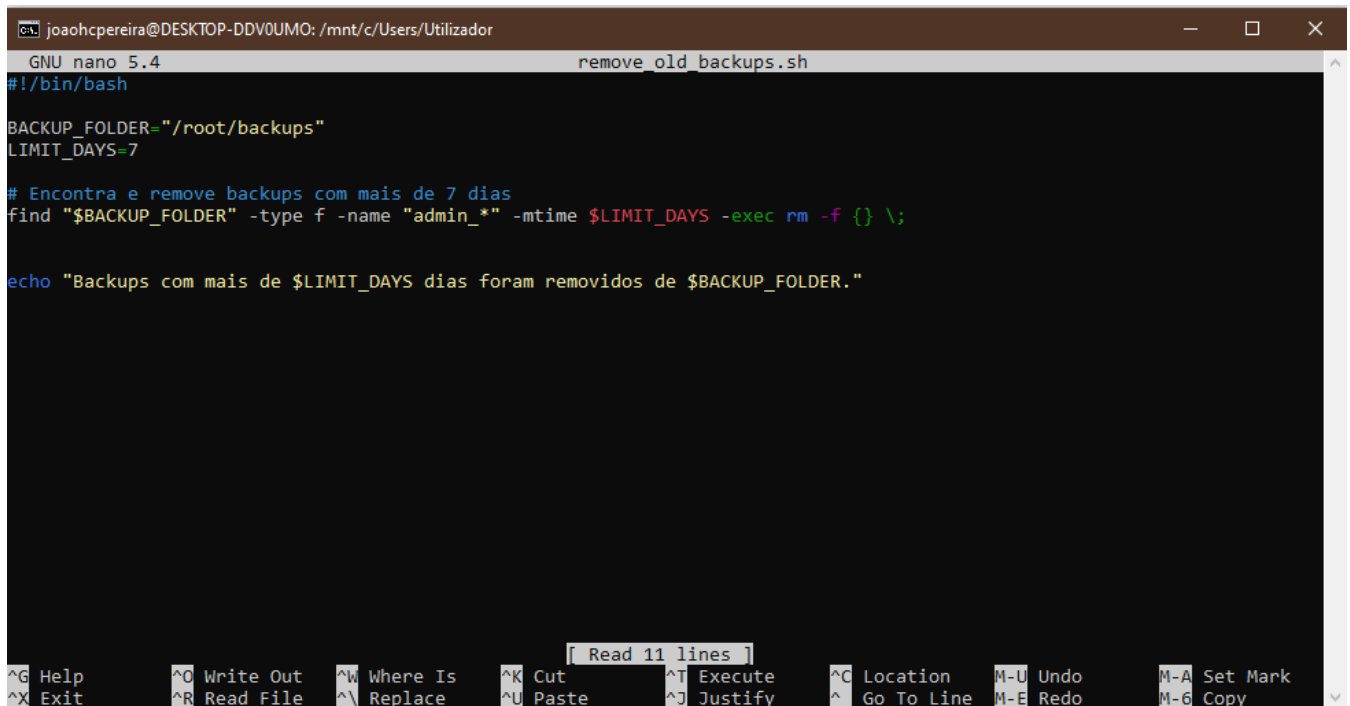
User Story 6

Pedido: Como administrador de sistemas quero que a cópia de segurança da US C3 tenha um tempo de vida não superior a 7 (sete) dias exceto no indicado na US C4.

Aluno Responsável: João Pereira (1211503)

Solução: O objetivo desta US é apagar as cópias de segurança com tempo de vida superior 7.

Para realizar esta funcionalidade foi desenvolvido o seguinte script: `remove_old_backups.sh`

A screenshot of a terminal window titled 'remove_old_backups.sh' showing the GNU nano 5.4 editor. The script content is as follows:

```
#!/bin/bash

BACKUP_FOLDER="/root/backups"
LIMIT_DAYS=7

# Encontra e remove backups com mais de 7 dias
find "$BACKUP_FOLDER" -type f -name "admin_*" -mtime $LIMIT_DAYS -exec rm -f {} \;

echo "Backups com mais de $LIMIT_DAYS dias foram removidos de $BACKUP_FOLDER."
```

The terminal window also shows the nano editor's status bar at the bottom with various shortcuts like ^G Help, ^O Write Out, ^W Where Is, ^K Cut, ^T Execute, ^C Location, M-U Undo, M-A Set Mark, ^X Exit, ^R Read File, ^_ Replace, ^U Paste, ^J Justify, ^_ Go To Line, M-E Redo, and M-6 Copy. A message '[Read 11 lines]' is visible above the status bar.

Figura 42 - Criação do script para apagar backups com tempo de vida superior a 7

Foi utilizado o comando “find” para procurar os scripts na pasta `/backups` que tenham tempo de vida superior a 7 dias. Se encontrar remove.

Para executar isto de forma automática foi adicionado uma entrada no “cron”



```
GNU nano 5.4 /tmp/crontab.6eFhfG/crontab *
# 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
* * * * * /usr/local/bin/generate-ssh-banner.sh
0 2 * * * /bin/bash /root/backupDBUS3.sh
1 2 * * * /bin/bash /root/remove_old_backups.sh
```

Figura 43 - Adição de uma entrada no crontab para executar o script automaticamente

Este script irá ser executado automaticamente às 02:01min todos os dias.

Demonstração da Funcionalidade

Para demonstrar a funcionalidade desta User Story, vamos proceder à realização de um caso de teste.

Primeiramente iremos executar o script para criar um backup da base de dados

```
root@vs405:~# ./backupDBUS3.sh
2024-12-18T19:46:23.637+0000 writing admin.system.users to /root/backups/admin/admin/system.users.bson
2024-12-18T19:46:23.639+0000 done dumping admin.system.users (1 document)
2024-12-18T19:46:23.641+0000 writing admin.system.version to /root/backups/admin/admin/system.version.bson
2024-12-18T19:46:23.643+0000 done dumping admin.system.version (2 documents)
2024-12-18T19:46:23.645+0000 writing admin.patientmedicalhistories to /root/backups/admin/admin/patientmedicalhistories.bson
2024-12-18T19:46:23.648+0000 writing admin.specializations to /root/backups/admin/admin/specializations.bson
2024-12-18T19:46:23.650+0000 writing admin.medicalconditions to /root/backups/admin/admin/medicalconditions.bson
2024-12-18T19:46:23.650+0000 writing admin.allergies to /root/backups/admin/admin/allergies.bson
2024-12-18T19:46:23.659+0000 done dumping admin.specializations (6 documents)
2024-12-18T19:46:23.661+0000 done dumping admin.patientmedicalhistories (13 documents)
2024-12-18T19:46:23.662+0000 done dumping admin.medicalconditions (4 documents)
2024-12-18T19:46:23.662+0000 done dumping admin.allergies (3 documents)
2024-12-18T19:46:23.662+0000 writing admin.appointments to /root/backups/admin/admin/appointments.bson
2024-12-18T19:46:23.664+0000 done dumping admin.appointments (0 documents)
```

Figura 44 - Criação de um backup com o script criado na US3

O script foi realizado com sucesso e foi criado um ficheiro .tar no diretório /backups

```
root@vs405:~/backups# ls
admin_20241218.tar.gz
```

Figura 45 - Verificação do backup criado

Agora, podemos alterar a data da criação deste ficheiro manualmente, com o comando “touch -d”
Alteramos a data de criação deste ficheiro para 10 dias atrás.

```
admin_20241218.tar.gz
root@vs405:~/backups# touch -d "10 days ago" admin_20241218.tar.gz
```

Figura 46 - Alterar a data de criação do backup para 10 dias atrás

De seguida iremos executar o script “remove_old_backups.sh”

```
root@vs405:~# ./remove_old_backups.sh
Backups com mais de 7 dias foram removidos de /root/backups.
```

Figura 47 - Execução do script para remover backups antigos

Voltamos agora ao diretório /backups e verificamos que o ficheiro foi apagado:

```
root@vs405:~# cd backups
root@vs405:~/backups# ls
root@vs405:~/backups#
```

Figura 48 - Nova verificação na pasta de backups, desta vez o script criado anteriormente encontra-se apagado visto que tinha tempo superior de 7 dias

Demonstração sequencial de todos os passos referidos anteriormente:

```
root@vs405:~# ./backupDBUS3.sh
2024-12-18T19:46:23.637+0000 writing admin.system.users to /root/backups/admin/admin/system.users.bson
2024-12-18T19:46:23.639+0000 done dumping admin.system.users (1 document)
2024-12-18T19:46:23.641+0000 writing admin.system.version to /root/backups/admin/admin/system.version.bson
2024-12-18T19:46:23.643+0000 done dumping admin.system.version (2 documents)
2024-12-18T19:46:23.645+0000 writing admin.patientmedicalhistories to /root/backups/admin/admin/patientmedicalhistories.bson
2024-12-18T19:46:23.648+0000 writing admin.specializations to /root/backups/admin/admin/specializations.bson
2024-12-18T19:46:23.650+0000 writing admin.medicalconditions to /root/backups/admin/admin/medicalconditions.bson
2024-12-18T19:46:23.650+0000 writing admin.allergies to /root/backups/admin/admin/allergies.bson
2024-12-18T19:46:23.659+0000 done dumping admin.specializations (6 documents)
2024-12-18T19:46:23.661+0000 done dumping admin.patientmedicalhistories (13 documents)
2024-12-18T19:46:23.662+0000 done dumping admin.medicalconditions (4 documents)
2024-12-18T19:46:23.662+0000 done dumping admin.allergies (3 documents)
2024-12-18T19:46:23.662+0000 writing admin.appointments to /root/backups/admin/admin/appointments.bson
2024-12-18T19:46:23.664+0000 done dumping admin.appointments (0 documents)
root@vs405:~# cd backups
root@vs405:~/backups# ls
admin_20241218.tar.gz
root@vs405:~/backups# touch -d "10 days ago" admin_20241218.tar.gz
root@vs405:~/backups# cd ..
root@vs405:~# ./remove_old_backups.sh
Backups com mais de 7 dias foram removidos de /root/backups.
root@vs405:~# cd backups
root@vs405:~/backups# ls
root@vs405:~/backups#
```

Figura 49 - Demonstração sequencial da execução do teste

User Story 7

Pedido: Como administrador da organização quero que me seja apresentado um BIA (Business Impact Analysis) da solução final, adaptando-se e onde aplicável o(s) risco(s) identificados no sprint anterior.

Aluno Responsável: Jack Pinheiro (1120419)

Solução: O *Business Impact Analysis* (BIA) tem como objetivo identificar as atividades críticas da organização e as suas dependências. Permite desta forma priorizar as operações de recuperação após uma interrupção. É composta pelos seguintes passos:

1. **Determinação dos processos críticos de negócio e da criticidade da recuperação.**

Os processos de negócio suportados pelo sistema são identificados, assim como o impacto de uma interrupção desses sistemas. Avalia-se o impacto de interrupções e o tempo estimado de indisponibilidade, que deve refletir o máximo de tempo que a organização pode tolerar sem comprometer o seu negócio.

2. **Identificação dos requisitos de recursos.**

Esforços de uma recuperação realistas exigem uma avaliação minuciosa dos recursos necessários para retomar os processos de negócio e as suas interdependências relacionadas o mais rapidamente possível.

3. **Identificação das prioridades de recuperação para os recursos do sistema.**

Com base nos resultados das atividades anteriores, os recursos do sistema podem ser mais claramente associados aos processos de negócio críticos. Podem ser definidos níveis de prioridade para sequenciar as atividades e recursos de recuperação.

Identificação dos processos críticos de negócio:

- Gestão de Dados Sensíveis dos Pacientes (Base de Dados)
- Sistema de Agendamento de Cirurgias

Análise de risco:

No sprint anterior já foi realizada esta tarefa:

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

Figura 50 - Matriz de risco referente a situações/possibilidades encontradas no sprint 2

Análise de impacto:

- Maximum Tolerable Downtime (MTD):**
 Representa o tempo total que os gestores estão dispostos a aceitar para a interrupção/falha de um processo de negócio.
- Recovery Time Objective (RTO):**
 Define o tempo máximo que um recurso do sistema pode permanecer indisponível antes de haver um impacto inaceitável noutros recursos do sistema, nos processos de negócio e no MTD.
- Recovery Point Objective (RPO):**
 Representa o ponto no tempo, anterior a uma interrupção/falha no sistema, até ao qual os dados do processo de negócio devem ser recuperados após uma falha.

Threat	MTD	RTO	RPO
Leak/Data Breach que compromete os dados	72h	12h	1h

sensíveis dos pacientes/staff			
Interdição de acesso ao sistema devido a um ciberataque	48h	8h	1h
Acesso não autorizado ao sistema (dados sensíveis)	36h	6h	4h
Perda de dados de pacientes devido a falha de backup	24h	4h	1h
Impedimento do acesso autorizado devido a falhas do serviço VPN	16h	2h	1h
Acesso não autorizado na pasta pública devido a permissões incorretas	24h	4h	2h
Erros/Atrasos nas atualizações, deixando o sistema exposto possíveis vulnerabilidades de segurança	48h	12h	Não aplicável

Figura 51 - Tabela referente à análise de impacto

Identificação dos requisitos de recursos

Nesta fase identificou-se os seguintes requisitos dos recursos dos processos de negócio:

- Administradores de sistemas para restaurar servidores e sistemas críticos.
 - Especialistas em *Cybersecurity* para conter quebras de segurança.
 - Sistemas de *backups* automáticos e frequentes para cumprir o RPO.
 - Ferramentas de recuperação rápida, como servidores de reserva ou ambientes *cloud*
- Identificação das prioridades de recuperação para os recursos do sistema**

Para manter a continuidade operacional dos processos de negócio foi necessário analisar as prioridades de recuperação. Optamos por as definir em três categorias: alta, média e baixa.

1. Prioridade alta

- *Leaks/Data Breaches* que comprometem os dados sensíveis dos pacientes/staff
- Interdição de acesso ao sistema devido a um ciberataque
- Perda de dados de pacientes devido a falha de backup
- Impedimento do acesso autorizado devido a falhas do serviço VPN

2. **Prioridade média**

- Acesso não autorizado ao sistema (dados sensíveis)

3. **Prioridade baixa**

- Acesso não autorizado na pasta pública devido a permissões incorretas
- Erros/Atrasos nas atualizações, deixando o sistema exposto possíveis vulnerabilidades de segurança

User Story 8

Pedido: Como administrador da organização quero que seja implementada uma gestão de acessos que satisfaça os critérios apropriados de segurança.

Aluno Responsável: Jack Pinheiro (1120419)

Solução: Consideramos que a implementação da **C.I.A. Triad** (Confidencialidade, Integridade e Disponibilidade) seja uma solução eficaz para a implementação de uma gestão de acessos que assegure os critérios de segurança.

A **Confidencialidade** garante que apenas *Users* autorizados tenham acesso a informações confidenciais.

- **Controlo de Acesso:** Implementar autenticação de dois fatores (2FA) assim como aplicar o princípio do menor privilégio, garantido que cada *User* tenha apenas os acessos necessários.
- **Criptografia dos Dados:** Utilizar **Criptografia de Disco** para proteger os dados armazenados e implementar **SSL/TLS** e **Criptografia de Ponta a Ponta** para proteger comunicações.
- **Políticas de Segurança:** Oferecer aos *Users* treinamento de sensibilização sobre as boas práticas de segurança, exigindo o uso de senhas fortes bem como alterá-las periodicamente. Implementar **monitoramento** de atividades para identificar acessos não autorizados.

A **Integridade** assegura que os dados permanecem consistentes e inalterados, garantido proteção contra modificações não autorizadas.

- **Checksums ou Funções Hash:** Uso de algoritmos de hash para gerar valores únicos dos dados e comparar os hashes para verificar se houve modificação dos dados.
- **Manutenção de Logs de Autoria:** Registo de quem acedeu, modificou ou tentou aceder aos dados e incluir *timestamps* para facilitar a análise de eventos e identificar anomalias.

A **Disponibilidade** garante que os dados e serviços estejam acessíveis sempre que sejam necessários, garantindo proteção contra interrupções e/ou falhas.

- **Redundância:** Distribuição de tráfego entre várias máquinas para evitar *overloads* e implementar *failover* automático para garantir continuidade em caso de falhas.
- **Backups e Recuperação:** Realização de *backups* periódicos e armazená-los em locais externos e testar regularmente os backups para garantir a recuperação eficiente dos dados.

-
- **Monitoramento e Alertas:** Utilização de ferramentas de **monitoramento** de sistemas para acompanhar o desempenho da rede e do hardware. Configuração de **alertas** automáticos para identificar problemas de **CPU overload**, memória ou **bandwidth**.

A implementação da **C.I.A. Triad** é essencial para garantir a segurança da informação. Ao adotar as práticas recomendadas, é possível proteger os dados contra acesso não autorizados, corrupções e indisponibilidade, garantindo que a organização esteja preparada para enfrentar ameaças de segurança.

User Story 9

Pedido: Como administrador da organização quero que seja implementado de forma justificada um sistema de clustering entre os sistemas que implementam o SPA.

Aluno Responsável: Jack Pinheiro (1120419)

Solução: Para implementarmos um sistema de clustering e solucionarmos este problema decidimos criar três novos servidores virtuais do DEI. Manteremos o nosso servidor vs405 como main server, usaremos o servidor vs718 como o *Cluster Host* (utilizando o HAProxy) e os outros dois restantes servidores como backups (vs725 e vs726).

Antes de instalarmos o HAProxy no *Cluster Host* é necessário clonar o servidor principal para que os servidores backup atuem da mesma forma.

Instalamos o *rsync* nos servidores para podermos efetuar a clonagem e em seguida corremos o seguinte comando duas vezes (uma para o servidor vs725 e outra para o vs726) no servidor principal (vs405):

```
root@vs405:~# sshpass -p "bXlH2M3q4i7oGm1Fl4Zw" rsync -aXv --progress / --exclude={"/dev/*","/proc/*","/sys/*","/tmp/*","/run/*","/mnt/*","/media/*","/lost+found"} root@vs725.dei.isep.ipp.pt:|
```

Figura 52- Clonagem da máquina principal para os dois servidores backup

```
root@vs405:~# sshpass -p "e909pxMWEIJKy3Ejm0gB" rsync -aXv --progress / --exclude={"/dev/*","/proc/*","/sys/*","/tmp/*","/run/*","/mnt/*","/media/*","/lost+found"} root@vs726.dei.isep.ipp.pt:|
```

Figura 53 - Clonagem da máquina principal para os dois servidores backup

De seguida instalamos o HAProxy no servidor vs718 que servirá então como *Cluster Host*:

```
root@vs718:~# sudo apt-get update  
sudo apt-get install haproxy|
```

Figura 54 - Instalação do HAProxy

Posteriormente é necessário a configuração do HAProxy que é feita no arquivo */etc/haproxy/haproxy.cfg*.

```

GNU nano 5.4 /etc/haproxy/haproxy.cfg *
# See: https://ssl-config.mozilla.org/#server=haproxy&server-version=2.0.3&config=inter
ssl-default-bind-ciphers ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDH
ssl-default-bind-ciphersuites TLS_AES_128_GCM_SHA256:TLS_AES_256_GCM_SHA384:TLS_CHACHA2
ssl-default-bind-options ssl-min-ver TLSv1.2 no-tls-tickets

defaults
    log      global
    mode     http
    option   httplog
    option   dontlognull
    timeout  connect 5000
    timeout  client  50000
    timeout  server  50000
    errorfile 400 /etc/haproxy/errors/400.http
    errorfile 403 /etc/haproxy/errors/403.http
    errorfile 408 /etc/haproxy/errors/408.http
    errorfile 500 /etc/haproxy/errors/500.http
    errorfile 502 /etc/haproxy/errors/502.http
    errorfile 503 /etc/haproxy/errors/503.http
    errorfile 504 /etc/haproxy/errors/504.http

# Frontend Configuration
frontend http_frontend
    bind 10.9.22.206:80
    stats enable
    mode http
    option httpclose
    option forwardfor
    default_backend http_backend

# Backend Configuration
backend http_backend
    balance roundrobin
    server server1 10.9.21.149:4200 check
    server server2 10.9.22.213:4200 check
    server server3 10.9.22.214:4200 check

```

Figura 55 - Configuração do ficheiro /etc/haproxy/haproxy.cfg

Comandos:

frontend http_frontend: Define um frontend chamado **http_frontend**, que é onde o HAProxy recebe as requisições dos clientes. O frontend é a parte do HAProxy que lida com as conexões de entrada.

bind 10.9.22.206:80: Faz o HAProxy escutar na interface de rede com o endereço IP 10.9.22.206 e na porta 80 (HTTP).

stats enable: Habilita a interface de monitoramento de status do HAProxy.

mode http: Define que o HAProxy irá operar no modo HTTP.

option httpclose: Faz com que o HAProxy feche a conexão com o cliente após cada resposta, ao invés de manter a conexão aberta para reutilização. Evita manter conexões abertas desnecessariamente.

option forwardfor: Adiciona o cabeçalho X-Forwarded-For às requisições HTTP. Esse cabeçalho carrega o endereço IP original do cliente, para o servidor *backend* saber qual o IP de origem da requisição.

default_backend http_backend: Define que as requisições recebidas pelo *frontend* **http_frontend** serão encaminhadas para o *backend* chamado **http_backend**. O *backend* é onde as requisições serão distribuídas entre os servidores.

backend http_backend: Define um *backend* chamado **http_backend**. O *backend* é onde as requisições serão enviadas após passarem pelo *frontend*. No *backend*, o **HAProxy** distribui o tráfego para os servidores que configuramos.

balance roundrobin: Define o método de balanceamento de carga. O *roundrobin* é o método mais simples, onde as requisições são distribuídas de forma sequencial entre os servidores (1ª requisição vai para o 1º servidor, 2ª para o 2º servidor, etc).

server server1 10.9.21.149:80 check: Define o primeiro servidor no *backend* (o nosso servidor principal (vs405)). O nome do servidor é `server1`, e ele tem o endereço IP 10.9.21.149, na porta 4200. A opção **check** faz com que o **HAProxy** verifique periodicamente se esse servidor está a funcionar corretamente antes de enviar tráfego para ele. Se o servidor não estiver disponível, o HAProxy não enviará requisições para ele.

O mesmo se aplica para os servidores `vs725` e `vs726` que são os nossos servidores backups.

Resumidamente, o **HAProxy** balanceia o tráfego entre estes 3 servidores usando o método *roundrobin* e verifica se estão disponíveis através do **check**.

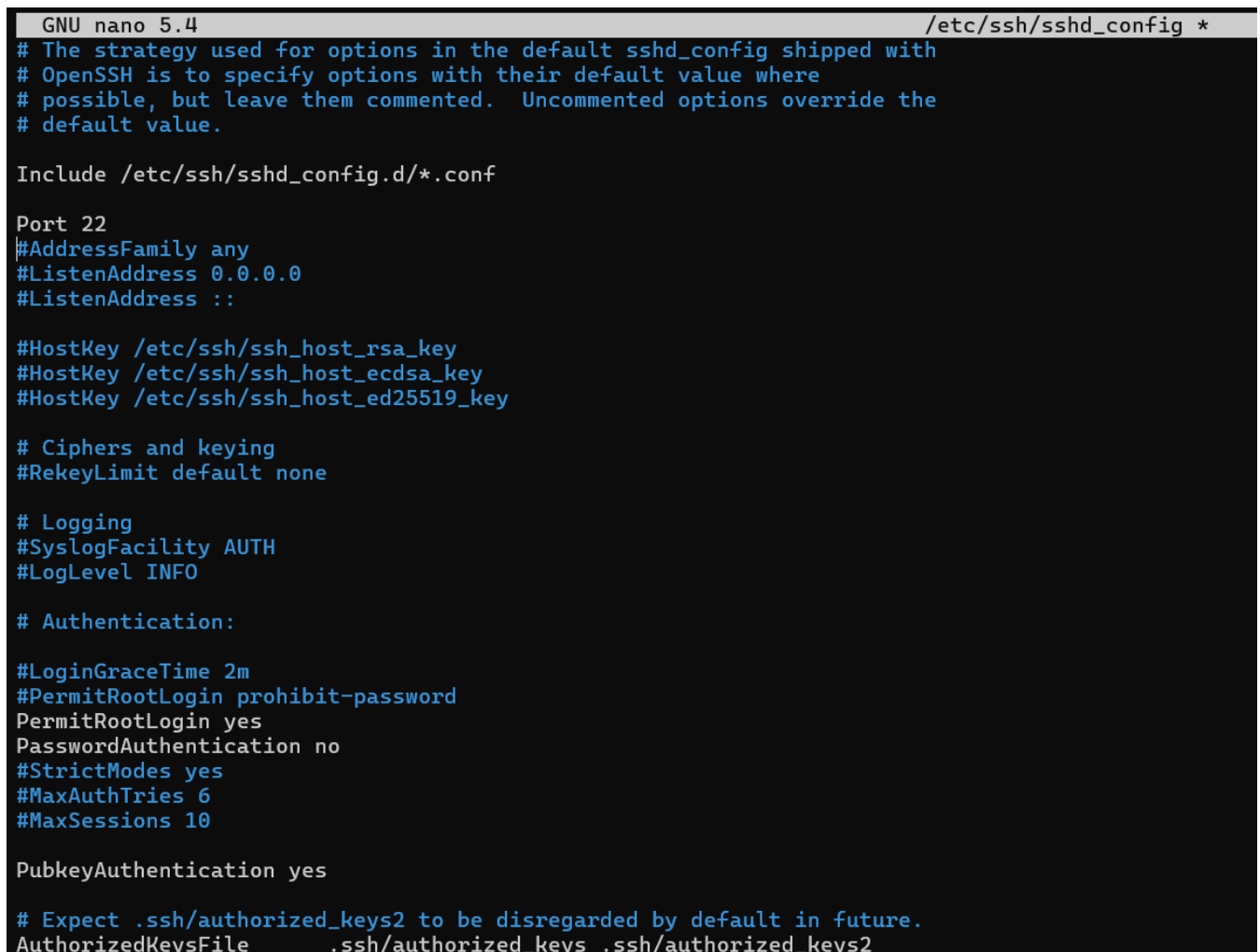
Próximo passo é reiniciar o **HAProxy** para que as alterações tenham efeito usando o comando **sudo systemctl restart haproxy** e encontra-se concluída a User Story.

User Story 10

Pedido: Como administrador de sistemas quero que o administrador tenha um acesso SSH à máquina virtual apenas por certificado, sem recurso a password.

Aluno Responsável: Jack Pinheiro (1120419)

Solução: Para a realização desta user story é necessário ir ao ficheiro `/etc/ssh/sshd_config`, que contém as configurações do servidor SSH, e verificar se determinadas permissões existem e realizar as alterações necessárias ao ficheiro para controlar o servido SSH.



```
GNU nano 5.4 /etc/ssh/sshd_config *
# The strategy used for options in the default sshd_config shipped with
# OpenSSH is to specify options with their default value where
# possible, but leave them commented. Uncommented options override the
# default value.

Include /etc/ssh/sshd_config.d/*.conf

Port 22
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::

#HostKey /etc/ssh/ssh_host_rsa_key
#HostKey /etc/ssh/ssh_host_ecdsa_key
#HostKey /etc/ssh/ssh_host_ed25519_key

# Ciphers and keying
#RekeyLimit default none

# Logging
#SyslogFacility AUTH
#LogLevel INFO

# Authentication:

#LoginGraceTime 2m
#PermitRootLogin prohibit-password
PermitRootLogin yes
PasswordAuthentication no
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10

PubkeyAuthentication yes

# Expect .ssh/authorized_keys2 to be disregarded by default in future.
AuthorizedKeysFile .ssh/authorized_keys .ssh/authorized_keys2
```

Figura 56 - Edição do ficheiro `/etc/ssh/sshd_config`

Port 22 – Porta em qual o SSH será executado, por default é o 22.

PermitRootLogin yes – Determina se é permitido fazer login diretamente como root via SSH.

PasswordAuthentication no – Verifica se é permitida autenticação por senha para conexões SSH.

PubkeyAuthentication yes – Determina se a autenticação por chave é permitida.

AuthorizedKeysFile .ssh/authorized_keys – Define o caminho para o arquivo que contém as chaves públicas dos users autorizados a se autenticar no servidor. **.ssh/authorized_keys** é o caminho padrão para esse arquivo.

Próximo passo é gerar as chaves SSH do tipo RSA no nosso computador, através do comando **ssh-keygen -t rsa**.

```
PS C:\Users\Jack-> ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (C:\Users\Jack-/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in C:\Users\Jack-/.ssh/id_rsa
Your public key has been saved in C:\Users\Jack-/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:q1KwYIiLSSHPukY2LCukTY8KX7tE8p5h10+B0EqF5CR0 jack-@JackPinheiro
The key's randomart image is:
+---[RSA 3072]-----+
|. + E
|. * =
|. + 0 .
|. +.X B
|=X X o S
|* * =.. .
|.o +.B.o.
| o 0.=o.
| +.+...
+---[SHA256]-----+
```

Figura 57 - Gerar chaves SSH com o comando `ssh-keygen -t rsa`

```
PS C:\Users\Jack-> type C:\Users\Jack-\.ssh\id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQGDdIfIeYYn0QeGIHMPsbegL80Qoq+sxj0XJ6LDK
AvbqGsEGSBVEvKEk6NUWoDN41I+REs13ip7JQ0EfBF/pHMPUQFTbNAIjN8T0gf0MJLfYPfJRp41L
PsrP0x3oA/sOHDq+MVk8L80Tlu0q1EdcVL7WyNiqG9wa36CqLGoczvnfdw4PMWZ0pKWYPZc7mFm5
NWYCIi6K08vBKaBSEyizKDV0vGBkYh9QJsoSKs0KPj0kzrI7y/LT6znJJqPU1bCb+pmm/2PXLwRt
gyw+0QW65L/JpVhq8XRu1jvxwGk7woHKWoic8bScLcvbmGmLHE/5KreWuM7GD78TcjPs7POB6Ib6
5a7hqDXVUemus7Cg9vBTRp0WJ6+P/gtN26AMEF++gh6Xc4fxLALPKY5xsFNVsa1itpozNvgkPBZz
0/45oskWeAt4Sh9frAuNaR4lkr6kCfuSAQBsAqvY8MCuPjC0QpRbuSeClvSPo1BUg3lc1Gphf/oa
GxoJ6cuJ9h9PhhutgyM= jack-@JackPinheiro
```

Figura 58 - Public key gerada

```
root@vs405:/# nano /root/.ssh/authorized_keys
root@vs405:/# nano /root/.ssh/authorized_keys.save
```

Figura 59 - Copiar a public key para o ficheiro authorized.keys e o ficheiro de backup authorized_keys.save

```
root@vs405:/# systemctl restart ssh
```

Figura 60 - Restart do serviço de ssh para implementar as alterações

```
PS C:\Users\Jack-> ssh root@vs405.dei.isep.ipp.pt
Welcome to the server!
Current Date: 2025-01-05 21:03:02
Users Logged In: 0

Linux vs405 5.4.0-173-generic #191-Ubuntu SMP Fri Feb 2 13:55:07 UTC 2024 x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.

Last login: Sun Jan  5 20:45:49 2025 from 10.8.52.29
Hello root, today's date is Sun Jan  5 09:03:30 PM UTC 2025
Nenhuma falha nos backups registada.
```

Figura 61 - teste de autenticação sem password bem sucedida

User Story 11

Pedido: Como administrador de sistemas quero que para agilização entre as várias equipas seja criada uma partilha pública de ficheiros, formato SMB/CIFS ou NFS.

Aluno Responsável: Vasco Sousa (1221700)

Solução: De forma a solucionar esta User Story, começámos por instalar os pacotes necessários para configurar o servidor Samba. Utilizámos o comando ***apt install samba***, que instala o software necessário para permitir a partilha de ficheiros entre sistemas Linux e Windows utilizando o protocolo SMB/CIFS.

Após a instalação do Samba, criámos uma pasta que servirá como diretório partilhado. Esta pasta foi criada com o comando ***mkdir /etc/sharedFolder***, e para aceder à mesma utilizámos ***cd /etc/sharedFolder***. Foi também gerado um ficheiro vazio de exemplo com o comando ***touch text***, para demonstrar como adicionar conteúdos à pasta partilhada.

De seguida, instalámos o cliente Samba, que permite aceder a partilhas SMB/CIFS a partir da linha de comandos. A instalação foi feita com ***apt install smbclient -y***.

Para gerir os acessos à partilha, foi adicionado um utilizador ao sistema com ***adduser vasco***. Durante a criação deste utilizador, foram fornecidos detalhes como a palavra-passe, embora os campos adicionais possam ser ignorados ao pressionar "Enter". Posteriormente, configurámos uma palavra-passe Samba para este utilizador através do comando ***smbpasswd -a vasco***, assegurando que apenas utilizadores autorizados poderão aceder à partilha.

```
root@vs405:/# smbpasswd -a vasco
New SMB password:
Retype new SMB password:
Added user vasco.
```

Figura 62 - Criação de uma password Samba para o utilizador vasco

Com os utilizadores configurados, testámos a partilha utilizando o comando ***smbclient -L //localhost -U vasco***, onde foram listadas as partilhas disponíveis. Por fim, reiniciámos os serviços do Samba para aplicar as alterações feitas, utilizando os comandos ***systemctl restart nmbd*** e ***systemctl restart smbd***.

```
root@vs405:/# smbclient -L //localhost -U vasco
Enter WORKGROUP\vasco's password:

      Sharename      Type      Comment
      -----      -
      print$         Disk      Printer Drivers
      PublicShare     Disk      Public Share
      IPC$           IPC       IPC Service (Samba 4.13.13-Debian)
SMB1 disabled -- no workgroup available
```

Figura 63 - Teste da partilha utilizando o comando smbclient -L

De forma a personalizar as configurações da partilha, editámos o ficheiro de configuração do Samba localizado em `/etc/samba/smb.conf` utilizando o comando `nano /etc/samba/smb.conf`. Aqui, definimos as permissões e parâmetros necessários para a pasta partilhada.

```
GNU nano 5.4 /etc/samba/smb.conf
; read only = yes

# Un-comment the following and create the profiles directory to store
# users profiles (see the "logon path" option above)
# (you need to configure samba to act as a domain controller too.)
# The path below should be writable by all users so that their
# profile directory may be created the first time they log on
[profiles]
; comment = Users Profiles
; path = /home/samba/profiles
; guest ok = no
; browseable = no
; create mask = 0600
; directory mask = 0700

[printers]
comment = All Printers
browseable = no
path = /var/spool/samba
printable = yes
guest ok = no
read only = yes
create mask = 0700

# Windows clients look for this share name as a source of downloadable
# printer drivers
[print$]
comment = Printer Drivers
path = /var/lib/samba/printers
browseable = yes
read only = yes
guest ok = no
# Uncomment to allow remote administration of Windows print drivers.
# You may need to replace 'lpadmin' with the name of the group your
# admin users are members of.
# Please note that you also need to set appropriate Unix permissions
# to the drivers directory for these users to have write rights in it
; write list = root, @lpadmin

[PublicShare]
comment = Public Share
path = /etc/sharedFolder
browseable = yes
read only = no
guest ok = yes
create mask = 0755
```

Figura 64 - Edição do ficheiro de configuração Samba - `/etc/samba/smb.conf`

As permissões e parâmetros relevantes para a configuração do Samba que pretendemos alcançar são os seguintes:

Comando	Descrição
<code>[PublicShare]</code>	Define o nome do compartilhamento Samba.
<code>comment = Public Share</code>	Fornece um comentário descritivo para o compartilhamento.
<code>path = /etc/sharedFolder</code>	Especifica o caminho do sistema de ficheiros para o diretório que será partilhado.
<code>browseable = yes</code>	Determina que o compartilhamento será visível ao navegar pela rede.
<code>read only = no</code>	Indica se o compartilhamento não é somente de leitura, ou seja, se os utilizadores podem criar, modificar e excluir ficheiros no compartilhamento.
<code>guest ok = yes</code>	Permite que os utilizadores acedam ao compartilhamento como convidados, sem necessidade de autenticação.
<code>create mask = 0755</code>	Define as permissões padrão para novos ficheiros criados no compartilhamento.

Tabela 1 - Tabela de definições dos comandos utilizados

Finalmente, no sistema Windows, mapeámos a pasta partilhada como uma unidade de rede através do Explorador de Ficheiros. Isto irá permitir aceder a uma pasta compartilhada noutro computador ou servidor como se fosse uma unidade local (como C:, D:, etc.).

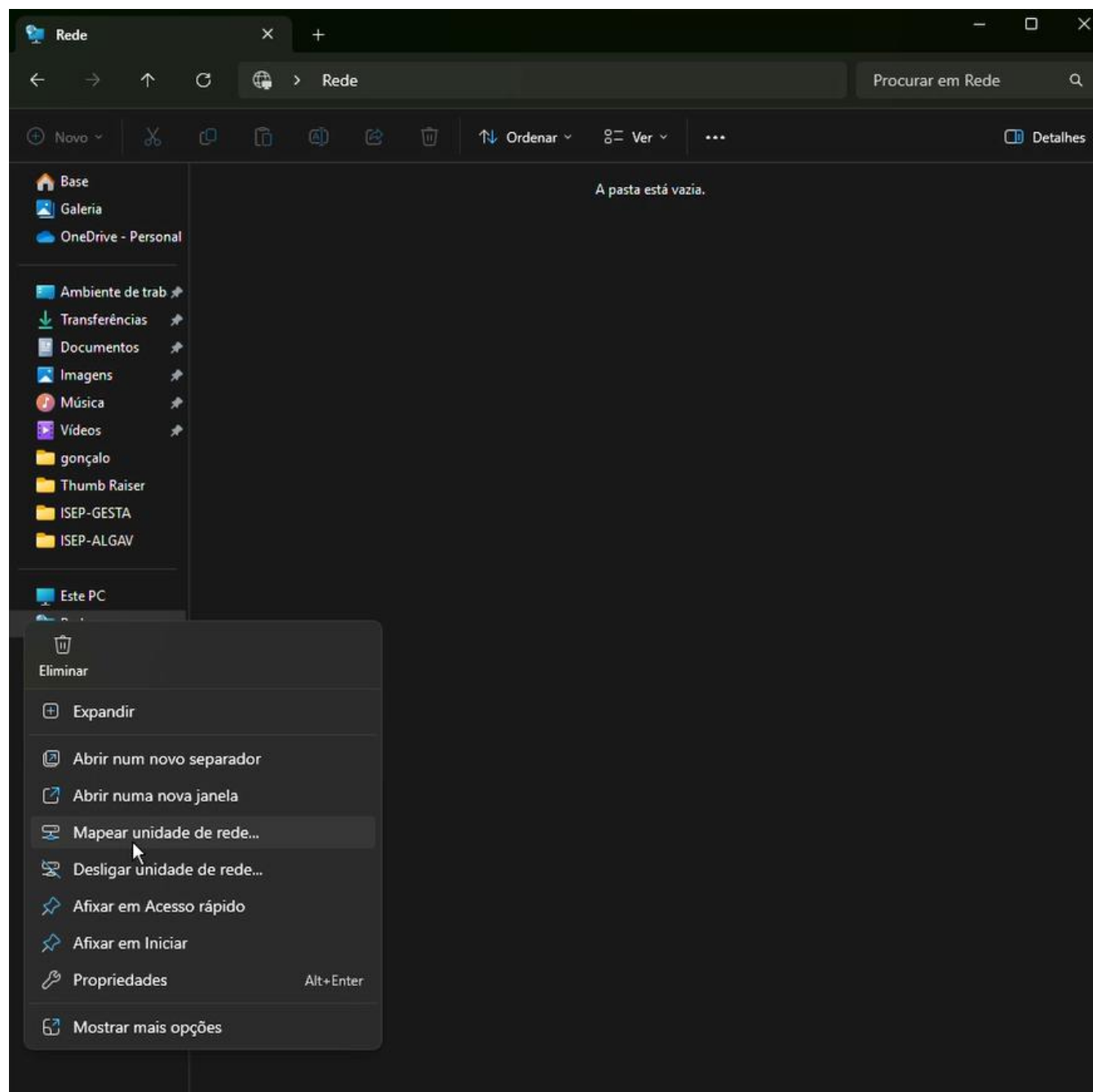


Figura 65 - Mapeamento da pasta partilhada no Windows

Durante o processo de mapeamento, definimos a unidade de rede e o caminho da pasta compartilhada na rede. O endereço IP 10.9.21.149 indica o computador ou servidor onde a pasta está localizada, e PublicShare é o nome da pasta compartilhada.

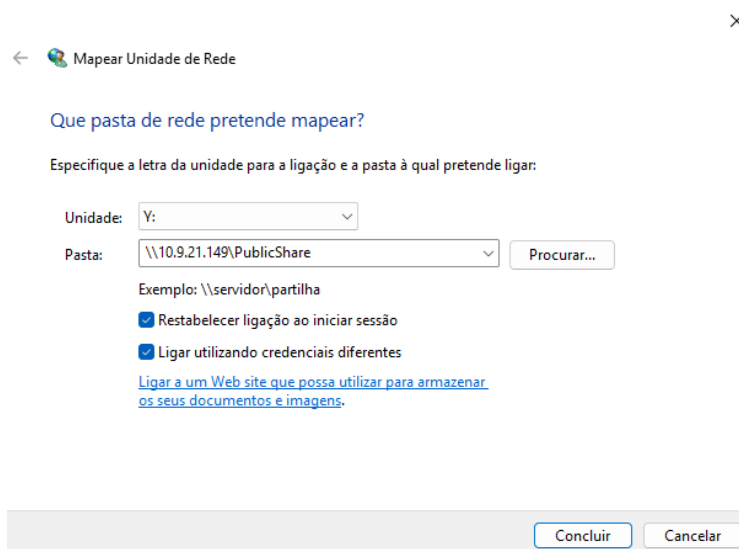


Figura 66 - Definição da unidade de rede e do caminho da pasta compartilhada na rede

Após este passo, a partilha ficou acessível como se fosse um disco local, permitindo a colaboração eficiente entre as equipas. Agora basta ao utilizador se autenticar com as credencias definidas anteriormente no servidor Linux.

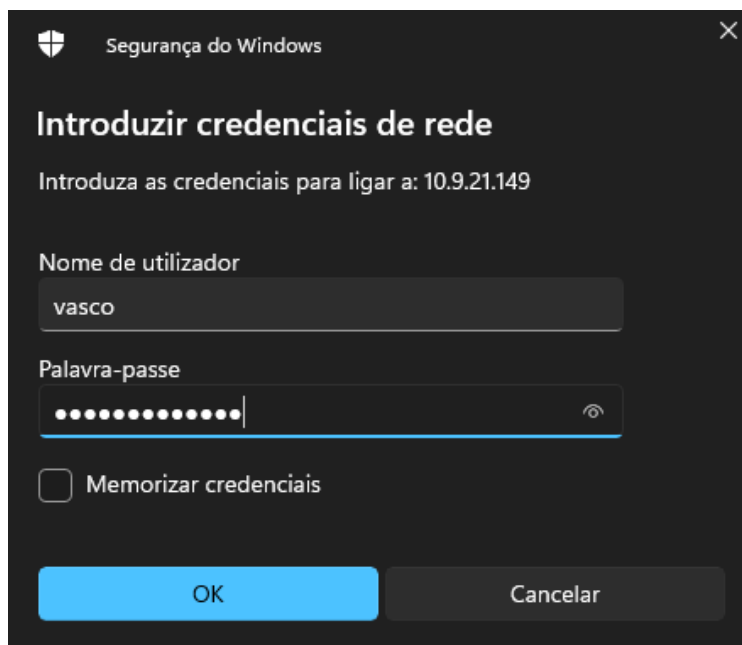


Figura 67 - Teste de autenticação do utilizador vasco

Em caso de autenticação bem-sucedida o sistema Windows redireciona-nos diretamente para a pasta partilhada.

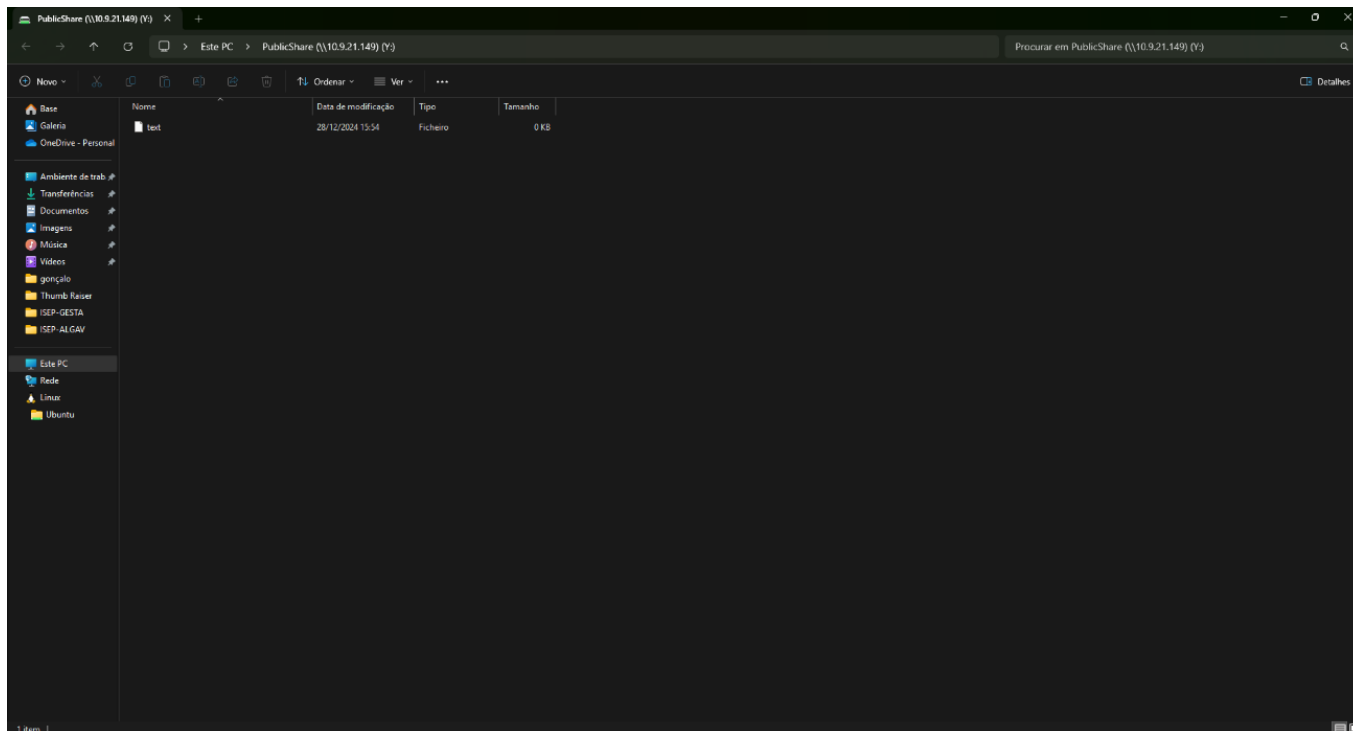


Figura 68 - Caso de autenticação bem-sucedida

User Story 12

Pedido: Como administrador de sistemas temos de garantir que em caso de necessidade os backups foram efetuados corretamente. Para isso devemos automatizar a sua reposição, validando no final o funcionamento do sistema (Ex. Base de Dados - executar uma query SQL com sucesso após reposição).

Aluno Responsável: Vasco Sousa (1221700)

Solução: De forma a solucionar esta User Story, começamos pela instalação do MongoDB e das ferramentas associadas. Utilizamos o comando ***apt-get install -y mongodb-org*** para instalar o servidor MongoDB, e adicionalmente instalamos o cliente MongoDB Shell e ferramentas extra com ***apt-get install -y mongodb-mongosh mongodb-database-tools-extra***.

Após a instalação, verificamos se todos os pacotes necessários foram instalados corretamente através do comando ***dpkg -l | grep mongo***. A lista apresentada confirmou a instalação dos seguintes pacotes:

- ***mongodb-org*** (base do sistema de base de dados orientada a documentos)
- ***mongodb-mongosh*** (interface de linha de comandos para interação com o MongoDB)
- ***mongodb-database-tools-extra*** (ferramentas adicionais para gestão do MongoDB)
- Entre outros componentes necessários, como ***mongodb-org-server*** e ***mongodb-org-tools***.

De seguida, criamos um script chamado ***verifyBackups.sh*** utilizando o editor de texto nano. Este script foi configurado para validar automaticamente os backups efetuados, garantindo que o sistema se encontra funcional após uma reposição.

```
#!/bin/bash

# MongoDB connection details
MONGO_HOST="vassao-dbi-lin-10p-pt"
MONGO_PORT="27017"
MONGO_USER="mongodb"
MONGO_PASS="c6276832afe34f39c8a84"
AUTH_DB="admin"
DB_NAME="test"
BACKUP_DIR="/root/backups"

# Set the current date for filename and log file
CURRENT_DATE=$(date +%Y%m%d)
FILENAME="mongo_${CURRENT_DATE}.tar.gz"
COLLECTION_NAME="allergies"
QUERY="{_id:'${MONGO_HOST}-${MONGO_PORT}-${MONGO_USER}-${MONGO_PASS}'}"

# Log file where script activities will be logged
LOG_FILE="/etc/restore/verifyLog_${CURRENT_DATE}.log"

# Print and log the current date
echo "Current date: ${CURRENT_DATE}" | tee -a $LOG_FILE

# Clean the dump directory if it is not empty
if [ -d "$BACKUP_DIR/dump" ]; then
  echo "Cleaning up the existing dump directory..." | tee -a $LOG_FILE
  rm -rf "$BACKUP_DIR/dump/*" | tee -a $LOG_FILE
fi

# Create the backup directory if it doesn't exist and navigate to it
mkdir -p "$BACKUP_DIR/dump" && cd "$BACKUP_DIR"

# Extract the backup file and log the output
echo "Extracting backup file: $FILENAME" | tee -a $LOG_FILE
tar -xzf $FILENAME -C dump | tee -a $LOG_FILE

# Ensure that the backup files are in the correct structure
# Move files into the 'dump/admin' directory for proper restoration
if [ -d "$BACKUP_DIR/dump/admin" ]; then
  echo "Moving files from 'dump/admin' to 'dump/dump' directory..." | tee -a $LOG_FILE
  mv "$BACKUP_DIR/dump/admin/*" "$BACKUP_DIR/dump/dump/" && echo "Move complete" | tee -a $LOG_FILE
else
  echo "Creating 'dump/admin' directory if it doesn't exist" | tee -a $LOG_FILE
  mkdir "$BACKUP_DIR/dump/admin" && echo "Directory created" | tee -a $LOG_FILE
fi

# Drop the current database before restoring from backup and log the output
echo "Dropping the existing database $DB_NAME" | tee -a $LOG_FILE
mongosh --host $MONGO_HOST --port $MONGO_PORT --username $MONGO_USER --password $MONGO_PASS --authenticationDatabase $AUTH_DB --eval "use($DB_NAME)" | tee -a $LOG_FILE

# Restore the MongoDB dump from the backup, excluding system collections and logging the output
echo "Restoring database from backup..." | tee -a $LOG_FILE
mongorestore --host $MONGO_HOST --port $MONGO_PORT --username $MONGO_USER --password $MONGO_PASS --authenticationDatabase $AUTH_DB --ifFrom=admin --ifNot=test --drop --noIndexRestore --dir "$BACKUP_DIR/dump/dump" --exclude="admin.system.version" --exclude="admin.system.users" | tee -a $LOG_FILE

# Drop the current database before restoring from backup and log the output
echo "Dropping the existing database $DB_NAME" | tee -a $LOG_FILE
mongosh --host $MONGO_HOST --port $MONGO_PORT --username $MONGO_USER --password $MONGO_PASS --authenticationDatabase $AUTH_DB --eval "use($DB_NAME)" | tee -a $LOG_FILE

# Query the 'allergies' collection to check all documents and inspect their structure
echo "Querying all documents in the 'allergies' collection" | tee -a $LOG_FILE
mongosh --host $MONGO_HOST --port $MONGO_PORT --username $MONGO_USER --password $MONGO_PASS --authenticationDatabase $AUTH_DB --eval "use($DB_NAME).find($QUERY)" | tee -a $LOG_FILE

# Clean up the backup directory
echo "Cleaning up the dump directory..." | tee -a $LOG_FILE
rm -rf dump/ | tee -a $LOG_FILE
```

Figura 69 - Criação do script ***verifyBackups.sh***

Os principais comandos definidos neste script são os seguintes:

Comando	Descrição
<i>if [-d "\$BACKUP_DIR/dump"; then ... fi</i>	Verifica se a pasta dump existe. Se existir, elimina todos os ficheiros no seu interior para limpar qualquer registo existente antes de validar a integridade dos backups.
<i>mkdir -p "\$BACKUP_DIR/dump"</i>	Cria a pasta dump caso esta não exista.
<i>cd "\$BACKUP_DIR"</i>	Altera o diretório de trabalho para a pasta onde se encontram os backups.
<i>tar -zxvf \$FILENAME -C dump</i>	Extraí o conteúdo do ficheiro de backup (tar.gz) para a pasta dump.
<i>if [-d "\$BACKUP_DIR/dump/admin/admin"]; then ... fi</i>	Move os ficheiros da subpasta admin/admin para dump/admin de forma a corrigir a estrutura de diretórios necessária para restaurar corretamente no MongoDB.
<i>mongosh --eval "db.dropDatabase()"</i>	Elimina a base de dados atual (test) utilizando o MongoDB Shell (mongosh), preparando-a para o restauro.
<i>mongorestore ...</i>	Restaura a base de dados a partir da pasta dump convertendo coleções admin.* para test.* e eliminando dados antigos.
<i>mongosh --eval "db.stats()"</i>	Mostra estatísticas sobre a base de dados restaurada, como o número de coleções, tamanho de armazenamento, entre outros.
<i>mongosh --eval "db.\$COLLECTION_NAME.find(\$QUERY)"</i>	Consulta a coleção allergies utilizando o filtro especificado (QUERY) para verificar os dados restaurados.
<i>rm -rf dump/</i>	Elimina a pasta dump após o processo de restauro de maneira a limpar ficheiros temporários.

Tabela 2 - Definição dos comandos utilizados no script

Após a criação do script foi necessário definir as permissões necessárias de forma que o ficheiro fosse executável. Para isto utilizamos o comando ***chmod +x verifyBackups.sh***.

De maneira a automatizar a execução deste script, editamos o cron jobs utilizando o comando **crontab -e**. Sendo assim, adicionamos a linha **0 3 * * * /root/verifyBackups.sh > /dev/null 2>&1**, que garante que o script irá ser executado diariamente às 3 da manhã. A configuração **> /dev/null 2>&1** assegura que todo o output (**stdout**) e erros (**stderr**) do script são descartados, evitando o aparecimento dos mesmos durante a execução do script no horário definido e assegurando assim que qualquer outra tarefa pode ser realizada durante o tempo de execução.

```
GNU nano 5.4 /tmp/crontab.VSgHCw/crontab
# 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
* * * * * /usr/local/bin/generate-ssh-banner.sh
0 2 * * * /bin/bash /root/backupDBUS3.sh
1 2 * * * /bin/bash /root/remove_old_backups.sh
0 3 * * * /root/verifyBackups.sh > /dev/null 2>&1
|
```

Figura 70 - Adição de uma entrada na cron table para a automatização do script

Com esta configuração, conseguimos garantir que o sistema executa os testes de validação automaticamente, detetando eventuais problemas de forma antecipada e reduzindo a intervenção manual necessária.

```
root@u495:~# ./verifyBackups.sh
2024-12-20
Extracting backup file: admin_20241220.tar.gz
admin/
admin/admin/
admin/admin/allergies.metadata.json
admin/admin/system.users.bson
admin/admin/medicalconditions.bson
admin/admin/system.version.metadata.json
admin/admin/system.version.bson
admin/admin/patientmedicalhistories.bson
admin/admin/patientmedicalhistories.metadata.json
admin/admin/appointments.bson
admin/admin/allergies.bson
admin/admin/patientmedicalhistories.metadata.json
admin/admin/medicalconditions.metadata.json
admin/admin/appointments.metadata.json
admin/admin/system.users.metadata.json
admin/admin/specializations.metadata.json
admin/admin/specializations.bson
Moving files from 'admin/admin' to 'dump/admin' directory...
Dropping the existing database test
[ ok ] Dropping 'test'
Restoring database from backup...
2024-12-30T20:48:51.587+0000 reading collections to restore from
2024-12-30T20:48:51.588+0000 reading metadata for test.allergies from dump/admin/allergies.metadata.json
2024-12-30T20:48:51.588+0000 reading metadata for test.appointments from dump/admin/appointments.metadata.json
2024-12-30T20:48:51.588+0000 reading metadata for test.medicalconditions from dump/admin/medicalconditions.metadata.json
2024-12-30T20:48:51.588+0000 reading metadata for test.patientmedicalhistories from dump/admin/patientmedicalhistories.metadata.json
2024-12-30T20:48:51.587+0000 reading metadata for test.specializations from dump/admin/specializations.metadata.json
2024-12-30T20:48:51.679+0000 restoring test.medicalconditions from dump/admin/medicalconditions.bson
2024-12-30T20:48:51.679+0000 restoring test.patientmedicalhistories from dump/admin/patientmedicalhistories.bson
2024-12-30T20:48:51.693+0000 finished restoring test.patientmedicalhistories (28 documents, 0 failures)
2024-12-30T20:48:51.694+0000 restoring test.medicalconditions (13 documents, 0 failures)
2024-12-30T20:48:51.763+0000 restoring test.allergies from dump/admin/allergies.bson
2024-12-30T20:48:51.767+0000 restoring test.specializations from dump/admin/specializations.bson
2024-12-30T20:48:51.771+0000 restoring test.appointments from dump/admin/appointments.bson
2024-12-30T20:48:51.784+0000 finished restoring test.specializations (6 documents, 0 failures)
2024-12-30T20:48:51.784+0000 finished restoring test.allergies (3 documents, 0 failures)
2024-12-30T20:48:51.794+0000 finished restoring test.appointments (1 document, 0 failures)
2024-12-30T20:48:51.794+0000 43 document(s) restored successfully. 0 document(s) failed to restore.
Showing database stats
{
  db: "test",
  collections: Long("5"),
  views: Long("0"),
  objects: Long("43"),
  avgObjSize: 24.3255013953468,
  dataSize: 14720,
  storageSize: 20480,
  indexes: Long("5"),
  indexSize: 20480,
  totalSize: 40960,
  scaleFactor: Long("1"),
  fileSize: 792853406464,
  fsTotalSize: 792853406464,
  ok: 1
}
Querying all documents in the 'allergies' collection
{
  _id: ObjectId("67985bba1257135b714c685"),
  severity: "Severe",
  allergen: "Peanut",
  allergyCode: "1778.40",
  allergyName: "Allergy to food: unspecified",
  allergyDescription: "This allergy refers to an abnormal immune response triggered by the consumption of specific foods. It may involve mild to severe symptoms, including life-threatening anaphylaxis in extreme cases. Common allergenic foods include peanuts, shellfish, dairy, and eggs.",
  allergyNotes: "Skin reactions such as hives, rash, or itching. Swelling of the face, lips, tongue, or throat. Difficulty breathing or wheezing. Gastrointestinal distress such as nausea, vomiting, or diarrhea. Dizziness, lightheadedness, or fainting.",
  createdAt: ISODate("2024-12-30T12:44:18.174Z"),
  updatedAt: ISODate("2024-12-30T20:48:52.892Z"),
}
Cleaning up the dump directory...
```

Figura 71 - Teste de recuperação de um ficheiro backup e execução de uma query para verificar a integridade do ficheiro