

# **Relatório de ASIST**

## **Sprint 3**

**Turma 3DG – Grupo 038**

**1190718 – João Beires**

**1190782 – José Soares**

**1190811 – Lourenço Melo**

**1191419 – José Maia**

# User Story C1

## Contactos de emergência

Nome	Função	Email	Telemóvel
José Maia	Administrador	<a href="mailto:1191419@isep.ipp.pt">1191419@isep.ipp.pt</a>	--- --- ---
José Soares	Técnico do sistema de monitorização	<a href="mailto:1190782@isep.ipp.pt">1190782@isep.ipp.pt</a>	--- --- ---
João Beires	Chefe do comité executivo de DRP	<a href="mailto:1190718@isep.ipp.pt">1190718@isep.ipp.pt</a>	--- --- ---
Lourenço Melo	Chefe da equipa de Resposta a Incidentes	<a href="mailto:1190811@isep.ipp.pt">1190811@isep.ipp.pt</a>	--- --- ---

### 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).

### Pré-planeamento e comité executivo de DRP

Primeiramente, foi criado um comité executivo que tem como objetivo planear e atualizar o DRP e orientar e fornecer apoio às equipas de projetos em assuntos referentes ao DRP. Os gerentes de projeto devem trabalhar com o comité 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.

## **Análise dos sistemas informáticos e avaliação da vulnerabilidade**

São feitos backups noutra plataforma para a salvaguarda dos dados gerais. Estes backups serão feitos de 4 em 4 horas para um “*site backup*” que se localiza a mais de 10 km de distância e com acesso controlado.

O único acesso ao exterior é através da internet e, para isso, existem firewalls para evitar acessos não autorizados.

Existe um gerador de energia que consegue alimentar os servidores e a base de dados durante um certo espaço de tempo se houver uma falha de energia.

### **Sistemas e equipamentos informáticos usados:**

- Luzes;
- Computadores fixos, monitores, teclados e ratos usados diariamente pelos trabalhadores;
- Computadores portáteis tanto pessoais como da empresa utilizados pelos colaboradores;
- Servidores da empresa;
- Sistema de videovigilância, que se encontra ligado durante a totalidade do dia. Em horas trabalho conectado ao centro de vigia do segurança do edifício;
- Sensores e alarmes de incêndio;
- Termómetros e sensores de humidade nas salas dos servidores e base de dados, para manter estas nas condições ideais ao funcionamento dos aparelhos;
- Alarmes anti roubo;
- Sistema de picagem do ponto e de autorização para entrar nas instalações, que usa o cartão de trabalhador para proceder a estas autenticações;
- Base de dados onde se armazena as informações necessárias ao negócio;

**Avaliação da vulnerabilidade:**

A Matriz de Riscos ou Matriz de Probabilidade permite de forma visual identificar quais são os riscos que devem receber mais atenção.

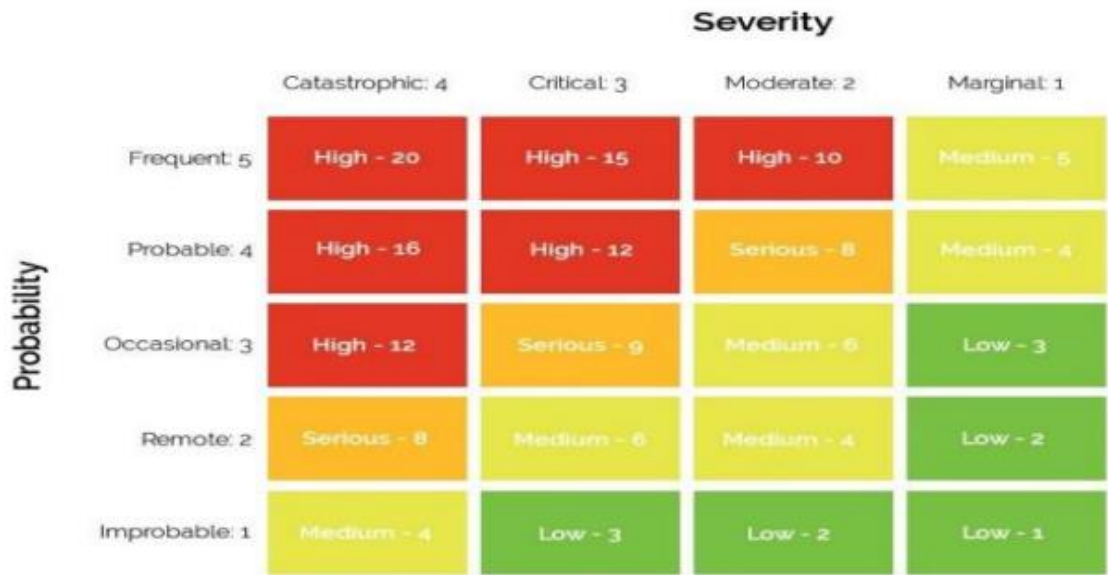


Figura 1 - Matriz de Risco

Ameaça	Probabilidade	Consequência	Risco
Falha energética	3	1	Baixo - 3
Sismo	1	3	Baixo - 3
Avaria num dos componentes informáticos	2	2	Médio - 4
Incêndio no data center	1	4	Médio - 4
Perda de informações essenciais	3	2	Médio - 6
Incêndio	3	3	Séria - 9
Ataques DDoS e DoS	3	3	Séria - 9
Falha da VM do DEI	4	2	Séria - 8
Falha nos servidores	3	3	Séria - 9
Sobrecarga do sistema	4	3	Alta - 12

## **Monitorização de falhas**

Foi concebido um sistema automatizado de monitorização de falhas que se encontra sempre ligado, e executa testes e verifica as condições dos componentes necessários ao negócio. Para isso, também foi necessário formar um técnico com capacidade de verificar o bom funcionamento deste sistema e atualizá-lo se necessário (técnico do sistema de monitorização).

Alguns destes testes são:

- Verificação da temperatura e da humidade ideal nas salas dos servidores e base de dados;
- Bom funcionamento dos servidores, testando a velocidade de resposta dos mesmos.
- Testa o bom funcionamento e rapidez da internet nas instalações;
- Sistema de deteção de intrusão na rede;
- Teste do bom funcionamento do gerador de energia;

## **Prevenção de falhas**

Para além de avaliar o tipo de falhas e monitorizá-las também é necessário precavê-las. Assim, a organização utiliza métodos de prevenção, tais como:

- Segurança para controlo do acesso físico às instalações;
- Temperatura e humidade controlada;
- Utilização de um gerador de energia para o caso de uma falha de energia;
- Sistemas de combate a incêndios, inundações e sismos;
- Uso de equipamentos de qualidade;
- Equipamentos extra caso haja a necessidade de troca;
- Sistema automatizado de monitorização de falhas;
- Educação e formação dos colaboradores e utilizadores do sistema;
- Atualizações e melhoramentos de software, componentes e instalações;
- Utilização de firewalls, e equipa de cyber segurança;
- Backups para outra plataforma para a salvaguarda dos dados necessários ao negócio;

## **Desastre**

**Acontecendo um desastre**, uma equipa de Resposta a Incidentes será chamada e terá como principais funções a recuperação e avaliação dos danos. A equipa tentará chegar ao local afetado o mais rapidamente possível e começa imediatamente a tratar do problema.

**Primeiramente, a equipa irá avaliar os danos**, percebendo que consequências este desastre teve para o bom funcionamento do negócio. A equipa terá testes pré-preparados e especialistas para a dada ocasião e assim consegue avaliar rapidamente os danos.

**Após a avaliação ser feita**, os planos de recuperação serão postos em prática. A equipa começará por notificar a equipa de suporte técnico para fornecer ajuda e explicação do sucedido a qualquer utilizador que esteja a ter problemas devido a este acontecimento.

A equipa irá coordenar esforços e recursos necessários á resolução do problema, por exemplo:

- Necessidade do uso dos dados de backup;
- Necessidade de chamar certos especialistas, técnicos e/ou equipas para a resolução de problemas específicos;
- Pedido de novos equipamentos ou componentes que foram danificados;
- Alerta das autoridades, se necessário.
- Alertar a organização do estado de emergência, de modo que todos os esforços necessários sejam focados no restauro do sistema;

**Caso a resolução do problema se torne demorada**, as equipa irá trabalhar em turnos de 4 horas para não haver sobrecarga de trabalho e fazer com que este seja o mais eficiente possível.

A manutenção normal da aplicação, enquanto a situação não se encontrar regularizada, não irá ser feita.

**Após a adversidade ter sido resolvida** a empresa será notificada, e a equipa de Resposta a incidentes (ERI) irá fazer um relatório detalhado sobre o sucedido. Depois da situação ser regularizada a organização voltará ao seu funcionamento normal e o relatório do desastre será depois analisado e discutido pela ERI, o comitê executivo de DRP e os administradores para reflexão do acontecimento e a tomada de medidas, se necessário.

## User story C2

O MTD (Maximum Tolerable Downtime) é o período máximo durante o qual uma empresa ou organização pode ficar sem acesso a um sistema ou serviço crítico antes que isso cause danos significativos à operação da empresa. O MTD é um parâmetro importante na avaliação da necessidade de investimento em medidas de proteção e tolerância a falhas, como sistemas de backup e alta disponibilidade.

De forma a assegurar um MTD de 20 minutos, é necessário implementar medidas de proteção e tolerância a falhas em todas as camadas da sua infraestrutura de TI.

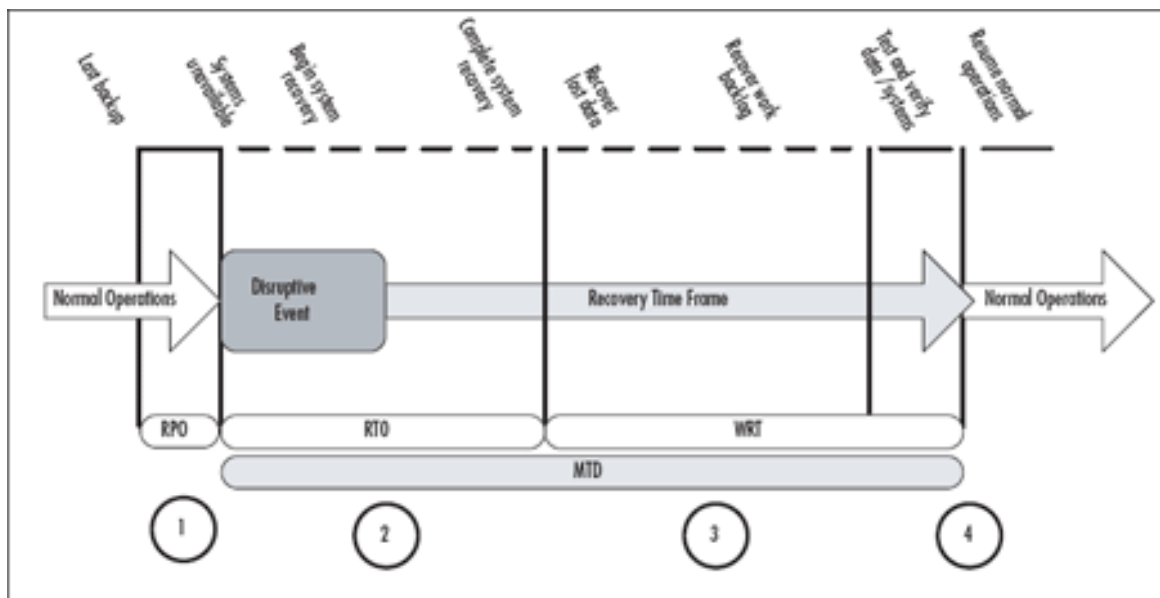


Figura 2 - Análise no impacto do negócio

Neste tópico há alguns pontos que são importantes referir e explicar o que são.

1. **RTO – Recovery Time Objective**, é uma técnica para calcular o tempo máximo que o sistema levaria para voltar à sua normalidade após um problema/erro. Para calcular este tempo temos de ter em atenção as prioridades do negócio.
2. **WRT – Work Recovery Time**, é o tempo necessário máximo para verificar com alguns testes todos os sistemas, por exemplo database, apps e sistemas. Cada vez mais esta etapa se procura que precise menos do esforço humano, assim a automatização é cada vez mais um objetivo e ponto fulcral desta etapa.
3. **MTD – Maximum Tolerable Downtime**, será então o tempo máximo que os sistemas podem estar down. Dependendo da prioridade dos softwares utilizados numa organização, pode haver diferentes MTD's para diferentes departamentos/sistemas/softwares da empresa.

Para concluir apresentamos algumas soluções para assegurar um MTD de 20 minutos:

- Utilizar sistemas de backup e recuperação de desastres para garantir que os dados críticos sejam protegidos e possam ser recuperados rapidamente em caso de falha.
- Implementar sistemas de alta disponibilidade, como clusters de servidores, para garantir que o sistema possa continuar a funcionar mesmo que um componente falhe.
- Realizar testes de failover regularmente para garantir que os sistemas de backup estão a funcionar corretamente e que a transição para os sistemas de backup é suave.
- Implementar medidas de proteção contra falhas humanas, como treino adequado e políticas de mudança controladas para minimizar a possibilidade de erros humanos.
- Manter os sistemas atualizados com as últimas correções de segurança e patches de software para minimizar a probabilidade de falhas.



## User Story C3

Para a realização desta US, criamos este script:

```
GNU nano 5.4                                dbbackup.sh
#!/bin/bash
BCDAY=$(date +%d)
BCLASTWEEK=$(date +%W)
BCMONTH=$(date +%m)
BCDAYWEEK=$(date +%u)
BCDAYYEAR=$(date +%j)
BCDATE=$(date +%Y%m%d)
if [ $BCDAYYEAR = "1" ];
then
    mkdir /root/backup/mongo/$BCDATE
    cd /root/backup/mongo/$BCDATE
    mongodump --uri="mongodb+srv://cluster0.1c3bjmr.mongodb.net/test" -u admin -p Pinguins1
    tar zcvf $BCDATE.tgz ./dump/
    cd /root/repoMongoDB/mongodb
    git pull
    mv /root/backup/mongo/$BCDATE/dump /root/backup/mongo/$BCDATE/test_$BCDATE
    mv /root/backup/mongo/$BCDATE/test_$BCDATE /root/repoMongoDB/mongodb/
    git add --all
    git commit -m "New Backup!"
    git push
elif [ $BCDAY = "1" ];
then
    mkdir /root/backup/mongo/$BCDATE
    cd /root/backup/mongo/$BCDATE
    mongodump --uri="mongodb+srv://cluster0.1c3bjmr.mongodb.net/test" -u admin -p Pinguins1
    tar zcvf $BCDATE.tgz ./dump/
    cd /root/repoMongoDB/mongodb
    git pull
    mv /root/backup/mongo/$BCDATE/dump /root/backup/mongo/$BCDATE/test_$BCDATE
    mv /root/backup/mongo/$BCDATE/test_$BCDATE /root/repoMongoDB/mongodb/
    git add --all
    git commit -m "New Backup!"
    git push
elif [ $BCMONTH = "12" ] && [ $BCDAYWEEK = "1" ];
then
    mkdir /root/backup/mongo/$BCDATE
    cd /root/backup/mongo/$BCDATE
    mongodump --uri="mongodb+srv://cluster0.1c3bjmr.mongodb.net/test" -u admin -p Pinguins1
    tar zcvf $BCDATE.tgz ./dump/
    cd /root/repoMongoDB/mongodb
    git pull
    mv /root/backup/mongo/$BCDATE/dump /root/backup/mongo/$BCDATE/test_$BCDATE
    mv /root/backup/mongo/$BCDATE/test_$BCDATE /root/repoMongoDB/mongodb/
    git add --all
    git commit -m "New Backup!"
    git push
elif [ $BCLASTWEEK = "53" ];
then
    mkdir /root/backup/mongo/$BCDATE
    cd /root/backup/mongo/$BCDATE
    mongodump --uri="mongodb+srv://cluster0.1c3bjmr.mongodb.net/test" -u admin -p Pinguins1
    tar zcvf $BCDATE.tgz ./dump/
    cd /root/repoMongoDB/mongodb
    git pull
    mv /root/backup/mongo/$BCDATE/dump /root/backup/mongo/$BCDATE/test_$BCDATE
    mv /root/backup/mongo/$BCDATE/test_$BCDATE /root/repoMongoDB/mongodb/
    git add --all
    git commit -m "New Backup!"
    git push
else
    echo "There are no backups for today!"
fi
```

O que não for explicado do script nesta “User Story” estará explicado na “User Story C4”.

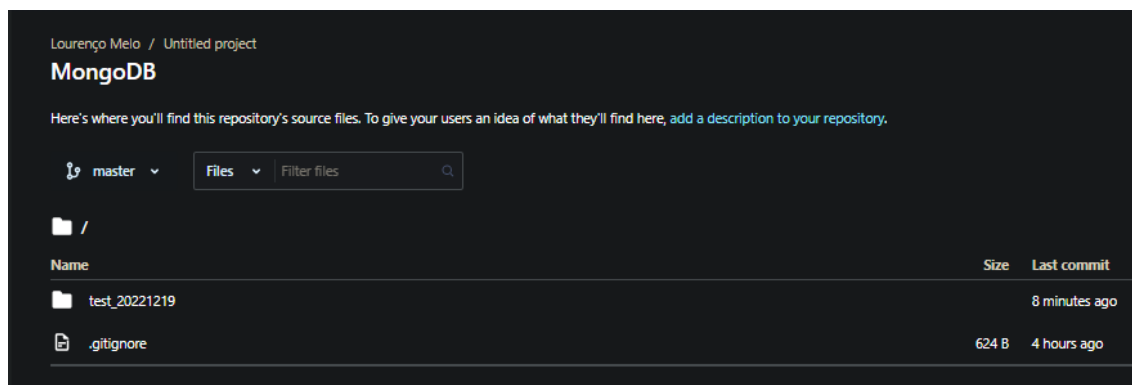
Como foi pedido que seja renomeado o ficheiro/diretório para algo com este formato “<nome\_da\_db>\_yyyymmdd”. Assim para conseguirmos obter a data naquele formato precisamos de criar uma variável chamada “BCDATE”. Como está a ser mostrado na figura acima, o formato foi alcançado com referindo primeiro “%Y” quer retorna o ano, após isso o “%m” que retorna o mês no mesmo formato pretendido e depois “%d” que retorna o dia. Ao definir a variável conseguimos satisfazer este requisito.

A base de dados tem como nome “test”.

A Cloud utilizada é o Bitbucket Cloud, com um repositório próprio para guardar todas as versões do “MongoDB”. Após descarregarmos a base de dados do “MongoDB”, alteramos o nome do diretório para o pedido e depois passamos para o nosso repositório git local para posteriormente dar “stage aos files” com o comando “git add --all”. Quando este comando acabar irá fazer um “commit” com uma mensagem “New Backup”. O argumento “-m” refere se a mensagem. Depois de termos o “commit” finalizado, iremos dar “push” para o repositório das alterações que temos, que neste caso é mais um diretório a mais, com uma versão mais atualizada.

Assim garantimos que todas as versões vão estar na “Cloud” e com o nome pedido.

### Demonstração:



Na “User Story C4” explica como este script corre todos os dias as “00:00”.

## User Story C4

Para conseguirmos realizar esta “User Story”, precisamos primeiro de instalar as ferramentas que nos permitam conectar à base de dados de “Mongo DB”. Para isso, instalamos o “MongoDB Server”, “MongoDB Shell” e “MongoDB Tools”.

Nesta máquina virtual precisamos de instalar uma versão mais antiga do “MongoDB Server”, sendo essa versão a “4.2.23”, devido às limitações físicas da máquina.

```
root@uvm038:~/mongodbshell# ls  
lei22_23_3dg_g38_spa      mongodb-org-server_4.2.23_amd64.deb  shell  
mongodb-database-tools-debian11-x86_64-100.6.1.deb  mongoexport  
mongodb-mongosh_1.6.1_amd64.deb                server
```

Assim fizemos estes comandos para instalar estas ferramentas:

```
root@uvm038:~/mongodbshell# dpkg -i mongodb-org-server_4.2.23_amd64.deb
```

```
root@uvm038:~/mongodbshell# dpkg -i mongodb-mongosh_1.6.1_amd64.deb
```

```
root@uvm038:~/mongodbshell# dpkg -i mongodb-database-tools-debian11-x86_64-100.6.1.deb
```

O dpkg gere os pacotes de distribuição e neste caso esta a instalar estes packages.

Figura 3 - Script "dbbackup.sh"

```
GNU nano 5.4                                dbbackup.sh
#!/bin/bash
BCDAY=$(date +%d)
BCLASTWEEK=$(date +%W)
BCMONTH=$(date +%m)
BCDAYWEEK=$(date +%u)
BCDAYYEAR=$(date +%j)
BCDATE=$(date +%Y%m%d)
if [ $BCDAYYEAR = "1" ];
then
    mkdir /root/backup/mongo/$BCDATE
    cd /root/backup/mongo/$BCDATE
    mongodump --uri="mongodb+srv://cluster0.1c3bjmr.mongodb.net/test" -u admin -p Pinguins1
    tar zcvf $BCDATE.tgz ./dump/
    cd /root/repoMongoDB/mongodb
    git pull
    mv /root/backup/mongo/$BCDATE/dump /root/backup/mongo/$BCDATE/test_$BCDATE
    mv /root/backup/mongo/$BCDATE/test_$BCDATE /root/repoMongoDB/mongodb/
    git add --all
    git commit -m "New Backup!"
    git push
elif [ $BCDAY = "1" ];
then
    mkdir /root/backup/mongo/$BCDATE
    cd /root/backup/mongo/$BCDATE
    mongodump --uri="mongodb+srv://cluster0.1c3bjmr.mongodb.net/test" -u admin -p Pinguins1
    tar zcvf $BCDATE.tgz ./dump/
    cd /root/repoMongoDB/mongodb
    git pull
    mv /root/backup/mongo/$BCDATE/dump /root/backup/mongo/$BCDATE/test_$BCDATE
    mv /root/backup/mongo/$BCDATE/test_$BCDATE /root/repoMongoDB/mongodb/
    git add --all
    git commit -m "New Backup!"
    git push
elif [ $BCMONTH = "12" ] && [ $BCDAYWEEK = "1" ];
then
    mkdir /root/backup/mongo/$BCDATE
    cd /root/backup/mongo/$BCDATE
    mongodump --uri="mongodb+srv://cluster0.1c3bjmr.mongodb.net/test" -u admin -p Pinguins1
    tar zcvf $BCDATE.tgz ./dump/
    cd /root/repoMongoDB/mongodb
    git pull
    mv /root/backup/mongo/$BCDATE/dump /root/backup/mongo/$BCDATE/test_$BCDATE
    mv /root/backup/mongo/$BCDATE/test_$BCDATE /root/repoMongoDB/mongodb/
    git add --all
    git commit -m "New Backup!"
    git push
elif [ $BCLASTWEEK = "53" ];
then
    mkdir /root/backup/mongo/$BCDATE
    cd /root/backup/mongo/$BCDATE
    mongodump --uri="mongodb+srv://cluster0.1c3bjmr.mongodb.net/test" -u admin -p Pinguins1
    tar zcvf $BCDATE.tgz ./dump/
    cd /root/repoMongoDB/mongodb
    git pull
    mv /root/backup/mongo/$BCDATE/dump /root/backup/mongo/$BCDATE/test_$BCDATE
    mv /root/backup/mongo/$BCDATE/test_$BCDATE /root/repoMongoDB/mongodb/
    git add --all
    git commit -m "New Backup!"
    git push
else
    echo "There are no backups for today!"
fi
```

No início do "scrip" estão definidas as variáveis que vão ser utilizadas, nas condições "if" ou para os nomes dos ficheiros.

Ao correr-mos o comando date no dia 18/12/2022 iremos obter este resultado:

dom 18 dez 2022 15:59:53 WET

Assim, para obtermos apenas as informações necessárias para satisfazer as condições “if” ou para definir os nomes utilizamos formatos que nos deem esses resultados.

Então vamos explicar estas variáveis:

- `$BCDAY`: irá retornar o dia do mês em que estamos. Em que o formato `%d` é o dia do mês.
- `$BCLASTWEEK`: irá retornar o número da semana em que estamos no ano. No dia 18/12/2022 por exemplo irá retornar 50.
- `$BCMONTH`: irá retornar o número do mês em que estamos no ano. Neste caso irá retornar 12
- `$BCDAYWEEK`: irá retornar o número da semana em que estamos, ou seja, segunda-feira representa o número 1 e o domingo o número 7.
- `$BCDAYYEAR`: irá retornar o dia do ano em que estamos, ou seja, como o ano tem 365 ou 366 irá retornar em que dia estamos nessa altura em concreto. Neste caso irá retornar 352.
- `$BCDATE`: irá retornar a data no formato `YYYYMMDD`. Neste caso será 20221218.

Após perceber o que representam estas variáveis, iremos agora explicar o script.

Para satisfazer que seja feita “1 Backup por mês no último ano” existem duas condições que o tornam isso possível. A primeira condição verifica caso seja o primeiro dia do ano. Se isso for verdade irá então dentro da pasta “mongo” criar uma pasta com a data do momento que será “01/01/2023”. E dentro dessa pasta com o comando “`cd /root/backup/mongo$BCYEAR/$BCDATE`”, irá descarregar a base de dados que está situada num cluster do mongodb. O comando “`mongodump --uri="mongodb+srv://cluster0.1c3bjmr.mongodb.net/test" -u admin -p xxxxxxxxx`” irá fazer um “dump” que irá criar um “binary export” dos contextos da base de dados. O argumento “uri” irá definir o uri para podermos aceder ao cluster. O “u” define o nome do utilizador e o “p” a password do utilizador, para podermos ter acesso bem-sucedido.

Desta vez, como está definido na “Figura 1” o que aparece a branco serve para esconder a password utilizada para aceder ao cluster por motivos de segurança.

Este comando irá criar um diretório chamado “/dump”. Para ocuparmos menos espaço com o possível aumento exponencial da utilização da base de dados decidimos comprimir este diretório com o comando “`tar zcvf $BCDATE.tgz ./dump/`”. O argumento “c” representa a criação, “v” para mostrar mais informação, ou seja, verboso ou palavroso, “f” para ficheiro e “z” para filtrar o arquivo através do gzip. Por não necessitarmos mais do diretório, naquele local movemos para o diretório para o repositório git com um nome diferente que posteriormente será útil.

A segunda condição que torna isso possível é (elif [ `$BCDAY = “1”` ] then), que verifica se estamos no primeiro dia do mês tornando assim possível o primeiro requisito. Após esta verificação são executados os comandos já previamente explicados.

Para satisfazer o segundo requisito, que é “1 backup por semana no último mês” fazemos a seguinte verificação (elif [ \$BCMONTH = “12” ] && [ \$BCDAYWEEK = “1” ] then) que irá verificar se estamos no mês de dezembro e se estamos no primeiro dia da semana, ou seja, se é segunda-feira, assim após executar os comandos previamente explicados somos capazes de satisfazer o segundo requisito.

Para satisfazer o terceiro requisito “1 backup por dia na última semana”, fazemos a seguinte verificação (elif [ \$BCLASTWEEK = “53” ] then), ou seja, ele verifica se estamos na última semana do ano que é a semana 53. Assim, em todos os dias da última semana ele irá fazer um backup da base dados.

Os comandos “if & elif & else” deram-nos a capacidade de resolver o problema abrangendo todas situações.

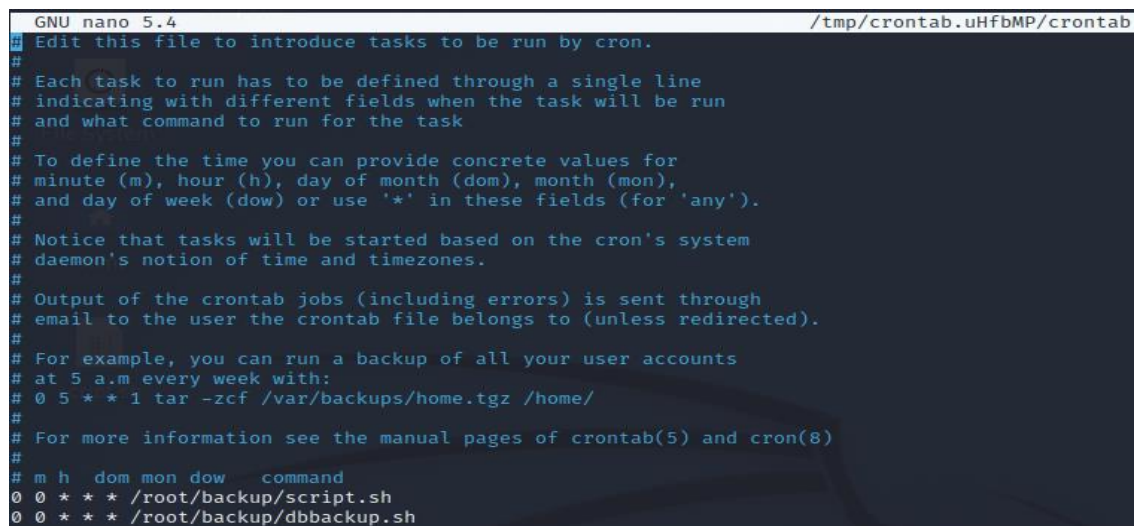
Caso seja necessário extrair o ficheiro por exemplo “12182022.tgz” teremos de executar o seguinte comando:

```
root@uvm038:~/backup# tar xvg 12182022.tgz
```

Em que “x” representa “extract”, “v” de verbose e “f” de file.

Então como foi referido anteriormente, precisamos de programar que a máquina de Linux para correr este script todos os dias às 00:00. Utilizamos o comando “crontab -e -u root”, em que “-e” representa edit e “-u” user. Pois inicialmente, apenas correndo o comando “crontab -e” é nos dito que o root não tem crontab.

*Figura2 - /tmp/crontab.uHfbMP/crontab*



```
GNU nano 5.4 /tmp/crontab.uHfbMP/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
0 0 * * * /root/backup/script.sh
0 0 * * * /root/backup/dbbackup.sh
```



## Demonstração do dia 19/12/2022:

Ao correremos o script no terminal ele fez o pretendido.

```
2022-12-19T17:15:24.671+0000    WARNING: On some systems, a password provided directly using --password
ord may be visible to system status programs such as 'ps' that may be invoked by other users. Consider
er omitting the password to provide it via stdin, or using the --config option to specify a configuration
ation file with the password.
2022-12-19T17:15:25.987+0000    writing test.trucks to dump/test/trucks.bson
2022-12-19T17:15:26.036+0000    writing test.users to dump/test/users.bson
2022-12-19T17:15:26.083+0000    writing test.roles to dump/test/roles.bson
2022-12-19T17:15:26.137+0000    writing test.paths to dump/test/paths.bson
2022-12-19T17:15:26.244+0000    done dumping test.trucks (3 documents)
2022-12-19T17:15:26.281+0000    done dumping test.users (0 documents)
2022-12-19T17:15:26.287+0000    done dumping test.roles (0 documents)
2022-12-19T17:15:26.289+0000    done dumping test.paths (28 documents)
./dump/
./dump/test/
./dump/test/trucks.metadata.json
./dump/test/paths.bson
./dump/test/trucks.bson
./dump/test/users.metadata.json
./dump/test/roles.metadata.json
./dump/test/users.bson
./dump/test/roles.bson
./dump/test/paths.metadata.json
root@uvm038:~/backup# ls
dbbackup  mongo2022  repos.tgz  script.sh
root@uvm038:~/backup# cd mongo2022/
root@uvm038:~/backup/mongo2022# ls
19122022
```

Aqui podemos ver o resultado do script no dia 19. De acordo com as condições ele deve fazer o backup por ser uma segunda-feira do último mês do ano, como foi programado.

```
root@uvm038:~/backup/mongo2022/19122022# ls
19122022.tgz
```

E dentro da pasta tem o ficheiro comprimido com a base de dados.

Importante referir que esta demonstração, foi realizada antes de melhorias e afinamentos no próprio script.

## User Story C5

Nesta us era pedido que o processo da US C3 fosse mantido no log do Linux e que, em caso de ocorrer um erro grave, fosse alertado o administrador durante o acesso à consola.

Numa 1ª parte foi editado o ficheiro de configuração do rsyslog de modo a escolher o diretório onde vão ser colocados os logs. Uma vez que o script de backup criado na US C3 é executado através do serviço Cron de forma agendada, o log do processo irá ser redirecionado para o diretório /var/log/cron.log:

```
#####
#### RULES ####
#####

#
# First some standard log files.  Log by facility.
#
auth,authpriv.*          /var/log/auth.log
*.*;auth,authpriv.none   -/var/log/syslog
cron.*                   /var/log/cron.log
daemon.*                 -/var/log/daemon.log
kern.*                   -/var/log/kern.log
lpr.*                    -/var/log/lpr.log
mail.*                   -/var/log/mail.log
user.*                   -/var/log/user.log
```

Se durante a execução do script no serviço cron ocorrer um erro grave, a mensagem de log irá ser guardada no diretório /var/log/cron.err:

```
#
# Split when cron script throws error
cron.err                  /var/log/cron.err
```

Após o processo estar a ser mantido no log do Linux, é necessário alertar o administrador no caso de ocorrer um erro grave durante o backup. Para isso foi criado um pequeno script.

```
#Check erros on crontab
CRONERR=$(wc -l /var/log/cron.err)
if [ $CRONERR != "0 cron.err" ];
then
echo "Errors found on cron.err. Backup with errors!"
else
echo "Backup without errors!"
fi
echo "-----"
```

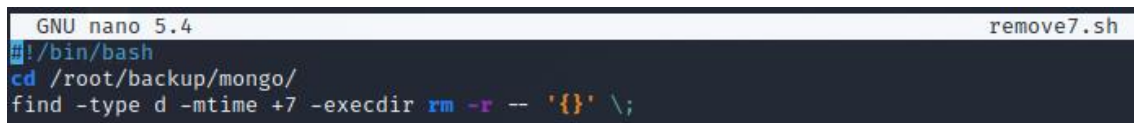


Como podemos ver na figura acima, é executado o comando `wc -l` que conta as novas linhas no diretório especificado à frente (neste caso o ficheiro `cron.err`). Caso tenha ocorrido algum erro grave, um log terá sido guardado neste ficheiro e, por isso, o `wc -l` terá de retornar um valor superior a 0. No entanto, caso o “`wc -l n`” encontre novas linhas no ficheiro, irá informar também o administrador que o backup correu com normalidade.

## User Story C6

Para conseguir resolver esta “User Story” foi necessário criar um script com a capacidade de remover todos os ficheiros de backup da base dados que existem há mais de 7 dias.

Assim, iremos explicar pormenorizadamente o comando que permite que isso aconteça.



```
GNU nano 5.4 remove7.sh
#!/bin/bash
cd /root/backup/mongo/
find -type d -mtime +7 -execdir rm -r -- '{}' \;
```

O comando “`find`” serve para encontrar ficheiros, diretórios e links. Após esse comando descrevemos o caminho para o diretório onde queremos encontrar os ficheiros com mais de 7 dias. Depois o “`-type d`” descreve que os ficheiros que procuramos são diretórios, o “`-mtime +7`” verifica se os diretórios tem mais que 7 dias de criação. Para cada resultado então desse comando o “`-execdir ... \;`” vai executar o comando que estará à sua frente que neste caso é o `remove`. A “flag” “`-r`” remove recursivamente os diretórios e ficheiros que estão dentro do diretório que pretendemos remover. Assim o “`{}`” é onde o resultado do `find` vai ser substituído e o “`—`” significa que acaba os parâmetros do comando evitando assim os erros dos ficheiros que comecem com hífen.

Depois de testar este script adicionamos a linha de código ao script “`dbbackup.sh`” dentro das condições “`if`” para só eliminar quando acabar de criar uma versão de backup. Assim garantimos que temos sempre pelo menos um backup da base de dados atualizados e posteriormente os backups com mais de 7 dias são eliminados, na máquina virtual do DEI. Já na Cloud iremos ter todas as versões.

```

elif [ $BCLASTWEEK = "53" ];
then
    mkdir /root/backup/mongo/$BCDATE
    cd /root/backup/mongo/$BCDATE
    mongodump --uri="mongodb+srv://cluster0.1c3bjmr.mongodb.net/test" -u admin -p Pinguins1
    tar zcvf $BCDATE.tgz ./dump/
    cd /root/repoMongoDB/mongodb
    git pull
    mv /root/backup/mongo/$BCDATE/dump /root/backup/mongo/$BCDATE/test_$BCDATE
    mv /root/backup/mongo/$BCDATE/test_$BCDATE /root/repoMongoDB/mongodb/
    git add --all
    git commit -m "New Backup!"
    git push
    cd /root/backup/mongo/
    find -type d -mtime +7 -execdir rm -r -- '{}' \;
else
    echo "There are no backups for today!"
fi

```

## User Story C7

### 1. Visão geral

Esta “*Business Impact Analysis*” (BIA) é desenvolvida como parte do processo de planeamento de contingência. Foi concluído a 08 de janeiro de 2023.

#### 1.1 Propósito

O objetivo do BIA é identificar e priorizar componentes do sistema, correlacionando-os com o processo da missão/negócio(os) que o sistema suporta, e utiliza esta informação para caracterizar o impacto no processo(os) se o sistema não estiver disponível.

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.

## 2.Descrição do sistema

A missão da nossa organização é gerir a **distribuição, da maneira mais rentável e fiável possível, de entregas entre armazéns** utilizando uma frota de camiões elétricos da empresa “*EletricAcme, S.A*”.

Os **serviços essenciais** da nossa aplicação são: a criação das melhores rotas de entregas para a empresa e a visualização de todos os dados acerca das entregas, rotas, camiões e armazéns. Nas instalações da empresa encontram-se os servidores e a base de dados do sistema.

São feitos backups noutra plataforma para a salvaguarda dos dados gerais e existe um gerador de energia que consegue alimentar os servidores e a base de dados durante um certo espaço de tempo.

## 3.Recolha de Dados BIA

### 3.1Determinar a criticidade do processo e sistema

**Primeiro passo do processo BIA** - Trabalhar com gestores, chefes de equipa e outros pontos de contacto internos ou externos, identificar os processos específicos de missão/negócio que suportam o sistema.

Missão/Processo de Negócios	Descrição
Base de dados	Guarda os dados necessários para o funcionamento do sistema
Servidores	Processam e executam solicitações feitas por usuários através de softwares
Serviço de login	Autenticação e validação de utilizadores. Criação de ambientes específicos para cada utilizador com as suas devidas permissões
Serviço de planeamento de rotas	Criação da rota mais eficiente de um dado camião para um certo dia
Serviço de consulta de dados	Possibilita a visualização de dados acerca de camiões, armazéns, entregas, planos de viagem e rotas

### 3.1.1 Identificar impactos de interrupções e downtime estimado

#### Impactos da Interrupção

As seguintes categorias de impacto representam áreas importantes a ter em conta em caso de perturbação.

#### Categorias de impacto:

- Segurança e integridade dos funcionários;
- Reputação e imagem da marca;
- Serviço ao cliente;
- Recursos materiais e de mão de obra;

#### Valores de impacto:

- Crítico;
- Alto;
- Médio;
- Baixo;
- Nula;

O quadro abaixo resume o impacto em cada missão/processo de negócio se este falhasse:

Missão/Processo de Negócios	Categoria de Impacto				
	Segurança	Imagem	Cliente	Recursos	Impacto
Base de dados	Baixo	Médio	Crítico	Alto	*Alto+
Servidores	Baixo	Medio	Crítico	Crítico	Crítico
Serviço de login	Nulo	Baixo	Medio	Baixo	Baixo
Serviço de planeamento de rotas	Nulo	Alto	Alto	Médio	Alto
Serviço de consulta de dados	Nulo	Baixo	Médio	Baixo	Baixo

\*note-se que o impacto de uma falha na base de dados apenas tem um impacto “Alto+” devido á existência de dados de backup. Caso este backup fosse afetado o impacto passaria a ser “Crítico”.

### Downtime estimado

Trabalhando diretamente com os proprietários de processos de missão/negócio, pessoal do departamento, gestores e outras partes interessadas, foram estimados os downtimes resultantes de um evento disruptivo.

- **Maximum Tolerable Downtime (MTD)** - O MTD representa a quantidade total de tempo que os líderes/gestores estão dispostos a aceitar para uma interrupção de um processo de missão/negócio e toma em consideração o impacto.
- **Recovery Time Objective (RTO)** - O RTO define o período máximo que um recurso do sistema pode permanecer indisponível antes de haver um impacto inaceitável em outros recursos do sistema, processos de missão/negócio e no MTD.
- **Recovery Point Objective (RPO)** - O RPO representa o ponto no tempo, antes de uma interrupção do sistema, para o qual os dados de missão/processo de negócio devem ser recuperados (utilizando os dados de backup mais recentes).

O quadro abaixo identifica o MTD, RTO e POR:

Missão/Processo de Negócios	MTD	RTO	RPO
Base de dados	12 horas	2 horas	4 horas
Servidores	12 horas	5 horas	4 horas
Serviço de login	24 horas	1/2 horas	24 horas
Serviço de planeamento de rotas	12 horas	1/2 horas	12 horas
Serviço de consulta de dados	24 horas	1/2 horas	24 horas

### 3.2 Identificar os requisitos de recursos

A tabela seguinte identifica recursos incluindo hardware, software e outros recursos:

Recurso/Componente do sistema	Plataforma/OS/Versão (conforme aplicável)
<i>Servidor Web 1</i>	Server Dell R740XD 24SFF
<i>Servidor Web 2</i>	Server Dell R740XD 24SFF
<i>Servidor Web 3</i>	Server Dell R740XD 24SFF
<i>Router Principal</i>	Cisco C9500-48X-E
<i>Router 2</i>	Cisco C1111-8PLTEEAW
<i>Router 3</i>	Cisco C1111-8PLTEEAW

### 3.3 Identificar prioridades de recuperação dos recursos do sistema

Prioridade	Objetivo do tempo de recuperação
<b>Servidor Web 1</b>	<i>4 horas</i>
<b>Servidor Web 2</b>	<i>4 horas</i>
<b>Servidor Web 3</b>	<i>6 horas</i>
<b>Router Principal</b>	<i>1 hora</i>
<b>Router 2</b>	<i>30 minutos</i>
<b>Router 3</b>	<i>30 minutos</i>

## User Story C10

Para concluir esta “User Story” foi necessário criar um par de chaves (privada e publica). E associar essas keys ao login “[root@10.9.10.38](mailto:root@10.9.10.38)”. Para isso foi necessário permitir o acesso à root pelo ssh. Então no ficheiro “/etc/ssh/sshd\_config” tivemos de dar permissão ao login à root pelo ssh definindo “PermitRootLogin yes #enabled”.

```
GNU nano 5.4 /etc/ssh/sshd_config
$OpenBSD: sshd_config,v 1.103 2018/04/09 20:41:22 tj Exp $

# This is the sshd server system-wide configuration file. See
# sshd_config(5) for more information.

# This sshd was compiled with PATH=/usr/bin:/bin:/usr/sbin:/sbin

# 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
PermitRootLogin yes #enabled
```

Depois disso tivemos de gerar as keys com o comando “ssh-keygen -t rsa” na máquina do administrador que irá se ligar à máquina virtual do DEI. Assim no diretório “~/ssh” terá os ficheiros “id\_rsa” e “id\_rsa.pub”.

```
([REDACTED]@kali)-[~]
$ ls ~/.ssh
authorized_keys  id_rsa  id_rsa.pub  known_hosts  known_hosts.old
```

Aqui está a key publica.

```
([REDACTED]@kali)-[~]
$ cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQgQC0Tigsm0QP0yJAUZmNyKAyZ+2y7VgpH5i8zMX4
8zsbNpEJEfM/PchfPToya7nBPLaYf1zMjg8c5lYawavfN07ZRyHKhNlh/6snCrptBa6pQnHfJ4jY
MNcTCG8sNctBdUa5HQKQfw/FrU3JOwhvWDYBZmHMPZETgivpdVTUuiKIzYdSUSVF0wttZQzBfo9m
mY4TSC4oGfMfPzTsV2ev49WvJjgGxOm2K7t0qt78Xeqj5uCNBKp86tAV2bB5favT3MjF5zXyeUDm
Ye2PaPZwJUBPHDiPrRnTI9y9GrnBG0DtBZIWfe7dk0cwOXglAdBTgHwE49nXpyW1THGS0vVt0s9x
uwaYb5my6g0E7GNVrnPs+ZGMU4umwuFYMu5//4bHMHvuMrU2Skhoi+JIjJXhY09eb0c03dnnU4Ln
bk0QZH0BcZzabt2kV/ypJfQ9kC+94aGUicMlDlr8oNUU8Hzt+XSxfarrigH6MDE0JNHuxs+gmcko
nsqdweE52iv/BjcnbfE=[REDACTED]@kali
```

Em vez de copiar o conteúdo desta key e colar no ficheiro “authorized\_keys” da máquina do DEI. Utilizamos um comando que torna isto possível de forma eficiente. Basicamente irá referenciar a nossa key publica ao login de ssh que iremos utilizar e adicionar no ficheiro “authorized\_keys” o conteúdo dessa key à máquina virtual do DEI.



```

kali)-[~]
$ ssh-copy-id -i ~/.ssh/id_rsa.pub root@10.9.10.38 1 ✖
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/lolas/.ssh/id_rsa.pub"
/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

=====
DATE:
01/11/2022
ACTIVE USERS:
1
=====
root@10.9.10.38's password:

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'root@10.9.10.38'"
and check to make sure that only the key(s) you wanted were added.

```

Aqui fazemos uma demonstração de como é possível ligar por SSH sem introduzir a password:

```

kali)-[~]
$ ssh root@10.9.10.38

=====
DATE:
01/11/2022
ACTIVE USERS:
1
=====
Linux uvm038 5.10.0-17-amd64 #1 SMP Debian 5.10.136-1 (2022-08-13) 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: Sat Jan  7 17:09:45 2023 from 10.8.61.70

=====
WELCOME TO THE MACHINE 038
DATE INFO:
sáb 07 jan 2023 17:20:32 WET
MY USER:
root
wc: /var/log/cron.err: Ficheiro ou pasta inexistente
-bash: [: ≠: operador unário esperado
Backup without errors!

=====
root@uvm038:~# █

```