Uma imagem com texto, Tipo de letra, logótipo, Gráficos

Descrição gerada automaticamenteUma imagem com Tipo de letra, texto, Gráficos, logótipo

Descrição gerada automaticamente

**ASIST**

**SPRINT 3**

**Realizado por:**

**Diogo Ferreira, 1220829**

**Francisco Osório, 1220846**

**Sérgio Moreira, 1220890**

**Rafael Ferraz, 1221104**

**01/2025**

**ÍNDICE**

[Divisão Tarefas 3](#_Toc183523809)

[User Stories 4](#_Toc183523810)

[User Story 1 4](#_Toc183523811)

[User Story 2 6](#_Toc183523812)

[User Story 3 7](#_Toc183523813)

[User Story 4 8](#_Toc183523814)

[User Story 5 9](#_Toc183523815)

[User Story 6 10](#_Toc183523816)

[User Story 7 11](#_Toc183523817)

[User Story 8 12](#_Toc183523818)

[User Story 9 13](#_Toc183523819)

[User Story 10 14](#_Toc183523820)

[User Story 11 15](#_Toc183523821)

[User Story 12 16](#_Toc183523822)

# Divisão Tarefas

Tabela 1- Divisão das Tarefas

|  |  |
| --- | --- |
| User Stories | Aluno |
| 1 | Francisco Osório |
| 2 | Sérgio Moreira |
| 3 | Rafael Ferraz |
| 4 | Sérgio Moreira |
| 5 | Diogo Ferreira |
| 6 | Rafael Ferraz |
| 7 | Sérgio Moreira |
| 8 | Francisco Osório |
| 9 | Francisco Osório |
| 10 | Sérgio Moreira |
| 11 | Diogo Ferreira |
| 12 | Rafael Ferraz |

# User Stories

## User Story 1

Nesta *user story* pretendia-se que desenvolvêssemos um plano de recuperação de desastre que satisfaça o *MBCO* definido no *sprint B*.

O *Disaster Recovery Plan (DRP)* é um documento detalhado que descreve as estratégias e procedimentos de como a organização deve responder a um incidente não planeado, de forma eficaz, e retomar as suas operações diárias.

Desta forma, para satisfazer o *MBCO* que foi definido no *Sprint B*, elaborou-se o seguinte *Disaster Recovery Plan* para a nossa organização.

O plano visa cobrir a recuperação dos módulos que foram definidos no *MBCO*: o *backend,* o  *frontend ,* o *3D visualization* eo *planning.* Sendo assim, este plano tem como objetivo:

* Assegurar que as funcionalidades mínimas sejam mantidas (*MBCO*) durante uma disrupção.
* Recuperar completamente as funcionalidades de todos os módulos dentro do *MTPD* definido.
* Minimizar impactos para a organização e seus clientes.

Para isso temos uma equipa de resposta, que é essencial para garantir rapidez na resolução de problemas e coordenação.

Tabela 2 - Equipa de Resposta

|  |  |  |
| --- | --- | --- |
| Nome | Responsabilidade | Contacto |
| Francisco Osório | Coordenação do *DRP* e comunicação externa e interna | [1220846@isep.ipp.pt](mailto:1220846@isep.ipp.pt) |
| Sérgio Moreira | Recuperação dos módulos | [1220890@isep.ipp.pt](mailto:1220890@isep.ipp.pt) |
| Diogo Ferreira | Recuperação dos módulos | [1220829@isep.ipp.pt](mailto:1220829@isep.ipp.pt) |
| Rafael Ferraz | Gestão dos backups | [12221104@isep.ipp.pt](mailto:12221104@isep.ipp.pt) |

Para garantir a recuperação rápida e segura dos módulos, foi definida uma estratégia robusta de backups que cobre os seguintes aspetos:

* Backups incrementais diariamente;
* Backups completos ao domingo;

Desta forma, com esta estratégia de backups, é possível oferecer um equilíbrio entre eficácia e eficiência.

Os procedimentos para recuperação foram desenvolvidos com foco em garantir que os módulos sejam restaurados dentro do *MTPD*. A sequência de recuperação é estabelecida para priorizar os módulos mais críticos, assim a prioridade será dada ao *backend*, seguido do *frontend*, posteriormente o *planning* e por fim o *3D* *Visualization*. Desta forma, temos os seguintes procedimentos:

* Identificação do escopo e do impacto do incidente;
* Ativação da equipa de resposta;
* Recuperação dos dados, através dos backups recentes;
* Recuperação dos módulos;
* Verificação da operacionalidade dos módulos;
* Elaboração de um relatório detalhado para documentar o incidente e as ações tomadas;

É essencial garantir a eficácia do plano, e não apenas desenvolver, desta forma serão realizados testes periódicos, simulando alguns cenários de disrupção que podem acontecer. Sendo assim, é possível identificar falhas no plano desenvolvido, preparar a equipa de resposta e validar a capacidade da recuperação dos módulos.

Concluindo, este plano será revisado frequentemente e estará sujeito a alterações sempre que necessário, para garantir a sua eficácia frente a mudanças na infraestrutura ou na organização.

## User Story 2

## User Story 3

## User Story 4

Nesta US é pedido que os backups realizados pela US 3 sejam guardados 1 por dia da última semana, 1 por semana do último mês, 1 por mês do último ano. Primeiro, precisamos de escalonar o script da US 3 para correr todos os dias, isso tratará dos backups diários, não nos precisamos de preocupar em apaga-los pois a US 6 já o faz. Para tal, precisamos de editar o *cron* (**crontab -e**):

Uma imagem com texto, Tipo de letra, captura de ecrã, Gráficos

Descrição gerada automaticamente

Figura - crontab US4 parte 1

Em seguida, precisamos de mover os backups diários para um outro diretório para evitar que sejam apagados pela US 6:

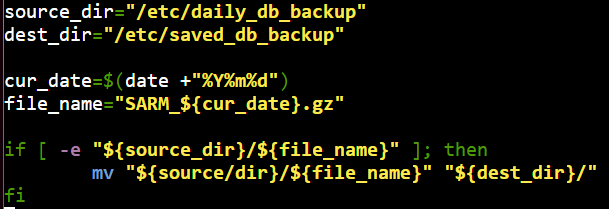


Figura 2 - db\_backup\_save.sh

Depois, precisamos de escalonar o *script* para correr semanalmente (por exemplo, todas as terças de madrugada) e mensalmente (por exemplo, todos os dias 2, de madrugada), para isso, adicionamos o seguinte ao *cron* (**crontab -e**):

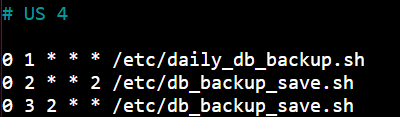


Figura 3 – crontab US4 parte 2

Como mencionado antes, a US 6 apaga os ficheiros diários com tempo de vida superior a 1 semana, mas também temos de apagar os semanais e mensais que perdem relevância:

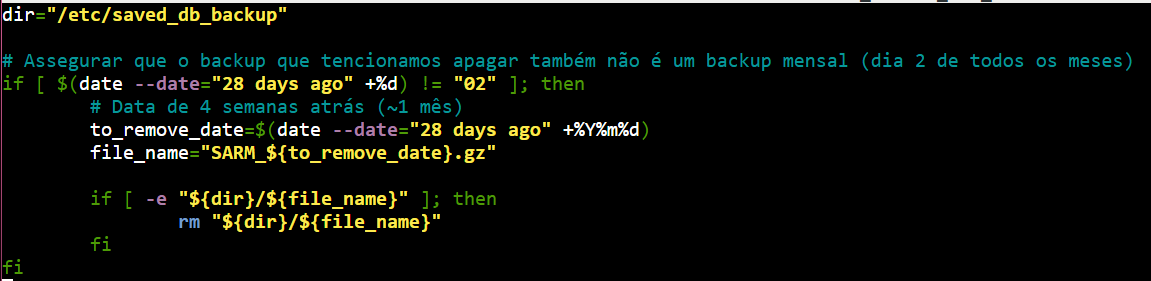


Figura 4 - db\_remove\_old\_weeklies.sh

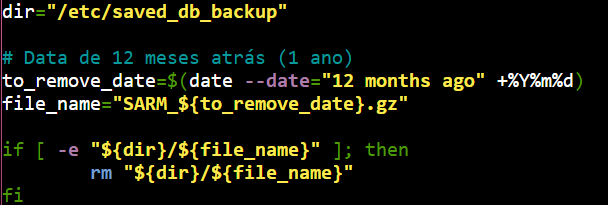


Figura 5 - db\_remove\_old\_monthlies.sh

Estes dois scripts apagam backups feitos 1 mês e 1 ano atrás, respetivamente, não esquecer que pode acontecer que haja um backup semanal exatamente 4 semanas após um backup mensal, nesse caso não o podemos apagar). Só precisamos agora de correr os dois scripts em simultâneo com os outros, fazendo as alterações finais ao *cron* (os de remover irão correr 5 minutos depois da hora):

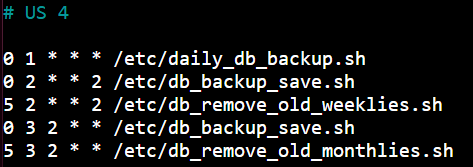


Figura 6 – crontab US4 final

## User Story 5

## User Story 6

Para esta US pede-se que os backups diários tenham um tempo de vida de 7 dia, exceto os que foram convertidos para semanais/mensais pela US 4. Para tal, iremos criar um script que apaga os backups realizados 7 dias atrás:

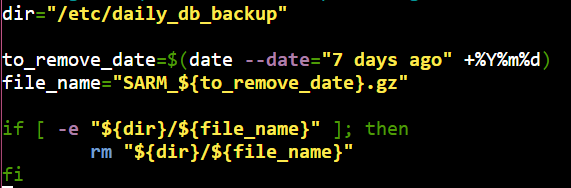


Figura 7 - db\_remove\_old\_dailies.sh

Por fim, temos de marcar o script para ser executado diariamente, este irá correr às 5 da manhã. Para fazer tal, temos de editar o *cron*, para tal, usamos o comando “***crontab -e***”, e adicionamos a seguinte linha:

Uma imagem com texto, Tipo de letra, captura de ecrã, Gráficos

Descrição gerada automaticamente

Figura 8 - crontab

## User Story 7

## User Story 8

Nesta *user story* foi solicitado que implementássemos uma gestão de acessos que satisfaça os critérios apropriados de segurança.

Desta forma, para termos uma gestão de acessos robusta e segura aplicaram-se as seguintes práticas:

* **Controlo de Acesso Baseado em *Roles* (*RBAC*)**: Foram definidas *roles* (*Admin, Patient, Doctor, Nurse, Technician*), atribuindo a cada uma, um conjunto de permissões que determina o acesso aos recursos ou funcionalidades. Desta forma, quando um utilizador dá login na aplicação, o *backend* retornará o *token* e a sua role, sendo estes armazenados no *localstorage* do *browser*. Sempre que o utilizador faz um pedido ao *backend*, este *token* será enviado sempre no *header* dos *requests*, sendo verificado assim no *backend* se o utilizador tem permissões para tal funcionalidade, através de políticas de autorização. Já no *frontend*, cada *role* tem a sua *dashboard* personalizada e para além disso, todas as rotas são protegidas por um *Guard* (neste caso *Role* *Guard*), garantindo assim maior segurança no acesso às funcionalidades do sistema.
* **Armazenamento das Passwords:** Na nossa aplicação estamos a usar um provedor externo, para facilitar a parte da autenticação e autorização, o *Auth0*. Desta forma, o *Auth0* armazena as passwords dos utilizadores de forma segura, utilizando práticas de segurança como o *hashing*,
* **Política de Password**: Foi definida uma política de password robusta que seguem os seguintes critérios:
  + Conter pelo menos 8 caracteres;
  + Incluir uma letra maiúscula, um número e um caractere especial
* **Bloqueio após exceder as tentativas de *Login*:** Para proteger as contas dos utilizadores contra ataques *Brute-Force* e acessos não autorizados, caso um utilizador falhe o login após 5 tentativas a conta do mesmo será bloqueado e enviada uma notificação por email.
* **Expiração do *Token*:** O *token* emitido pelo *backend* após o login foi definido para ter uma expiração de 3600 segundos, ou seja, 1 hora, independentemente da atividade do utilizador.
* ***Single Sign-On* (*SSO*)**: Para simplificar a experiência do utilizador e aumentar a segurança, utilizou-se um *SSO* que já vem definido por *default* no *Auth0*, sendo assim possível aceder a várias aplicações da organização uma única credencial de login, não havendo credenciais duplicadas.
* ***Logs* de Auditoria:** Foi implementado um sistema de *logs* de auditoria, também fornecido já pelo *Auth0*, que regista todas as atividades relevantes de autenticação e autorização. Esses *logs* incluem eventos como tentativas de login (bem-sucedidas ou não), alterações nas credenciais dos utilizadores, mudanças nas permissões de acesso e ações administrativas (como a criação, atualização e deleção de utilizadores e roles).

Concluindo, através das práticas anteriormente descritas, é possível estabelecer uma gestão de acessos segura, desde a proteção das contas dos utilizadores contra os ataques e acessos não autorizados, mas também oferecer uma experiência confiável e eficiente para todos os utilizadores da aplicação.

## User Story 9

Nesta *user story* foi pedido implementado de forma justificada um sistema de *clustering* entre os sistemas que implementam o *SPA*. Desta forma, realizou-se o *deploy* do *frontend* da aplicação em dois servidores do DEI (*vs215* e *vs540*), da mesma forma que no sprint anterior, adaptando apenas o script.

Criou-se, posteriormente outra máquina no servidor do DEI (*vs414*), que será utilizado como *Load Balancer*. Desta forma, instalou-se HAProxy através dos comandos “*apt update*” e “*apt install haproxy*”, respetivamente.

De seguida, passou-se para a configuração do *HAProxy*, editando-se o ficheiro “*haproxy.cfg*” com o comando “*nano /etc/haproxy/haproxy.cfg*”, adicionando as configurações que podemos ver de seguida :

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Figura 9 - Configuração do ficheiro haproxy.cfg

Explicando agora as configurações anteriores, inicialmente na secção do *frontend* designada “*http\_front*”, configura-se o *HAProxy* para ouvir as requisições *HTTP* na interface “*10.9.21.158*” e na porta “*2224*” (porta para aceder do exterior á *VM*). De seguida, ativou-se a página de estatísticas, que será utilizado para monitorar o estado do balanceamento de carga em tempo real, e definiu-se que estaria disponível pela *URL* *“/haproxy?stats”*. Para além disso, definiu-se as configurações “*option httpclose*” e “*option forwardfor*”, para garantir o encerramento adequado das conexões *HTTP* e a propagação do endereço *IP* do cliente original para os servidores *backend*, respetivamente.

De seguida, as requisições recebidas no *frontend* são encaminhadas para o *backend*, identificado como “*http\_back*”. Nesta secção, definiu-se que o *HAProxy* iria utilizar o algoritmo de balanceamento *round-robin*, distribuindo assim as requisições de forma equitativa entre os servidores configurados, neste caso, “*server1*” (*vs215*) com o *IP* “*10.9.20.215*” e “*server2”* (*vs540*) com o *IP* “*10.9.22.28*”, ambos na porta “*2224*”. Para cada servidor configurou-se a opção “*check*”, que ativa verificações periódicas, para garantir que o servidor está disponível. Caso um dos servidores falhe ou fique indisponível, o *HAProxy* remove-o temporariamente e redireciona as requisições para o servidor que permanece funcional.

Após a configuração do *HAProxy,* executou-se os seguintes comandos:

Uma imagem com texto, Tipo de letra, captura de ecrã, file

Descrição gerada automaticamente

Figura 10 - Validação do ficheiro de configuração e reinício do serviço HAProxy

O primeiro comando utilizou-se para garantir que a nova configuração definida não possui erros, sendo que o segundo aplica as alterações, reiniciando o serviço.

Para testar a implementação o sistema de *clustering*, recorreu-se á a interface de estatísticas do *HAProxy*, que se definiu na configuração. De seguida, num terminal, através do comando *“while true; do curl 10.9.21.158:2224; sleep 1; done”*, realizou-se várias requisições para e podemos ver o resultado do balanceamento da carga de seguida:

Uma imagem com texto, captura de ecrã, software, Software de multimédia

Descrição gerada automaticamente

Figura 11 - Teste do sistema de clustering e balanceamento de carga

Pela imagem anterior, é possível verificar que a carga foi distribuída equitativamente pelos servidores (*server1* e *server2*).

Assim, através desta implementação de *clustering*, garante-se alta disponibilidade, balanceamento de carga e uma experiência confiável para os utilizadores. Em caso de falha de um servidor, o tráfego é redirecionado automaticamente para outros servidores funcionais, garantindo a continuidade da aplicação. Através do balanceamento *round-robin* distribui-se as requisições de forma equilibrada, evitando sobrecarga. Para além disso, a solução é escalável, permitindo adicionar novos servidores caso haja essa necessidade

## User Story 10

## User Story 11

## User Story 12

Nesta US é pedido que se confirme que os backups estão a ser feitos corretamente, incluindo uma validação. Para isto desenvolveu-se o script *db\_restore.sh*, que faz o descrito acima:

Uma imagem com texto, captura de ecrã, software, Software de multimédia

Descrição gerada automaticamente

Figura - db\_restore.sh parte 1

Uma imagem com texto, captura de ecrã, software

Descrição gerada automaticamente

Figura - db\_restore.sh parte 2

Primeiro, é executada a função *restoreBackup()*, que restaura o backup fornecido por argumento na base de dados, se tal suceder, corre-se a próxima função. A segunda função do script, *validateCollections* faz uma *query* à base de dados para verificar a existência das coleções. Finalmente, o script limpa a pasta temporária que criou durante a sua execução.

Podemos ver o resultado do script em execução: Uma imagem com texto, captura de ecrã, menu, Tipo de letra

Descrição gerada automaticamente

Figura - Execução db\_restore parte 1

Uma imagem com captura de ecrã, texto, preto e branco, monocromático

Descrição gerada automaticamente

Figura - Execução db\_restore parte 2

Acima podemos ver a função de restauração em execução.

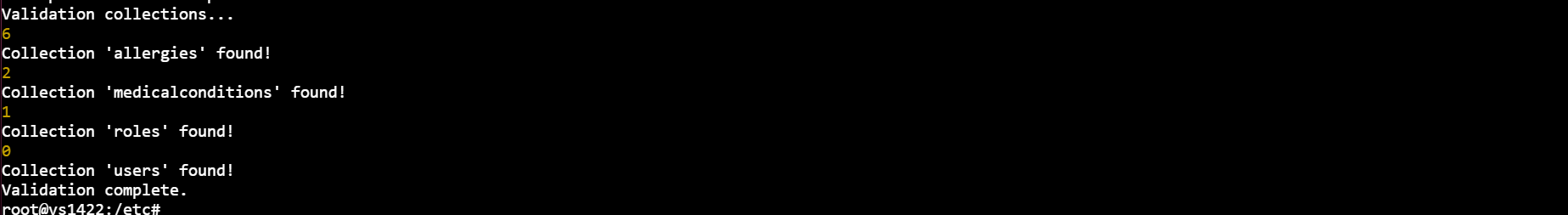


Figura - Execução db\_restore parte 3

Finalmente, o script verifica as coleções com uma *query*.