

Relatório de ASIST

Sprint 3

Turma 3DI - Grupo 50

1191296 - Gabriel Gonçalves

1191369 - Tiago Leite

1201305 - Tiago Afonso

1211304 - Francisco Bogalho

Data: 02/01/2024

Índice

Índice de quadros, figuras, abreviaturas	3
Distribuição de tarefas	4
User Story 1.....	5
User Story 2.....	9
User Story 3.....	11
User Story 4.....	14
User Story 5.....	15
User Story 6.....	20
User Story 7.....	21
User Story 8.....	24
User Story 9.....	26
User Story 10.....	31
User Story 11.....	33
User Story 12.....	37

Índice de quadros, figuras, abreviaturas

Figura 1 - Script para a realização do backup diário da base de dados de produção.....	12
Figura 2 - Ficheiro /etc/crontab com a configuração para executar diariamente a script de cópia de segurança da base de dados.....	13
Figura 3 – save_database_backup.sh	14
Figura 4 – crontab	14
Figura 5 - Configurações de log default para as facilities usadas frequentemente.....	15
Figura 6 - Configurações personalizadas de eventos.....	16
Figura 7 - Alterações na script /etc/daily_database_backup.sh.....	17
Figura 8 - Alterações na script de login /etc/profile	18
Figura 9 - Script para verificar se existem novos logs de severity mínima “emerg”.....	18
Figura 10 - Avisos ao administrador sobre falhas no sistema.....	19
Figura 11 remove_old_backups.sh	20
Figura 12 /etc/crontab.....	20
Figura 13 - Possível configuração do HAProxy.....	27
Figura 14 - Configuração do HAProxy	27
Figura 15 - Validação da configuração do HAProxy	28
Figura 16 - Aplicação das alterações do HAProxy	29
Figura 17 - Monitorização do desempenho e estado dos servidores.....	29
Figura 18 - Realização de teste de balanceamento de carga.....	29
Figura 19 - Realização de teste de failover	30
Figura 20 - Configuração da partilha de ficheiros no formato SMB/CIFS	33
Figura 21 - Efeito da opção “browsable = yes”	34
Figura 22 - Criação de um ficheiro com um pequeno texto de boas-vindas na partilha “RobDroneGo” ...	34
Figura 23 - Acesso à pasta da partilha através de um cliente Windows.....	35
Figura 24 - Conteúdo do ficheiro da partilha	35
Figura 25 - Negação de alteração nos ficheiros da partilha criada.....	36
Figura 26 - Acesso negado à tentativa de criação de novo ficheiro na partilha	36
Figura 27 - restore_and_validate_backup.sh	37
Figura 28 - Exemplo de execução de restore_and_validate_backup.sh.....	38

Distribuição de tarefas

As *User Stories* foram distribuídas pelos membros do grupo da seguinte forma:

User Story	Membro
1	Francisco Bogalho (1211304)
2	Gabriel Gonçalves (1191296)
3	Tiago Leite (1191369)
4	Tiago Afonso (1201305)
5	Tiago Leite (1191369)
6	Tiago Afonso (1201305)
7	Francisco Bogalho (1211304)
8	Gabriel Gonçalves (1191296)
9	Gabriel Gonçalves (1191296)
10	Francisco Bogalho (1211304)
11	Tiago Leite (1191369)
12	Tiago Afonso (1201305)

User Story 1

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

O plano de recuperação de desastres é um documento formal que auxilia a empresa a se preparar e responder de maneira eficaz e eficiente caso ocorram falhas e interrupções inesperadas.

De maneira a satisfazer o MBCO definido no Sprint B e tendo em conta os riscos definidos no Sprint B e o Business Impact Analysis elaborado neste Sprint elaboramos o plano de recuperação de desastres enunciado.

Propósito

As organizações não podem evitar desastres, porém podem desenvolver ações para minimizar seus efeitos. O objetivo de um DRP (Plano de Recuperação de Desastre) é reduzir o downtime e a perda de dados da organização no caso de desastres criados por fatores fora do controlo da empresa (incêndio, inundação, falha de energia, sabotagem, avaria, ...).

Objetivos

- Reduzir ao máximo o tempo de paragem e perdas de dados em caso de desastre;
- Estabelecer planos de emergência;
- Otimizar a reposição dos serviços afetados;
- Definir cenários de desastre e os processos de recuperação.

Pré-planeamento e comitê executivo de DRP

Primeiramente, foi criado um comitê executivo que tem como objetivo planejar e atualizar o DRP e orientar e fornecer apoio às equipas de projetos em assuntos referentes ao DRP. Os gerentes do projeto devem trabalhar com o comitê para finalizar o planeamento detalhado e desenvolver entrevistas para avaliar a segurança e elaborar a análise de impacto no negócio.

Existe um programa para a educação da gerência e das pessoas chave do projeto em relação ao DRP e aos procedimentos nele referidos.

Contactos

Tabela 1 - Contactos e funções dos elementos integradores do comité executivo DRP

Nome	Função	Email	Telemóvel
Francisco Bogalho	Administrador	1211304@isep.ipp.pt	111 111 111
Tiago Afonso	Técnico do sistema de monitorização	1201305 @isep.ipp.pt	222 222 222
Tiago Leite	Chefe do comité executivo de DRP	1191369@isep.ipp.pt	333 333 333
Gabriel Gonçalves	Chefe da equipa de Resposta a Incidentes	1191296@isep.ipp.pt	444 444 444

Armazenamento da documentação do plano

Cópias do plano, em formato físico e digital, serão armazenadas em múltiplas localizações seguras no recinto da empresa, definidas pela mesma. Todos os membros da equipa terão acesso a uma cópia digital do DRP. Os membros da equipa de recuperação após desastre têm acesso a uma cópia digital, assim como a uma cópia física.

Estratégia de criação de backups

O sistema como um todo, é a junção entre vários componentes, e a relação entre si. Desta forma, é crítico que existam backups para todos os componentes de sistema. Para qualquer um dos componentes foi definido que existirão 3 backups:

- um backup na cloud, salvaguardando de danos físicos, como furtos ou catástrofes,
- um backup num servidor remoto,
- um backup numa cópia num disco externo, guardada num local seguro, salvaguardando de possíveis falhas de rede.

Esta estratégia segue o princípio “3,2,1”, que diz devem existir 3 cópias, em 2 formatos diferentes, sendo que 1 cópia deve ficar armazenada num local diferente.

Ativadores do plano

O DRP é automaticamente ativado aquando da ocorrência de uma ou mais destas situações:

- Incapacitação da internet;

- Perda total ou parcial da frota de robots;
- Inundação do edifício;
- Avaria na base de dados;
- Perda de informação na base de dados;
- Avaria nos servidores;
- Falha de energia.

Plano de recuperação

Em todos os casos, a seguir apresentados, a equipa será dividida em duas subequipas que trabalharão por turnos de 4 horas cada até que o problema seja resolvido.

Sendo o desastre natural o adequado é chamar as autoridades competentes para controlar o sucedido e permitir á equipa de respostas a incidentes operar nas instalações para diagnosticar e retificar qual quer que seja o problema.

Se o desastre for oriundo do espectro de defeitos do hardware o procedimento é diagnosticar, seguido de arranjar se possível, se não substituir a infraestrutura, podendo ser necessário cuidados posteriores, como no caso das bases de dados e da frota de robots:

- Para as bases de dados é necessária reposição dos dados da base de dados pelo backup mais recente dos disponíveis;
- Para as frotas é necessária a reposição dos robots em falta e a verificação da funcionalidade dos robots quando ligados ao sistema.

Se o desastre for oriundo de um bug ou corrompimento de software o procedimento é chamar a equipa adequada para diagnosticar o problema, de seguida corrigi-lo ou reverter a versão do software para a última versão estável.

Formulário pós desastre

Após o final do período de desastre, deve ser preparado um relatório acerca das atividades exercitadas durante a execução do DRP.

O relatório deve conter:

- Informação sobre a emergência, quem foi notificado e quando, e a ação tomada pelos responsáveis;
- As pessoas notificadas da emergência;

- As ações tomadas pelos membros da equipa de recuperação de um desastre;
- Resultados diretos dessas ações;
- Aprendizagens com os eventos.

Atualização do DRP

Para atualizar o DRP, é necessário definir bem as mudanças propostas, com o intuito de corrigir eventuais lacunas no plano em vigor.

O DRP deve estar em constante melhoria, sendo necessário a sua atualização após cada incidente tendo em conta o formulário pós desastre mais recente, assim como constantemente testado, de modo a apresentar-se o mais robusto possível e, desta forma, combater de forma mais eficaz as ameaças.

User Story 2

Esta *US* visa apresentar de forma justificada as várias alterações a realizar na infraestrutura de forma a assegurar um *MTD* (*Maximum Tolerable Downtime*) de 20 minutos.

Um *MTD* (*Maximum Tolerable Downtime*) é o tempo de inatividade máxima tolerável de um sistema. Para assegurar isso, é crucial avaliar e reconhecer os sistemas e serviços vitais para o funcionamento da organização. Em seguida, aplicar medidas de alta disponibilidade e resistência a falhas a esses serviços e sistemas torna-se essencial. Para assegurar um *MTD* de no máximo 20 minutos, devem ser adotadas medidas como:

- **Estabelecer sistemas de monitorização e alerta** – Para prever problemas no sistema é crucial assegurar a presença de um sistema de monitorização capaz de detetar falhas ou problemas de maneira ágil e enviar alertas aos responsáveis. Isso possibilita a resolução eficiente de questões e a manutenção da operacionalidade do sistema;
- **Instaurar sistemas de cópia de segurança e recuperação** – Um plano de *backup* eficaz é essencial para assegurar a integridade dos dados e dos sistemas críticos, como a base de dados. A realização frequente de *backups* é fundamental para assegurar a continuidade do sistema, possibilitando uma recuperação rápida em caso de falha;
- **Recorrer a sistemas de *clustering* ou balanceamento de carga** – A implementação de mecanismos que distribuam a carga entre vários servidores na aplicação é crucial para assegurar a continuidade do serviço, mesmo perante falhas em um ou mais módulos;
- **Assegurar a formação dos funcionários** – Capacitar os funcionários para lidar com problemas ou falhas do sistema, incluindo a identificação e comunicação de *bugs*/problemas, bem como a execução do plano de *backup*;
- **Executar testes de carga e stress** – A verificação da capacidade dos sistemas e serviços críticos para suportar níveis elevados de tráfego e carga é imperativo. A partir dos resultados, tomar medidas corretivas para resolver eventuais problemas identificados, garantindo assim a robustez da infraestrutura;
- **Realizar simulações de recuperação de desastres** – A realização de simulações regulares testam a eficácia do plano de recuperação de desastres, familiarizando a equipa com os procedimentos;

- **Conduzir auditorias de segurança regularmente** – A execução de auditorias de segurança periódicas identificam potenciais vulnerabilidades e garantem a conformidade com as melhores práticas de segurança;
- **Atualizar proativamente *software* e *hardware*** – A implementação de um processo contínuo de atualização de *software* e *hardware* permite manter a infraestrutura resiliente contra ameaças e compatível com as tecnologias mais recentes;
- **Monitorizar o desempenho continuamente** – Além da monitorização de falhas, incluir a monitorização de desempenho para identificar tendências e otimizar a capacidade dos sistemas antes de atingirem limites críticos.

Em suma, é imperativo estar previamente preparado para enfrentar potenciais desafios no sistema, assegurando uma resposta eficaz para restaurar a normalidade rapidamente. Essa prontidão visa minimizar ao máximo os transtornos aos utilizadores, promovendo uma experiência contínua e eficiente.

User Story 3

Nesta *User Story* (US), é pedido que “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*”.

A nossa base de dados de produção atualmente encontra-se implementada nos servidores virtuais do DEI, pelo que o servidor utilizado para realizar esta US é também um servidor virtual do DEI. Apesar de estar feito desta forma, como estamos a desenvolver um projeto académico, não existe muito problema nisto. Mas se isto fosse um projeto a aplicar no “mundo real”, esta cópia de segurança da base de dados deveria ser feita num servidor fisicamente distante do servidor original. Isto deveria ser dessa forma pois, um bom administrador de sistemas tem de pensar em todos os casos possíveis que podem acontecer. Se alguma coisa inesperada acontece na zona que tem o servidor da base de dados, e que por sua vez, neste caso, também tem o servidor que possuiu e faz as cópias de segurança, será tudo perdido na mesma (por exemplo, alguém entra na sala que tem os servidores e destrói tudo). Desta forma, a cópia de segurança apenas está limitada a abranger um conjunto de problemas que possam acontecer, mas não 100% deles, ou lá perto. Temos de ter sempre em mente que a cópia de segurança da nossa base de dados deve estar, e ser feita, num servidor fisicamente afastado do servidor que contém a base de dados em produção.

Dado o enquadramento, vamos agora demonstrar como esta US foi desenvolvida. No *sprint* passado, já estava a ser feito uma cópia de segurança da nossa base de dados, só que esta não atende os requisitos desta US. Atualmente, temos uma cópia de segurança incremental durante os dias da semana e total ao domingo. Agora, iremos realizar uma cópia de segurança total diária e com o padrão requerido pela própria US. Foi definido que o formato dos *backups* deve ser <nome_da_db>_yyyymmdd, sendo que <nome_da_db> é 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.

Na implementação da US, começamos por criar a *script* necessária para ir de encontro ao que é pretendido. Criou-se então a *script* no caminho “/etc/daily_database_backup.sh” com o comando *touch*. Após isso, mudou-se as permissões para que apenas o utilizador que criou a *script* (neste caso, o utilizador *root*) a possa ler, escrever e executar, sendo que para isso utilizou-se o comando “*chmod 700 /etc/daily_database_backup.sh*”.

Depois escreveu-se o código necessário na *script*, sendo que esta ficou com o seguinte aspeto.

```

GNU nano 7.2 /etc/daily_database_backup.sh
#!/bin/bash

db_backup_dir=/etc/daily_database_backup
db_dump_dir=/etc/daily_database_dump
db_backup_logs_dir=/etc/daily_database_backup/logs

mkdir -p $db_backup_logs_dir

today_date=$(date +%Y%m%d)
db_log_file=$db_backup_logs_dir/RobDroneGo_${today_date}.log

mkdir -p $db_dump_dir
chmod -R 700 $db_dump_dir
mongodump --uri mongodb://mongoadmin:1f8402684f9645194a515c31@vs624.dei.isep.ipp.pt:27017/RobDroneGo?authSource=admin --out $db_dump_dir 2>&1 | tee $db_log_file

if [ $? -eq 0 ]; then
    mkdir -p $db_backup_dir
    chmod -R 700 $db_backup_dir
    tar -c -z -v -p -f $db_backup_dir/RobDroneGo_${today_date}.tgz $db_dump_dir/RobDroneGo >> $db_log_file 2>&1

    if [ $? -eq 0 ]; then
        echo "***Sucesso:** O backup foi realizado com sucesso." >> $db_log_file
        rm -r $db_dump_dir
    else
        echo "***Erro:** O backup falhou. Ocorreu um erro ao realizar o comando tar." >> $db_log_file
    fi
else
    echo "***Erro:** O backup falhou. Ocorreu um erro no mongodump." >> $db_log_file
fi

```

Figura 1 - Script para a realização do backup diário da base de dados de produção

A script começa por definir variáveis com os caminhos das pastas necessárias para a realização do *dump* da base de dados e do *backup*. Estes caminhos de pastas são criados ao longo da script caso ainda não existam, sendo que para isso é utilizado o comando “*mkdir -p /path/to/something*”. Ao criar estes caminhos de pastas, também são alteradas as permissões para que apenas o *owner* desta *script* consiga ler, escrever e executar coisas nessas pastas, sendo que para isso utiliza-se o comando “*chmod -R 700 /path/to/folder*”.

Para atender ao formato necessário para o nome dos *backups*, criou-se uma variável com esta informação, sendo que desta forma ela é reutilizada sempre que necessário.

Esta script faz ainda o *dump* da base de dados para uma localização de ficheiros específica, em que os *outputs* gerados por este comando são escritos no ficheiro de *logs* associado. Se o *dump* da base de dados tiver sucesso, é feita a cópia de segurança dos ficheiros gerados, senão é guardado nos *logs* um aviso sobre o sucedido e o programa termina.

Se a cópia de segurança feita com o comando *tar* for bem-sucedida, é colocada uma mensagem de sucesso no ficheiro de *logs*, senão é colocada uma mensagem de insucesso. Se uma cópia de segurança for realizada com sucesso, a pasta que contém o *dump* da base de dados é eliminada, pois deixa de ser necessária.

O comando *tar* utiliza alguns parâmetros, sendo esses:

“-c” – Cria um ficheiro novo;

“-z” – É para utilizar a compressão *gzip*;

“-v” – Ativa a saída detalhada (*verbose*) do comando, mostrando o progresso na consola enquanto o ficheiro é criado;

“-p” – Preserva as permissões dos ficheiros e pastas na pasta comprimida;

“-f” – Especifica o nome da pasta comprimida a ser criada, indicando também em que caminho irá ser colocada.

Após a criação da *script*, utilizou-se o comando “*bash -n /etc/daily_database_backup.sh*” para verificar se esta não tinha erros de sintaxe. Também se procedeu à sua execução, de forma a verificar que tudo estava a funcionar como era esperado.

Agora é necessário que a *script* seja executada diariamente, para que exista uma cópia de segurança diária. Neste caso, a cópia de segurança tem um *Recovery Point Objective (RPO)* de 24h, o que quer dizer que existe a possibilidade de haver perda de dados de no máximo de 24h. Esta perda de dados poderá acontecer entre as 2h30min da madrugada de cada dia, pois é quando é feito o *backup* à base de dados. Para que a *script* fosse então executada diariamente, editou-se o ficheiro “*/etc/crontab*” onde se acrescentou a última linha antes do #, como é possível visualizar na imagem seguinte.

```
GNU nano 7.2 /etc/crontab
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# Example of job definition:
# .----- minute (0 - 59)
# | .----- hour (0 - 23)
# | | .----- day of month (1 - 31)
# | | | .----- month (1 - 12) OR jan,feb,mar,apr ...
# | | | | .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# | | | | |
# * * * * * user-name command to be executed
17 * * * * root cd / && run-parts --report /etc/cron.hourly
25 6 * * * root test -x /usr/sbin/anacron || { cd / && run-parts --report /etc/cron.daily; }
47 6 * * 7 root test -x /usr/sbin/anacron || { cd / && run-parts --report /etc/cron.weekly; }
52 6 1 * * root test -x /usr/sbin/anacron || { cd / && run-parts --report /etc/cron.monthly; }
30 2 * * tue,wed,thu,fri,sat root cd /root/apps/RobDroneGo-MD && ./buildMDTask.sh > logs/buildMDTaskLog.txt
30 2 * * * root /etc/mongodb-backup.sh
30 2 * * * root /etc/daily_database_backup.sh
#
```

Figura 2 - Ficheiro */etc/crontab* com a configuração para executar diariamente a *script* de cópia de segurança da base de dados

Podemos então ver que a *script* criada, será executada diariamente às 2h30min da madrugada, sendo que a sua execução é desencadeada pelo utilizador *root*.

User Story 4

A User Story (US) pretende que os backups realizados diariamente, na US 3, sejam guardados de acordo com o seguinte calendário: 1 backup por mês no último ano, 1 backup por semana no último mês e 1 backup por dia na última semana. A US 12 elimina os backups diários com tempo de vida superior a 7 dias, garantindo assim "1 backup por dia na última semana". Para implementar "1 Backup por mês no último ano e 1 backup por semana no último mês", foi criado o script **save_database_backup.sh**, que move o backup realizado no dia atual para a pasta **/etc/saved_database_backups**. É também persistido num ficheiro log se o backup foi movido ou não com sucesso.

```
#!/bin/bash

source_directory="/etc/daily_database_backup"
destination_directory="/etc/saved_database_backups"

current_date=$(date +%Y%m%d)
file_format="RobDroneGo_${current_date}.tgz"
log_directory="${destination_directory}/logs"
mkdir -p "${log_directory}"
log_file="${log_directory}/move_file_log_${current_date}.txt"

if [ -e "${source_directory}/${file_format}" ]; then
    mv "${source_directory}/${file_format}" "${destination_directory}/"
    echo "$(date +%Y-%m-%d %H:%M:%S) - File moved successfully." >> "${log_file}"
else
    echo "$(date +%Y-%m-%d %H:%M:%S) - File not found: ${source_directory}/${file_format}" >> "${log_file}"
fi
```

Figura 3 – save_database_backup.sh

Este script é executado mensalmente no primeiro dia do mês e semanalmente ao domingo às 4:30 da manhã, garantindo assim que o calendário é cumprido.

```
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# Example of job definition:
# .----- minute (0 - 59)
# | .----- hour (0 - 23)
# | | .----- day of month (1 - 31)
# | | | .----- month (1 - 12) OR jan,feb,mar,apr ...
# | | | | .----- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# | | | | |
# * * * * * user-name command to be executed
17 * * * * root cd / && run-parts --report /etc/cron.hourly
25 6 * * * root test -x /usr/sbin/anacron || { cd / && run-parts --report /etc/cron.daily;>
47 6 * * 7 root test -x /usr/sbin/anacron || { cd / && run-parts --report /etc/cron.weekly;>
52 6 1 * * root test -x /usr/sbin/anacron || { cd / && run-parts --report /etc/cron.monthly;>
30 2 * * tue,wed,thu,fri,sat root cd /root/apps/RobDroneGo-MD && ./buildMDTask.sh > logs/buildMDTaskLog.txt
30 2 * * * root /etc/mongodb-backup.sh
30 2 * * * root /etc/daily_database_backup.sh
30 2 * * * root /etc/remove_old_backups.sh
# Backup mensal (1º dia)
30 2 1 * * root /etc/save_database_backup.sh
# Backup semanal (domingo)
30 2 * * 0 root /etc/save_database_backup.sh
```

Figura 4 – crontab

User Story 5

Nesta *User Story* (US), é pedido que “como administrador de sistemas quero que o processo da US da cópia de segurança da DB seja mantido no *log* do Linux, num contexto adequado, e alertado o administrador no acesso à consola se ocorrer uma falha grave neste processo”.

É muito importante que um sistema tenha sempre um registo de eventos (*logs*) a monitorizar a execução de aplicações, da realização de *backups*, entre outros. Isto facilita bastante a vida do administrador de sistemas (e não só) quando ocorrem falhas, pois será mais rápido de corrigir o problema, pois já se tem uma noção de onde pode vir o erro. Estes *logs* permitem que seja detetado algo que não funciona como pretendido, bem como detetar situações inesperadas que apesar de não comprometerem o funcionamento aparente dos sistemas, possam ser importantes de resolver. Vamos então demonstrar a implementação da monitorização do nosso *backup* à base de dados através da utilização de ficheiros de *log*.

Como ainda não existia um sistema de registo de eventos instalado na nossa máquina virtual, começou-se por fazer a instalação do sistema RSYSLOG, em que para isso utilizou-se o comando “*sudo apt install rsyslog*”.

Aqui, a classificação de eventos é feita através dos atributos *facility* e *severity*, sendo que à sua frente encontra-se a ação a ser realizada. No ficheiro “*/etc/rsyslog.conf*”, que é onde estão as configurações relativas ao sistema de registo de eventos agora instalado, por defeito, já existiam classificações de eventos. Uma delas era uma classificação de eventos para a *facility cron* e para todos os tipos de *severity* (por causa do carácter “*” que representa todos). Este padrão tem a ação de escrever os *logs* no ficheiro “*/var/log/cron.log*”. Desta forma, todos os *logs* de todos os níveis de *severity* da *facility cron*, irão ser escritos nesse ficheiro.

```
#
# Log commonly used facilities to their own log file
#
auth,authpriv.*      /var/log/auth.log
cron.*                -/var/log/cron.log
kern.*                -/var/log/kern.log
mail.*                -/var/log/mail.log
user.*                -/var/log/user.log
```

Figura 5 - Configurações de log default para as facilities usadas frequentemente

Aqui quando existe o caracter “-” atrás do caminho da ação, por exemplo “-/var/log/cron.log”, ele significa que os *logs* que deem *match* neste padrão, serão escritos no ficheiro de forma síncrona, ou seja, são imediatamente escritos no ficheiro. Caso não tivessem essa especificação, os *logs* poderiam não ser logo escritos no ficheiro de registo de eventos.

Para termos uma melhor perceção dos *logs* que representam uma falha, criou-se três ficheiros diferentes, sendo estes os ficheiros “/var/log/cron_info.log”, “/var/log/cron_emerg.log” e por fim “/var/log/special_cron_emerg.log”, em que para isso utilizou-se o comando *touch*. No primeiro ficheiro, serão guardados todos os *logs* que tenham uma *severity* maior ou igual a “info”, e no segundo apenas aqueles que tenham uma *severity mínima* de “emerg”. O terceiro é um ficheiro especial, que será utilizado para guardar os *logs* de nível mínimo “emerg” para quando o administrador fizer o *login* na consola, estes lhe sejam mostrados. Posteriormente, estes *logs* deste ficheiro são apagados para que na próxima vez que voltar a aceder à consola, o administrador não volte a ter o aviso de algo que provavelmente já corrigiu. Isto poderia ser feito de outra forma, em que o administrador apagava o conteúdo desse ficheiro manualmente após as falhas serem corrigidas, mas preferiu-se que fosse automático, pois os *logs* das falhas continuam de qualquer das formas no ficheiro “/var/log/cron_emerg.log”. Vale a pena sublinhar que para todos estes três ficheiros, a *facility* utilizada para os eventos é a *cron*.

Posteriormente, adicionou-se as seguintes linhas ao ficheiro “/etc/rsyslog.conf” para ir de em conta a estas configurações aqui abordadas.

```
#
# Log info severity from cron facility logs in specific file
#
cron.info                                /var/log/cron_info.log

#
# Log emerg severity from cron facility logs in specific file
#
cron.emerg                               -/var/log/cron_emerg.log

#
# Alert administrator of an emergency log when he is logged in the console
#
*.emerg                                  root

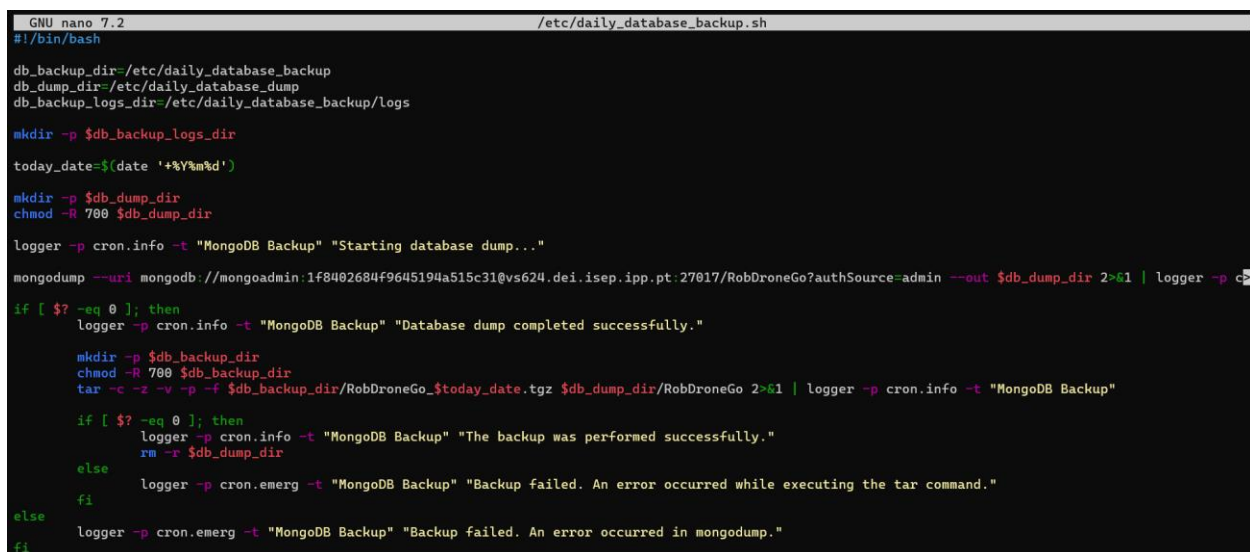
#
# Log emerg severity to special log file that will be used to warn administrator
# when he log into the console
#
cron.emerg                               -/var/log/special_cron_emerg.log
```

Figura 6 - Configurações personalizadas de eventos

Para além do falado anteriormente, foi definido uma configuração para que todos os eventos de qualquer *facility* mas que tivessem uma *severity* mínima de “*emerg*”, fossem notificadas ao utilizador *root* quando acontecem enquanto ele está conectado ao servidor pela consola.

Após a configuração deste ficheiro, foi necessário reiniciar o sistema *RSYSLOG*, em que para isso utilizou-se o comando “*systemctl restart rsyslog*”.

Para realizar o registo dos eventos do *backup* da base de dados, realizou-se a alteração da *script* “*/etc/daily_database_backup.sh*” que foi criada na *US 3*. Aqui, anteriormente eram feitos *logs* para ficheiros .txt, sendo que para esta *US* as únicas alterações necessárias a fazer, foram no local e na forma em como os *logs* eram registados. Para fazer o registo dos eventos nos ficheiros de *log* com a *facility* e *severity* apropriada a cada local da *script*, utilizou-se o comando *logger -p facility.severity -t “MongoDB Backup” “Appropriate message”*, em que o *-p* representa a prioridade e o *-t* a *tag* do *log*.



```
GNU nano 7.2 /etc/daily_database_backup.sh
#!/bin/bash

db_backup_dir=/etc/daily_database_backup
db_dump_dir=/etc/daily_database_dump
db_backup_logs_dir=/etc/daily_database_backup/logs

mkdir -p $db_backup_logs_dir

today_date=$(date '+%Y%m%d')

mkdir -p $db_dump_dir
chmod -R 700 $db_dump_dir

logger -p cron.info -t "MongoDB Backup" "Starting database dump..."

mongodump --uri mongodb://mongoadmin:1f8402684f9645194a515c31@vs624.dei.isep.ipp.pt:27017/RobDroneGo?authSource=admin --out $db_dump_dir 2>&1 | logger -p cron.info -t "MongoDB Backup"

if [ $? -eq 0 ]; then
    logger -p cron.info -t "MongoDB Backup" "Database dump completed successfully."

    mkdir -p $db_backup_dir
    chmod -R 700 $db_backup_dir
    tar -c -z -v -p -f $db_backup_dir/RobDroneGo_$today_date.tgz $db_dump_dir/RobDroneGo 2>&1 | logger -p cron.info -t "MongoDB Backup"

    if [ $? -eq 0 ]; then
        logger -p cron.info -t "MongoDB Backup" "The backup was performed successfully."
        rm -r $db_dump_dir
    else
        logger -p cron.emerg -t "MongoDB Backup" "Backup failed. An error occurred while executing the tar command."
    fi
else
    logger -p cron.emerg -t "MongoDB Backup" "Backup failed. An error occurred in mongodump."
fi
```

Figura 7 - Alterações na *script* */etc/daily_database_backup.sh*

Agora falta avisar o administrador quando ele faz o *login* na consola e aconteceu uma falha na sua ausência. Tal como dito anteriormente, foi para isso que se criou o ficheiro “*/var/log/special_cron_emerg.log*”, sendo que desta forma pode-se contar e listar as falhas e depois apagá-las do ficheiro. Para que isso fosse então possível, criou-se uma *script* para esse fim. Definiu-se a sua execução na *script* de *login* “*/etc/profile*”. Esta *script* criada apenas é chamada na parte em que o código executado é exclusivo para o administrador.

```
GNU nano 7.2 /etc/profile
# /etc/profile: system-wide .profile file for the Bourne shell (sh(1))
# and Bourne compatible shells (bash(1), ksh(1), ash(1), ...).

if [ "$(id -u)" -eq 0 ]; then
    PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"

    # Script para verificar se existem novos logs com severity emerg
    /etc/check_emerg_logs.sh

else
    PATH="/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games"
fi
export PATH

if [ "${PS1-}" != "" ]; then
```

Figura 8 - Alterações na script de login /etc/profile

```
GNU nano 7.2 /etc/check_emerg_logs.sh
#!/bin/bash

num_emerg_logs=$(wc -l /var/log/special_cron_emerg.log | awk '{print $1}')

if [ "$num_emerg_logs" -ne 0 ]; then
    # Sequência de escape ANSI para começar a formatação em vermelho
    RED='\033[0;31m'
    # Sequência de escape ANSI para resetar a formatação
    NC='\033[0m'

    echo -e "${RED}*****${NC}"
    echo -e "${RED}There are ${num_emerg_logs} emergency logs that you need to look at!${NC}"
    echo "The logs are: "
    cat /var/log/special_cron_emerg.log
    echo -e "${RED}*****${NC}"
    true > /var/log/special_cron_emerg.log # Limpar conteúdo do ficheiro
fi
```

Figura 9 - Script para verificar se existem novos logs de severity mínima “emerg”

Utilizou-se o comando *touch* para criar esta *script*, e depois alterou-se as suas permissões para que apenas o *owner* (neste caso o *root*) a possa ler, alterar e executar, sendo que para isso usou-se o comando “*chmod 700 /etc/check_emerg_logs.sh*”.

Agora sempre que o administrador faz *login* na consola, se existirem *logs* no ficheiro “*/var/log/special_cron_emerg.log*”, ele será avisado da quantidade de falhas existentes e também lhe será mostrado os *logs* que lá existem. Após isso, o conteúdo desse ficheiro é removido.

Para testar esta situação, provocou-se propositadamente eventos do tipo *cron.emerg* através do comando “*logger -p cron.emerg 'Error message'*”. Posteriormente, saiu-se da consola e voltou-se a conectar como administrador para verificar se tudo funcionava como esperado.

```

root@vs197:~# logger -p cron.emerg "Error message"
root: Error message

Broadcast message from systemd-journald@vs197 (Thu 2023-12-21 18:06:50 WET):

root[30335]: Error message

root@vs197:~# logger -p cron.emerg "Error message 1"
root: Error message 1

Broadcast message from systemd-journald@vs197 (Thu 2023-12-21 18:06:52 WET):

root[30336]: Error message 1

root@vs197:~# exit
logout
Connection to vsgate-ssh.dei.isep.ipp.pt closed.

C:\Users\ti_ma>ssh -p 10197 root@vsgate-ssh.dei.isep.ipp.pt
root@vsgate-ssh.dei.isep.ipp.pt's password:
Linux vs197 4.15.0-213-generic #224-Ubuntu SMP Mon Jun 19 13:30:12 UTC 2023 x86_64

Debian GNU/Linux 12

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: Thu Dec 21 17:49:33 2023 from 10.9.0.38
*****
There are 2 emergency logs that you need to look at!
The logs are:
2023-12-21T18:06:50.918000+00:00 vs197 root: Error message
2023-12-21T18:06:52.539805+00:00 vs197 root: Error message 1
*****
root@vs197:~#

```

Figura 10 - Avisos ao administrador sobre falhas no sistema

Como se pode observar, o administrador foi notificado das falhas enquanto estava conectado à consola, e também depois quando saiu da conta e voltou a fazer o login na consola. Se ele voltasse a sair e a entrar, estes *logs* já não iam voltar a aparecer. Também poderia haver um mecanismo de verificar se o administrador recebeu os *logs* enquanto estava na consola, e desta forma não os iria voltar a receber quando entrasse novamente na sua conta pela consola. No entanto, apenas é requerido a parte de o administrador receber esses avisos ao entrar na consola. E dificilmente ele irá estar *online* à hora do *backup*.

Também seria uma boa ideia implementar no futuro o *Logrotate*, visto que os ficheiros de *log* vão crescer sucessivamente todos os dias. Ao utilizar esta funcionalidade, seria possível, por exemplo, limpar os *logs* que já são antigos e já não são necessários manter, bem como organizar melhor os *logs* diários.

User Story 6

Esta US tem como objetivo garantir que as cópias de segurança realizadas na US4 tenham um tempo de vida não superior a 7 (sete) dias, ou seja, todas as cópias de segurança com mais de 7 dias devem ser eliminadas.

Para tal, foi criado o script em bash denominado `remove_old_backups.sh`:

```
#!/bin/bash

backup_folder="/etc/daily_database_backup"
days_threshold=7

threshold_date=$(date -d "$days_threshold days ago" +"%Y%m%d")

find "$backup_folder" -type f -name "RobDroneGo_*" -mtime +$days_threshold -exec rm -f {} \;
```

Figura 11 `remove_old_backups.sh`

Este usa o comando **find** para encontrar os ficheiros do tipo “RobDroneGo_*” (* é uma *wildcard*) modificados há mais de 7 dias (-mtime +\$days_threshold), sendo estes removidos (-exec rm -f {} \;)

Este script é executado diariamente às 2:30, e para isso, foi editado o arquivo `/etc/crontab`:

```
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab`
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# Example of job definition:
# .----- minute (0 - 59)
# | .----- hour (0 - 23)
# | | .----- day of month (1 - 31)
# | | | .----- month (1 - 12) OR jan,feb,mar,apr ...
# | | | | .----- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# * * * * *
# * * * * * user-name command to be executed
17 * * * * root cd / && run-parts --report /etc/cron.hourly
25 6 * * * root test -x /usr/sbin/anacron || { cd / && run-parts --report /etc/cron.daily; }
47 6 * * 7 root test -x /usr/sbin/anacron || { cd / && run-parts --report /etc/cron.weekly; }
52 6 1 * * root test -x /usr/sbin/anacron || { cd / && run-parts --report /etc/cron.monthly; }
30 2 * * tue,wed,thu,fri,sat root cd /root/apps/RobDroneGo-MD && ./buildMDTask.sh > logs/buildMDTaskLog.txt
30 2 * * * root /etc/mongodb-backup.sh
30 2 * * * root /etc/daily_database_backup.sh
30 2 * * * root /etc/remove_old_backups.sh
```

Figura 12 `/etc/crontab`

User Story 7

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) da US B4.

O objetivo do Business Impact Analysis (BIA) é determinar os impactos das falhas que possam eventualmente surgir e definir medidas para minimizar ou recuperar desses impactos. Ajuda a antecipar falhas, identificar pontos fracos no negócio e desenvolver planos de contingência de maneira a lidar com os problemas.

O BIA é composto por três passos:

1. Determinar processos do negócio. São identificados processos/missão do negócio apoiados pelo sistema e é determinado o impacto que uma rutura do sistema teria nesses processos, juntamente com o impacto da paralisação e o downtime estimado. O downtime deve refletir o máximo de tempo que a organização consegue tolerar, mantendo a missão.
2. Identificar os recursos necessários. É necessária uma avaliação aprofundada dos recursos necessários para retomar o mais rapidamente possível os processos de negócio. Exemplos de recursos que devem ser identificados incluem: instalações, pessoal, equipamentos, software, ficheiros de dados e componentes do sistema.
3. Identificar recursos do sistema com maior prioridade de recuperação, com base nos resultados anteriores. Haverá recursos do sistema e processos mais críticos para o negócio. Serão estabelecidos níveis prioritários para atividades e recursos a recuperar.

Análise de risco

Como foi apresentado no sprint passado já foram definidos os riscos da nossa aplicação, a sua probabilidade, as consequências e o seu risco.

Tabela 2 - Análise de risco

Ameaça	Probabilidade	Consequência	Risco
Blackout nos servidores	2	2	Média - 4
Avaria nos servidores	1	4	Média - 4
Perda da base de dados	1	4	Média - 4
Acesso indevido por falta de autenticação	3	4	Alta - 12

Exposição de Dados Sensíveis	1	3	Baixo - 3
Vulnerabilidades de Software	1	4	Médio - 4

Análise de Impacto

O RTO (Recovery Time Objective) refere-se ao período máximo que um serviço deve demorar até ser restaurado após uma interrupção ou falhas.

O RPO (Recovery Point Objective) é o período durante o qual a quantidade de dados que o sistema poderá vir a perder é tolerável para a empresa em caso de uma interrupção ou falhas.

O MTD (Maximum Tolerable Downtime) é o tempo de inatividade máxima tolerável de um sistema em caso de uma interrupção ou falhas.

Tabela 3 - Análise de impacto

Ameaça	RTO	RPO	MTD
Blackout nos servidores	8h	8h	16h
Avaria nos servidores	12h	12h	24h
Perda da base de dados	24h	24h	36h
Acesso indevido por falta de autenticação	16h	16h	32h
Exposição de Dados Sensíveis	16h	16h	32h
Vulnerabilidades de Software	8h	8h	16h

Recursos necessários

Na fase de identificação dos recursos essenciais para a recuperação dos processos de negócio, a nossa análise concentrou-se em elementos-chave. Começamos por delinear as necessidades em várias categorias.

- Definimos as instalações físicas necessárias, considerando alternativas em caso de falha nas principais.
- Em relação à equipa, listamos funções e capacidades essenciais, estabelecendo um plano de comunicação para notificação e mobilização eficaz.
- No âmbito de equipamento e software, identificamos os indispensáveis, garantindo a validade das licenças e o suporte necessário.
- Ao mesmo tempo, destacamos dados críticos e estabelecemos estratégias robustas de backup.
- Além disso, listamos componentes específicos, do sistema, essenciais para a operação contínua.

Prioridade de recuperação

Para estabelecer um negócio fidedigno e que satisfaça necessidades de negócio é necessário definir prioridades para a recuperação dos diferentes recursos do sistema:

1. Perda da base de dados;
2. Acesso indevido por falta de autenticação;
3. Avaria nos servidores;
4. Vulnerabilidades de software;
5. Exposição de dados sensíveis;
6. Blackout nos servidores.

As prioridades foram definidas tendo em conta regras de negócio e objetivos da empresa.

User Story 8

Na *User Story 8*, procura-se implementar uma gestão de acessos que cumpra os critérios apropriados de segurança. Para alcançar essa gestão segura, é essencial que o plano avalie determinados critérios como:

Autenticação forte

Este parâmetro concentra-se em assegurar que os utilizadores possuam credenciais robustas para o *login* na nossa aplicação *SPA*. Para isso foi implementado requisitos rigorosos para as palavras-passe, incluindo critérios como um mínimo de 10 caracteres, pelo menos 1 letra maiúscula, 1 letra minúscula, 1 dígito e 1 símbolo. Além disso, todas as palavras-passe são armazenadas de forma codificada na nossa base de dados, garantindo a segurança dos dados.

Controlo de Acesso

Este parâmetro desempenha um papel crucial na garantia da segurança e na gestão eficaz das operações na nossa aplicação *SPA*. Ao implementar o controlo de acesso, visamos não apenas proteger os dados sensíveis, mas também personalizar a experiência do utilizador com base nas suas responsabilidades na empresa.

Quando um utilizador realiza o *login* no sistema, é recebido uma resposta do *backend* que contém um *token* e a função do utilizador. Este *token* é armazenado localmente e utilizado como uma chave de autorização para aceder às funcionalidades do *backend*. Cada pedido enviado ao *backend* passa por um *middleware* que verifica a autenticidade do *token*, a associação a um utilizador válido e as permissões necessárias para executar a ação solicitada.

Além disso, o componente do menu é dinamicamente personalizado com base na função do utilizador. Ao aceder à aplicação, o utilizador é apresentado com um menu que reflete as funcionalidades disponíveis de acordo com a sua posição na empresa. Isso não só simplifica a navegação, mas também reforça a segurança, garantindo que os utilizadores só tenham acesso a funcionalidades pertinentes às suas responsabilidades.

A proteção das funcionalidades estende-se também às rotas. Cada rota é associada a funções específicas, e um serviço de verificação é implementado para garantir que o utilizador possui as funções necessárias para aceder à página correspondente. Este duplo controlo, tanto no nível de rota como no

nível de função, adiciona uma camada adicional de segurança, reduzindo o risco de acessos não autorizados.

Em resumo, o controlo de acesso não é apenas uma barreira de segurança, mas uma ferramenta estratégica para otimizar a experiência do utilizador e manter um ambiente de trabalho eficiente e seguro. Ao personalizar o acesso com base nas funções e responsabilidades, garantimos que cada utilizador tenha uma interação relevante e segura com a aplicação.

Políticas de Acesso

Este parâmetro concentra-se na definição de regras para a utilização dos recursos da organização. Para orientar todos os funcionários sobre as expectativas, estabelecemos as seguintes diretrizes:

- Não utilizar a conta de outro funcionário em nenhuma circunstância;
- Ao encontrar problemas técnicos, informar um administrador para que este possa entrar em contacto com a equipa de desenvolvimento;
- Em caso de inatividade de qualquer dos servidores, entrar em contacto diretamente com a equipa de manutenção.

Estas regras visam garantir um ambiente de trabalho seguro e eficiente. A cooperação de todos é fundamental para a integridade e a funcionalidade do sistema manter-se.

Controlo de acesso ao hardware

Este parâmetro foca-se na proteção de equipamentos e recursos críticos, como os servidores que executam a nossa aplicação. Apenas o administrador do sistema consegue aceder a estes servidores, pois é o único que detém as credenciais *root* de cada servidor. Além disso, o acesso é exclusivamente concedido via *SSH* com palavra-passe, sendo obrigatório o uso da *VPN* do *DEI*.

Num contexto digital em constante transformação, a gestão de acessos é essencial para a segurança e eficiência operacional da nossa aplicação *SPA*. A implementação de uma autenticação robusta protege as credenciais dos utilizadores, enquanto o controlo de acesso personalizado garante uma experiência relevante e segura. As políticas de acesso reforçam a responsabilidade e colaboração, e o controlo de acesso ao hardware preserva a integridade dos recursos críticos. Em conjunto, estas medidas não só defendem contra ameaças, mas também fortalecem a coesão da equipa e a integridade do sistema.

User Story 9

Na *User Story 9*, propõe-se a implementação justificada de um sistema de *clustering* entre os sistemas que suportam o *SPA*. Para concretizar esta *US*, é necessário que a nossa aplicação *SPA* esteja a ser executada em vários servidores. No sprint anterior, a aplicação foi instalada e configurada para operar num servidor específico (*vs484*), e neste sprint expandimos a execução para mais dois servidores (*vs1172* e *vs1248*).

Com o intuito de estabelecer um sistema de *clustering*, foi adicionado um novo servidor (*vs795*), designado como o gestor do cluster. Neste servidor, procedemos à instalação do *HAProxy* utilizando o comando "***sudo apt-get update && sudo apt-get install haproxy***".

```
root@cluster:~# sudo apt-get update && sudo apt-get install haproxy
Hit:1 http://deb.debian.org/debian bookworm InRelease
Hit:2 http://deb.debian.org/debian bookworm-updates InRelease
Hit:3 http://deb.debian.org/debian-security bookworm-security InRelease
Reading package lists... Done
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  liblua5.3-0 libopentracing-c-wrapper0 libopentracing1
Suggested packages:
  vim-haproxy haproxy-doc
The following NEW packages will be installed:
  haproxy liblua5.3-0 libopentracing-c-wrapper0 libopentracing1
0 upgraded, 4 newly installed, 0 to remove and 46 not upgraded.
Need to get 2,248 kB of archives.
After this operation, 5,131 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://deb.debian.org/debian bookworm/main amd64 liblua5.3-0 amd64 5.3.6-2 [123 kB]
Get:2 http://deb.debian.org/debian bookworm/main amd64 libopentracing1 amd64 1.6.0-4 [53.4 kB]
Get:3 http://deb.debian.org/debian bookworm/main amd64 libopentracing-c-wrapper0 amd64 1.1.3-3+b1 [29.6 kB]
Get:4 http://deb.debian.org/debian bookworm/main amd64 haproxy amd64 2.6.12-1 [2,042 kB]
Fetched 2,248 kB in 0s (4,916 kB/s)
debconf: delaying package configuration, since apt-utils is not installed
Selecting previously unselected package liblua5.3-0:amd64.
(Reading database ... 17008 files and directories currently installed.)
Preparing to unpack .../liblua5.3-0_5.3.6-2_amd64.deb ...
Unpacking liblua5.3-0:amd64 (5.3.6-2) ...
Selecting previously unselected package libopentracing1:amd64.
Preparing to unpack .../libopentracing1_1.6.0-4_amd64.deb ...
Unpacking libopentracing1:amd64 (1.6.0-4) ...
Selecting previously unselected package libopentracing-c-wrapper0:amd64.
Preparing to unpack .../libopentracing-c-wrapper0_1.1.3-3+b1_amd64.deb ...
Unpacking libopentracing-c-wrapper0:amd64 (1.1.3-3+b1) ...
Selecting previously unselected package haproxy.
Preparing to unpack .../haproxy_2.6.12-1_amd64.deb ...
Unpacking haproxy (2.6.12-1) ...
Setting up liblua5.3-0:amd64 (5.3.6-2) ...
Setting up libopentracing1:amd64 (1.6.0-4) ...
Setting up libopentracing-c-wrapper0:amd64 (1.1.3-3+b1) ...
```

Figure 1 - Instalação do HAProxy

Posteriormente, configuramos o *HAProxy*, editando o ficheiro *haproxy.cfg* através do comando "***nano /etc/haproxy/haproxy.cfg***". Neste ficheiro adicionou-se a seguinte possível configuração:

```

# Frontend Configuration
frontend http_proxy_frontend
    bind 10.9.23.27:80
    stats enable
    mode http
    option httpclose
    option forwardfor
    default_backend http_proxy_backend

# Backend Configuration
backend http_proxy_backend
    balance roundrobin
    server server1 10.9.21.228:80 check
    server server2 10.9.24.148:80 check
    server server3 10.9.24.224:80 check backup

```

Figura 13 - Possível configuração do HAProxy

Esta possível configuração revela uma pequena limitação, uma vez que o servidor *server3* (*vs1248*) só é ativado no caso de falha simultânea dos servidores *server1* (*vs484*) e *server2* (*vs1172*). Optou-se por adotar uma solução mais ideal, na qual a ativação do *server3* ocorrerá imediatamente após a detecção de falha em qualquer um dos dois servidores ativos, proporcionando uma maior robustez ao sistema. Neste sentido, foi decidido adotar uma abordagem alternativa para otimizar a resiliência da infraestrutura.

```

# Frontend Configuration
frontend http_proxy_frontend
    bind 10.9.23.27:80
    stats enable
    mode http
    option httpclose
    option forwardfor

    acl not_enough_capacity nbsrv(http_proxy_backend) le 1
    use_backend http_proxy_backend_backup if not_enough_capacity

    default_backend http_proxy_backend

# Backend Configuration
backend http_proxy_backend
    balance roundrobin
    server server1 10.9.21.228:80 check
    server server2 10.9.24.148:80 check

backend http_proxy_backend_backup
    balance roundrobin
    option allbackups
    server backup_server1 10.9.21.228:80 check backup
    server backup_server2 10.9.24.148:80 check backup
    server backup_server3 10.9.24.224:80 check backup

```

Figura 14 - Configuração do HAProxy

Esta configuração do *HAProxy* permite que a ativação do *server3* (*vs1248*) ocorra imediatamente após a detecção de falha em qualquer um dos dois servidores principais.

Esta configuração contém uma seção do *frontend*, onde gerimos o tráfego que entra no *HAProxy*, nomeamos o *frontend* como "**http_proxy_frontend**". Configurámo-lo para escutar no endereço *IP* 10.9.23.27 que corresponde ao IP do nosso servidor designado como o gestor do cluster e na porta 80. Ativamos as estatísticas para monitorização com "**stats enable**" e configurámos o modo para *HTTP*, fechando a conexão após o envio da resposta ao cliente ("**option httpclose**"). Adicionalmente, ativámos a inclusão do cabeçalho *X-Forwarded-For* para preservar informações sobre o cliente original.

Introduzimos uma condição (*ACL*) chamada "**not_enough_capacity**" para verificar se há capacidade suficiente nos servidores do *backend* "**http_proxy_backend**". Se o número de servidores disponíveis for menor ou igual a 1, a condição é acionada.

Utilizamos a instrução "**use_backend**" para redirecionar o tráfego para o *backend* de backup "**http_proxy_backend_backup**" se a condição da *ACL* for verdadeira. Caso contrário, o tráfego segue para o *backend* padrão "**http_proxy_backend**".

Na configuração do *backend* padrão, "**http_proxy_backend**", configurámos o balanceamento de carga "**roundrobin**" com dois servidores, "**server1**" (*vs484*) e "**server2**" (*vs1172*).

Para o novo *backend* de backup, "**http_proxy_backend_backup**", configurámos o balanceamento "**roundrobin**" e adicionámos a opção "**allbackups**", indicando que todos os servidores neste *backend* são considerados *backups*. Incluímos três servidores de backup: "**backup_server1**" (*vs484*), "**backup_server2**" (*vs1172*) e "**backup_server3**" (*vs1248*), cada um com o seu respetivo endereço IP e porta. A opção "**check backup**" indica que estes servidores serão ativados em caso de falha nos servidores principais do *backend* padrão.

Com esta configuração do *HAProxy* pretende-se aprimorar a resiliência do sistema, ativando automaticamente os servidores de *backup* quando a capacidade dos servidores principais é insuficiente.

Através do comando "**haproxy -c -f /etc/haproxy/haproxy.cfg**" é possível verificar se a configuração do *HAProxy* encontra-se válida.

```
root@cluster:~# haproxy -c -f /etc/haproxy/haproxy.cfg
Configuration file is valid
```

Figura 15 - Validação da configuração do *HAProxy*

Por último, reiniciamos o *HAProxy* para aplicar as alterações, através do comando "**service haproxy restart**".

```
root@cluster:~# service haproxy restart
```

Figura 16 - Aplicação das alterações do HAProxy

Para monitorizar de forma mais eficaz o desempenho e o estado dos servidores vamos utilizar a interface de estatísticas do HAProxy que encontra-se em `/haproxy?stats`. Ao efetuar a monitorização, observamos que, no nosso backend `"http_proxy_backend"`, os servidores 1 e 2 estão operacionais. No backend `"http_proxy_backend_backup"`, todos os servidores encontram-se atualmente em modo de backup, conforme esperado.

Note: NO LB / DRAIN = UP with load-balancing disabled.

http_proxy_frontend																					
	Queue			Session rate			Sessions						Bytes		Denied		Errors		Warnings		Status
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	
Frontend				2	2	-	2	3		524 265	2		0	0	0	0	0				OPEN

http_proxy_backend																					
	Queue			Session rate			Sessions						Bytes		Denied		Errors		Warnings		Status
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	
server1	0	0	-	0	0		0	0	-	0	0	?	0	0		0		0	0	0	3s UP
server2	0	0	-	0	0		0	0	-	0	0	?	0	0		0		0	0	0	3s UP
Backend	0	0		0	0		0	0	52 427	0	0	?	0	0	0	0		0	0	0	3s UP

http_proxy_backend_backup																					
	Queue			Session rate			Sessions						Bytes		Denied		Errors		Warnings		Status
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	
backup_server1	0	0	-	0	0		0	0		-	0	0	?	0	0		0		0	0	3s UP
backup_server2	0	0	-	0	0		0	0		-	0	0	?	0	0		0		0	0	3s UP
backup_server3	0	0	-	0	0		0	0		-	0	0	?	0	0		0		0	0	3s UP
Backend	0	0		0	0		0	0	52 427	0	0	?	0	0	0	0		0	0	0	3s UP

Figura 17 - Monitorização do desempenho e estado dos servidores

Vamos agora realizar um teste de balanceamento de carga. Ao efetuar múltiplos pedidos ao nosso servidor através do comando de uma linha `"while true; do curl 10.9.23.27:80; sleep 1; done"`, e ao examinar a interface de estatísticas, constatamos que a carga foi distribuída de forma equitativa entre os servidores 1 e 2, conforme esperado.

http_proxy_frontend																						
	Queue			Session rate			Sessions				LbTot	Last	Bytes		Denied		Errors		Warnings		Status	
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total			In	Out	Req	Resp	Req	Conn	Resp	Retr		Redis
Frontend				1	1	-	1	2		524 265	51		3 850	542 350	0	0	0				OPEN	

http_proxy_backend																						
	Queue			Session rate			Sessions				LbTot	Last	Bytes		Denied		Errors		Warnings		Status	
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total			In	Out	Req	Resp	Req	Conn	Resp	Retr		Redis
server1	0	0	-	0	1		0	1		25	25	8s	1 925	271 175		0	0	0	0	0	1m2s UP	
server2	0	0	-	0	1		0	1		25	25	8s	1 925	271 175		0	0	0	0	0	1m2s UP	
Backend	0	0		0	1		0	1		52 427	50	8s	3 850	542 350	0	0	0	0	0	0	1m2s UP	

http_proxy_backend_backup																						
	Queue			Session rate			Sessions				LbTot	Last	Bytes		Denied		Errors		Warnings		Status	
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total			In	Out	Req	Resp	Req	Conn	Resp	Retr		Redis
backup_server1	0	0	-	0	0		0	0		-	0	?	0	0		0	0	0	0	0	1m2s UP	
backup_server2	0	0	-	0	0		0	0		-	0	?	0	0		0	0	0	0	0	1m2s UP	
backup_server3	0	0	-	0	0		0	0		-	0	?	0	0		0	0	0	0	0	1m2s UP	
Backend	0	0		0	0		0	0		52 427	0	?	0	0	0	0	0	0	0	0	1m2s UP	

Figura 18 - Realização de teste de balanceamento de carga

Vamos agora realizar um teste de failover. Para isso, iremos desativar temporariamente o servidor `server1` e observar como o sistema responde à falha. Ao continuar a efetuar pedidos ao nosso servidor e monitorar a interface de estatísticas, pretendemos avaliar a capacidade de *failover* do sistema.

Idealmente, o HAProxy deverá redirecionar automaticamente o tráfego para os servidores operacionais (servidores de backup) sem interrupções significativas no serviço.

Este teste permitirá verificar a robustez da nossa configuração, assegurando que a aplicação continua a funcionar de forma estável mesmo em situações de falha.

http_proxy_frontend																			
	Queue			Session rate			Sessions				Bytes			Denied		Errors		Warnings	
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Redis
Frontend				1	2	-	1	2		524 265	103		8 241	1 113 218	0	0	1		OPEN
http_proxy_backend																			
	Queue			Session rate			Sessions				Bytes			Denied		Errors		Warnings	
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Redis
server1	0	0	-	0	1	0	1			25	25	13m/s	1 925	271 175	0	0	0	0	1m/s DOWN
server2	0	0	-	0	1	0	1			25	25	13m/s	1 925	271 175	0	0	0	0	14m UP
Backend	0	0		0	1		0	1	52 427	50	50	13m/s	3 850	542 350	0	0	0	0	14m UP
http_proxy_backend_backup																			
	Queue			Session rate			Sessions				Bytes			Denied		Errors		Warnings	
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Redis
backup_server1	0	0	-	0	0	0	0	0	0	-	0	0	?	0	0	0	0	0	1m/s DOWN
backup_server2	0	0	-	0	1	0	1			25	25	3s	1 925	271 175	0	0	0	0	14m UP
backup_server3	0	0	-	1	1	0	1			25	25	2s	1 925	271 150	0	0	0	0	14m UP
Backend	0	0		1	1	0	1			52 427	50	2s	3 850	542 325	0	0	0	0	14m UP

Figura 19 - Realização de teste de failover

Em suma, a opção de implementar quatro servidores, incluindo um dedicado à gestão do cluster (vs795), foi tomada com o objetivo de encontrar uma solução que conjuga eficiência, escalabilidade e disponibilidade para a nossa aplicação SPA. Ao centralizar as operações de gestão no servidor vs795, simplificamos a administração do cluster. A distribuição da execução da aplicação pelos servidores vs484 e vs1172 melhora a capacidade de resposta e a escalabilidade. A integração do HAProxy na configuração proporciona um balanceamento de carga eficiente, maximizando a utilização dos recursos disponíveis. Além disso, ao designar o servidor vs1248 como *backup* e integrar as opções de *backup* no HAProxy, adicionamos uma camada adicional de redundância. Isso contribui para a robustez e a disponibilidade contínua da aplicação, mesmo em situações de falhas imprevistas. Assim, a arquitetura adotada visa maximizar o desempenho, a eficiência e a fiabilidade da nossa aplicação, garantindo uma experiência consistente para os utilizadores, mesmo perante eventos inesperados.

User Story 10

Nesta user story propõe-se que o administrador de sistemas tenha um acesso SSH à máquina virtual, apenas por certificado, sem recurso a password.

Para isso foi necessário permitir o acesso ao root por ssh, no “/etc/ssh/sshd_config”.

```
GNU nano 5.4 /etc/ssh/sshd_config
# Authentication:

#LoginGraceTime 2m
PermitRootLogin yes
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10
```

Depois foram geradas as keys públicas e privadas com o comando “ssh-keygen -t rsa”. Estas keys foram guardadas no “/root/.ssh/id_rsa.pub” e “/root/.ssh/id_rsa”, respetivamente.

```
root@uvm050:~# ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa
Your public key has been saved in /root/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:E8iE8YoodyqrVpWxGK5Soek7FtxBXTDsCtwLsb1feGY root@uvm050
The key's randomart image is:
+---[RSA 3072]-----+
|      +=+.          |
| ..o.B..           |
|.oB.+ B .          |
|o*.B *             |
|= .B O . S         |
|o* O . E .         |
|o = . =            |
| B .               |
|* .                |
+---[SHA256]-----+
```

De seguida foi criado um ficheiro “authorized_keys”, também no “/root/.ssh/”, para guardar as chaves públicas permitidas para se fazer login.

```
root@uvm050:~# ls /root/.ssh/
authorized_keys  id_rsa  id_rsa.pub  known_hosts
```

Seguidamente copiou-se o conteúdo do “id_rsa.pub” para o “authorized_keys”, com o comando “ssh-copy-id -i /root/.ssh/id_rsa.pub root@10.9.10.50”.

```
root@uvm050:~# ssh-copy-id -i /root/.ssh/id_rsa.pub root@10.9.10.50
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/root/.ssh/id_rsa.pub"
The authenticity of host '10.9.10.50 (10.9.10.50)' can't be established.
ECDSA key fingerprint is SHA256:kD2KGGGCjptYreTTd0z6o+SzwEkSuHP50BXq5ScZgfA.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
-----
--0000000000--0000000000--0000000000--000--000--0000000000--0000000000--0000000000--
--0000000000--0000000000--0000000000--000--000--0000000000--0000000000--0000000000--
--000--000--000--000--000--000--000--000--000--000--000--000--000--000--000--000--
--000--000000--000--000--000--000--000--000--000--000--000--0000000000--000--000--
--000--000000--00000000--000--000--000--000--0000000000--0000000000--000--000--
--000--000--0000000000--000--000--000--000--000--0000000000--0000000000--000--000--
--0000000000--000--000--0000000000--0000000000--000--0000000000--0000000000--
--0000000000--000--000--0000000000--0000000000--000--0000000000--0000000000--
-----

Welcome comrade, this is our server uvm050!

root@10.9.10.50's password:

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'root@10.9.10.50'"
and check to make sure that only the key(s) you wanted were added.
```

Depois foi acrescentado ao “/etc/ssh/sshd_config”, “PubKeyAuthentication yes” e “PasswordAuthentication no”.

```
GNU nano 5.4 /etc/ssh/sshd_config

PubkeyAuthentication yes

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

#AuthorizedPrincipalsFile none

#AuthorizedKeysCommand none
#AuthorizedKeysCommandUser nobody

# For this to work you will also need host keys in /etc/ssh/ssh_known_hosts
#HostbasedAuthentication no
# Change to yes if you don't trust ~/.ssh/known_hosts for
# HostbasedAuthentication
#IgnoreUserKnownHosts no
# Don't read the user's ~/.rhosts and ~/.shosts files
#IgnoreRhosts yes

# To disable tunneled clear text passwords, change to no here!
PasswordAuthentication no
#PermitEmptyPasswords no
```

Depois do passo anterior é necessário reiniciar o serviço ssh com o comando “systemctl restart ssh”.

User Story 11

Nesta *User Story (US)*, é pedido que “como administrador de sistemas quero que para agilização entre as várias equipas seja criada uma partilha pública de ficheiros, formato SMB/CIFS ou NFS”.

Para a realização desta *US*, escolheu-se a utilização do formato SMB/CIFS para que a partilha também seja possível para clientes Windows. Caso se pretendesse fazer uma partilha apenas para clientes Linux, poderíamos utilizar antes o formato NFS, pois é o mais indicado para essa situação.

Esta partilha irá ser feita para que as pessoas das diferentes equipas, possam aceder a manuais de ajuda sobre a aplicação (tutoriais/instruções), bem como a avisos/comunicados que possam ser necessários fazer, ou seja, a comunicação irá fluir apenas num sentido.

Após tomar as decisões acima mencionadas, começou-se por instalar o Samba com o comando “*sudo apt install samba*”. Para que fosse possível partilhar uma pasta, configurou-se uma nova definição no final do ficheiro “*/etc/samba/smb.conf*”, em que se criou uma partilha com o nome “*RobDroneGo*”, como é possível visualizar na imagem seguinte.

```
# Partilha que irá conter tutoriais/instruções e  
# comunicados referentes ao projeto RobDroneGo  
[RobDroneGo]  
    path = /RobDroneGo  
    browsable = yes  
    guest ok = yes  
    read only = yes
```

Figura 20 - Configuração da partilha de ficheiros no formato SMB/CIFS

Nesta configuração, podemos então ver que o nome da partilha é “*RobDroneGo*”, como foi dito anteriormente, que é o nome que os utilizadores utilizam para aceder a esta partilha. A pasta do nosso servidor que é utilizada para fazer esta partilha é a pasta “*/RobDroneGo*”, sendo que todos os ficheiros da partilha estarão aqui nesta pasta. A opção “*browsable = yes*” determina se a partilha é visível ao navegar pela rede. No nosso caso ela está definida como “*yes*”, então a partilha é exibida quando os utilizadores procuram por partilhas disponíveis no nosso servidor.

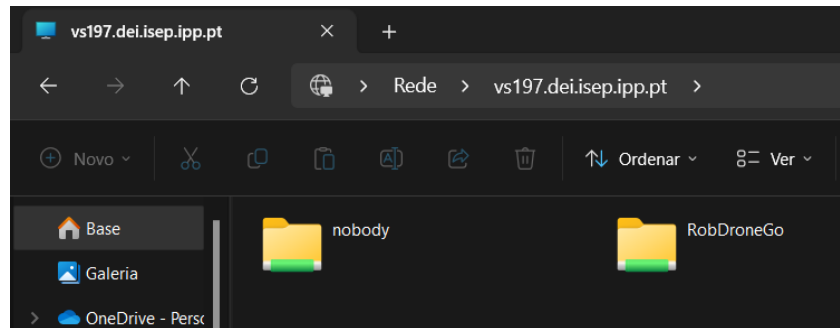


Figura 21 - Efeito da opção “browsable = yes”

A opção “*guest ok = yes*” permite o acesso de convidado (acesso sem autenticação). Como no nosso caso utilizamos a opção “*yes*”, os utilizadores podem aceder à partilha sem efetuar o *login*. Caso quiséssemos que fosse obrigatório autenticar para aceder à partilha, teríamos de colocar esta opção como “*no*”.

Já a opção “*read only = yes*” configura a partilha como apenas de leitura, sendo que esta opção no nosso caso poderia ser omitida, visto que por defeito a partilha samba é apenas de leitura. Os utilizadores podem visualizar e copiar os ficheiros da partilha, mas não podem modificar ou criar ficheiros. Caso fosse decidido que todos os utilizadores poderiam utilizar a partilha para criar ou editar os ficheiros, esta opção deveria ser colocada como “*no*”.

Ao configurar esta partilha, colocou-se o caminho da partilha para a pasta do servidor “*/RobDroneGo*”, sendo que ela ainda não existia no projeto, então foi necessário criá-la através do comando “*mkdir /RobDroneGo*”.

Agora para esta partilha começar a funcionar, é necessário reiniciar os dois serviços compreendidos pelo Samba, sendo eles o *nmbd* e o *smbd*, sendo que para isso utilizou-se os comandos “*systemctl restart nmbd*” e “*systemctl restart smbd*”.

Por fim, já podemos testar se a partilha já se encontra em funcionamento. Para isso, criou-se um ficheiro de boas-vindas na pasta da partilha com um pequeno texto para verificarmos se ele será acessível através da partilha. O comando utilizado para criar o ficheiro com o pequeno texto de boas-vindas, pode ser visualizado na imagem seguinte.

```
root@vs197:~# echo "Welcome to RobDroneGo project from group 50! We Hope you enjoy it." > /RobDroneGo/Welcome.txt
root@vs197:~#
```

Figura 22 - Criação de um ficheiro com um pequeno texto de boas-vindas na partilha “*RobDroneGo*”

Para testar esta partilha, foi utilizado um cliente Windows, em que se abriu o “*File Explorer*” e procurou-se pela partilha do servidor através do caminho “*\\vs197.dei.isep.ipp.pt\RobDroneGo*”.

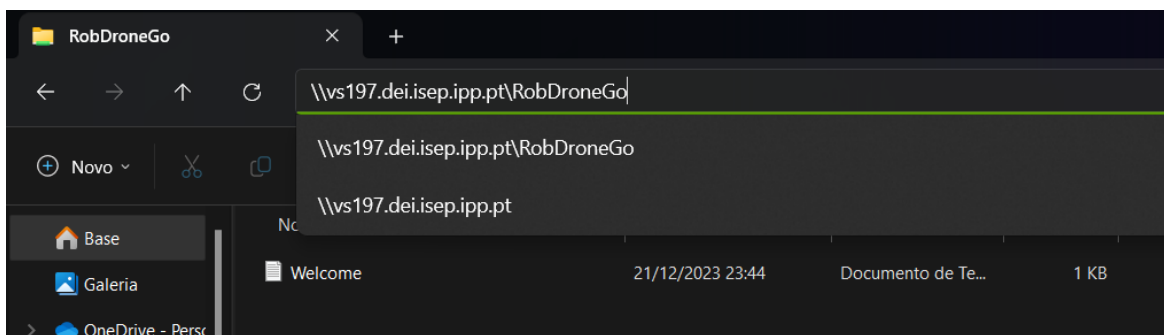


Figura 23 - Acesso à pasta da partilha através de um cliente Windows

Ao abrir o ficheiro “Welcome.txt”, podemos ler a mensagem que lá existe.

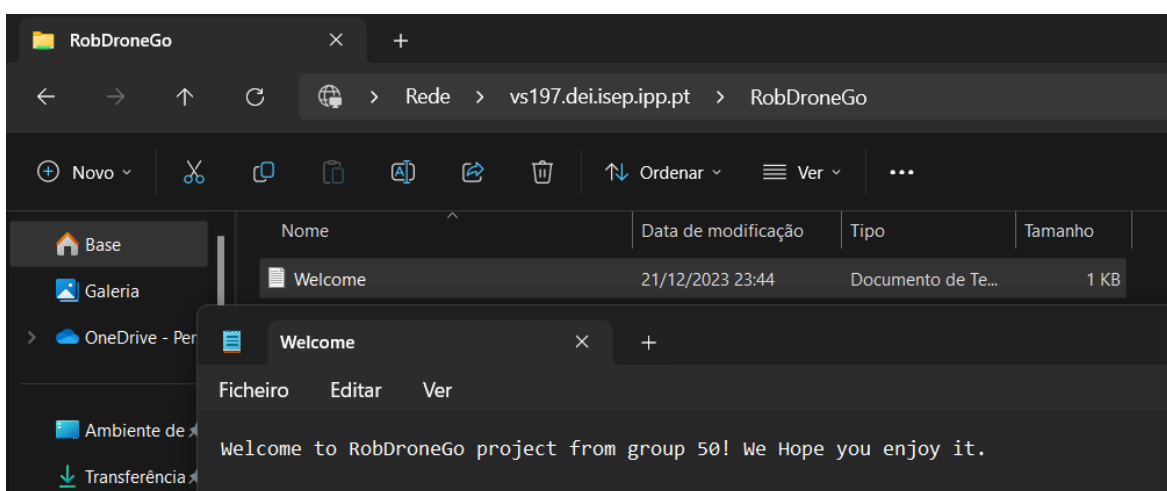


Figura 24 - Conteúdo do ficheiro da partilha

Se tentarmos alterar o conteúdo deste ficheiro e guardá-lo tentando substituir o ficheiro atual, obtemos um aviso sobre não termos permissões para isso e nada é alterado, tal como nós queríamos!

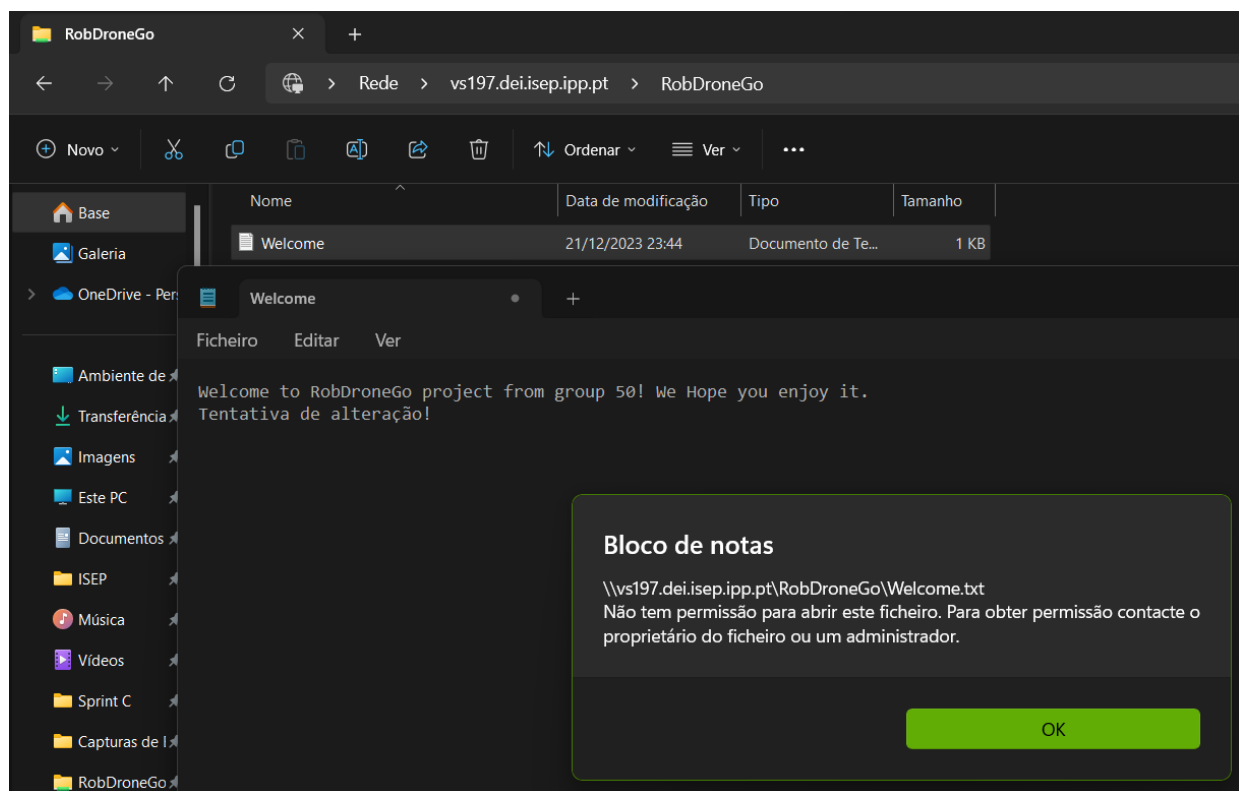


Figura 25 - Negação de alteração nos ficheiros da partilha criada

Se tentarmos também criar um ficheiro nesta partilha, esta operação também será negada.

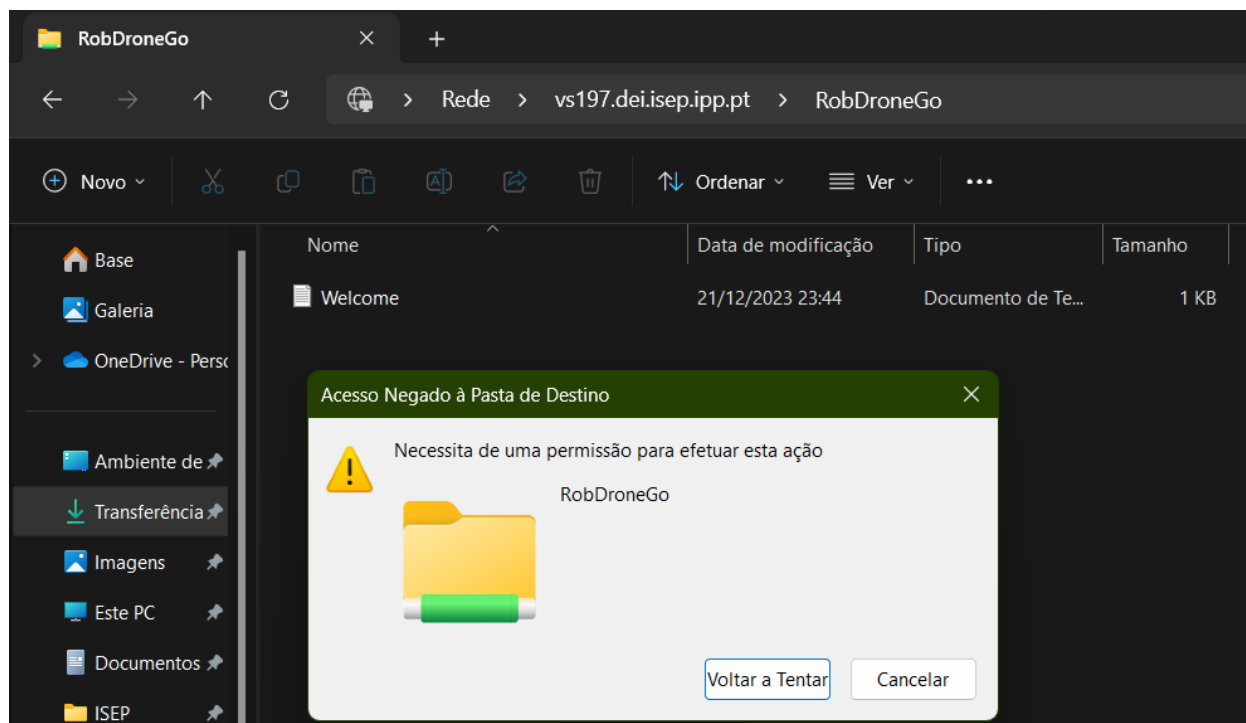


Figura 26 - Acesso negado à tentativa de criação de novo ficheiro na partilha

User Story 12

A seguinte US visa automatizar a reposição de backups, validando no final o funcionamento do sistema.

Para tal foi desenvolvido o script **restore_and_validate_backup.sh**:

```
#!/bin/bash

MONGODB_URI="mongodb://mongoadmin:1f8402684f9645194a515c31@vs624.dei.isep.ipp.pt:27017/RobDroneGo?authSource=admin"
BACKUP_FILE="$1"
TEMP_DIR="/tmp/restore_temp"
COLLECTIONS=("buildings" "devices" "devicetypes" "passageways" "surveillancetasks" "users")

restore_mongodb_backup() {
    echo "Restoring MongoDB backup to database..."

    mkdir -p $TEMP_DIR

    tar -xzf $BACKUP_FILE -C $TEMP_DIR

    mongorestore --uri $MONGODB_URI --dir $TEMP_DIR --nsInclude RobDroneGo.*

    if [ $? -eq 0 ]; then
        echo "Backup restored to the database successfully."
        return 0
    else
        echo "Error restoring the backup to the database."
        return 1
    fi
}

validate_collections() {
    echo "Validating the existence of collections in the database..."

    for collection in "${COLLECTIONS[@]}; do
        # check if the collection exists
        mongosh $MONGODB_URI --eval "printjson(db.${collection}.count())"

        if [ $? -eq 0 ]; then
            echo "Collection '${collection}' exists."
        else
            echo "Collection '${collection}' does not exist or an error occurred."
            return 1
        fi
    done

    echo "Validation successful."
    return 0
}

# clean up temporary directory
cleanup() {
    rm -rf $TEMP_DIR
}

# execute the restoration and validation process
restore_mongodb_backup
if [ $? -eq 0 ]; then
    validate_collections
    cleanup
else
    echo "Failed to restore the backup to the database. Check the logs for more details."
fi
```

Figura 27 - *restore_and_validate_backup.sh*

A função **restore_mongodb_backup()** extrai o backup passado por parâmetro e com o comando **mongorestore** restaura o backup na base de dados, verificando se a restauração foi bem-sucedida com base no código de saída.

A função **validate_collections()** executa uma *query* que verifica se as coleções existem na base de dados. Finalmente é feito o *cleanup*, apagando o backup extraído.

```

root@vs197:/etc# ./restore_and_validate_backup.sh /etc/daily_database_backup/RobDroneGo_20231223.tgz
Restoring MongoDB backup to database...
2023-12-26T18:18:12.339+0000 The --db and --collection flags are deprecated for this use-case; please use --nsInclude instead, i.e. with --nsInclude=${DATABASE}.${COLLECTION}
2023-12-26T18:18:12.346+0000 building a list of collections to restore from /tmp/restore_temp_dir
2023-12-26T18:18:12.352+0000 0 document(s) restored successfully. 0 document(s) failed to restore.
Backup restored to the database successfully.
Validating the existence of collections in database...
Current Mongosh Log ID: 658b170750ea999275b6e6eb3
Connecting to: mongosh://<credentials>@vs624.dei.isep.ipp.pt:27017/RobDroneGo7authSource=admin&directConnection=true&appName=mongosh+2.1.1
Using MongoDB: 7.0.4
Using Mongosh: 2.1.1

For mongosh info see: https://docs.mongodb.com/mongosh-shell/

-----

The server generated these startup warnings when booting
2023-12-17T09:18:36.215+00:00: The configured WiredTiger cache size is more than 80% of available RAM. See http://dochub.mongodb.org/core/faq-memory-diagnostics-wt
2023-12-17T09:18:45.567+00:00: vm.max_map_count is too low

-----

4
Collection 'buildings' exists.
Current Mongosh Log ID: 658b17149f64bdf2de394515
Connecting to: mongosh://<credentials>@vs624.dei.isep.ipp.pt:27017/RobDroneGo7authSource=admin&directConnection=true&appName=mongosh+2.1.1
Using MongoDB: 7.0.4
Using Mongosh: 2.1.1

For mongosh info see: https://docs.mongodb.com/mongosh-shell/

-----

The server generated these startup warnings when booting
2023-12-17T09:18:36.215+00:00: The configured WiredTiger cache size is more than 80% of available RAM. See http://dochub.mongodb.org/core/faq-memory-diagnostics-wt
2023-12-17T09:18:45.567+00:00: vm.max_map_count is too low

-----

0
Collection 'devices' exists.
Current Mongosh Log ID: 658b171dc53883e8fbf66894
Connecting to: mongosh://<credentials>@vs624.dei.isep.ipp.pt:27017/RobDroneGo7authSource=admin&directConnection=true&appName=mongosh+2.1.1
Using MongoDB: 7.0.4
Using Mongosh: 2.1.1

For mongosh info see: https://docs.mongodb.com/mongosh-shell/

-----

The server generated these startup warnings when booting
2023-12-17T09:18:36.215+00:00: The configured WiredTiger cache size is more than 80% of available RAM. See http://dochub.mongodb.org/core/faq-memory-diagnostics-wt
2023-12-17T09:18:45.567+00:00: vm.max_map_count is too low

-----

0
Collection 'devicetypes' exists.
Current Mongosh Log ID: 658b17271633bcf56238899f
Connecting to: mongosh://<credentials>@vs624.dei.isep.ipp.pt:27017/RobDroneGo7authSource=admin&directConnection=true&appName=mongosh+2.1.1
Using MongoDB: 7.0.4
Using Mongosh: 2.1.1

For mongosh info see: https://docs.mongodb.com/mongosh-shell/

-----

The server generated these startup warnings when booting
2023-12-17T09:18:36.215+00:00: The configured WiredTiger cache size is more than 80% of available RAM. See http://dochub.mongodb.org/core/faq-memory-diagnostics-wt
2023-12-17T09:18:45.567+00:00: vm.max_map_count is too low

-----

12
Collection 'passageways' exists.
Current Mongosh Log ID: 658b17310a45164166f03170
Connecting to: mongosh://<credentials>@vs624.dei.isep.ipp.pt:27017/RobDroneGo7authSource=admin&directConnection=true&appName=mongosh+2.1.1
Using MongoDB: 7.0.4
Using Mongosh: 2.1.1

For mongosh info see: https://docs.mongodb.com/mongosh-shell/

-----

The server generated these startup warnings when booting
2023-12-17T09:18:36.215+00:00: The configured WiredTiger cache size is more than 80% of available RAM. See http://dochub.mongodb.org/core/faq-memory-diagnostics-wt
2023-12-17T09:18:45.567+00:00: vm.max_map_count is too low

-----

0
Collection 'surveillancetasks' exists.
Current Mongosh Log ID: 658b173b00df4bcbe57cd818
Connecting to: mongosh://<credentials>@vs624.dei.isep.ipp.pt:27017/RobDroneGo7authSource=admin&directConnection=true&appName=mongosh+2.1.1
Using MongoDB: 7.0.4
Using Mongosh: 2.1.1

For mongosh info see: https://docs.mongodb.com/mongosh-shell/

-----

The server generated these startup warnings when booting
2023-12-17T09:18:36.215+00:00: The configured WiredTiger cache size is more than 80% of available RAM. See http://dochub.mongodb.org/core/faq-memory-diagnostics-wt
2023-12-17T09:18:45.567+00:00: vm.max_map_count is too low

-----

5
Collection 'users' exists.
Validation successful.

```

Figura 28 - Exemplo de execução de `restore_and_validate_backup.sh`