Macintosh HD:Users:alexandre:Desktop:LOGOS_DEPARTAMENTOS_PAPEL_DE_CARTA:CIENCIAS_ULISBOA_INFORMATICA_CMYK.eps



**Relatório de Situação do Projeto**(4ª Reunião de Avaliação)

# Grupo 05

|  |  |  |
| --- | --- | --- |
|  | Nome | Número |
| 1 | Nome | Número |
| 2 | André Silva | 58644 |
| 3 | Bernardo Caldas | 58650 |
| 4 | David Carreira | 58629 |
| 5 | Martim Emauz | 58668 |
| 6 | Miguel Martins | 58661 |

# Requisitos não-funcionais

**RNF-01 – Fonte aberta:** O sistema deve ser de código fonte aberto.

Estado: Concluído

Percentagem concluída: 75%

Aspetos implementados:

* O repositório do GitHub (https://github.com/MartimEmauz/PTI-PTR) é **público** e contém o código necessário para a implementação do projeto, bem como scripts e outras ajudas;
* O software e ferramentas usadas para o desenvolvimento do sistema são públicos.

Aspetos em falta:

* Que a API seja acessível por um utilizador externo ao sistema, independentemente do website.

**RNF-02 – Ambiente de desenvolvimento:** O ambiente de desenvolvimento deve fazer uso de ferramentas de controlo de versões que suportem uma equipa de programadores. Adicionalmente devem ser usados mecanismos de automação que facilitem o desenvolvimento da aplicação.

Estado: Concluído

Percentagem concluída: 100%

Aspetos implementados:

* É utilizado um repositório partilhado com ***branches*** para cada desenvolvedor;
* Foram criados **guiões** para a instalação do software necessário para o desenvolvimento do projeto.
* Foi elaborado e atualizado um ***Backlog*** com as tarefas a realizar por ordem de prioridade

**RNF-03 – Lançamento para produção****:** O sistema deve ter um automatismo que permite lançá-lo em produção a partir de um repositório de código.

Estado: Não desenvolvido

**RNF-04 – Interoperabilidade:** O sistema deve ter uma API (REST, por exemplo) que lhe permite interagir com outras aplicações.

Estado: Em desenvolvimento

Percentagem concluída: 75%

Aspetos implementados:

* A API está feita de forma modular e dinâmica, permitindo vários tipos de ações (ver secção 2.2);
* Segue a arquitetura **REST**.

Aspetos em falta:

* Abertura ao público, ou seja, ser acessível através de um IP público através do qual se podem fazer pedidos HTTPS.

**RNF-05 –** **Escalabilidade:** O sistema deve ter uma capacidade ajustável para atendimento de pedidos. Pode ser horizontal através da inclusão de mais nós e/ou vertical através de adicionar mais recursos aos nós existentes. A escalabilidade horizontal deve contar com balanceamento de carga.

Estado: Concluído

Percentagem concluída: 70%

Aspetos implementados:

* O atendimento de pedidos aos ***servidores web*** é feito através de **balanceadores de carga**, que através do algoritmo **leastconn** redirecionam os pedidos HTTPS sob HTTP;
* O atendimento de pedidos das ***APIs*** é feito através de **balanceadores de carga**, que através do algoritmo ***leastconn*** redirecionam os pedidos HTTP dos servidores web;
* São instanciadas, inicialmente, **2** máquinas virtuais para os balanceadores de carga dos servidores web, **2** máquinas para os servidores web, **2** para os balanceadores de carga da APIs, **2** para as APIs e **2** para as bases de dados;
* As máquinas dos servidores foram reconfiguradas para melhor suportar as necessidades de computação do projeto.

Aspetos em falta:

* Utilização de *scripts* para a criação/remoção de máquinas conforme a necessidade do sistema.

**RNF-06 – Elasticidade:** É desejável que o sistema possua capacidade de elasticidade. Embora este não seja um requisito obrigatório para o projeto, será valorizado na avaliação

Estado: Não desenvolvido

**RNF-07 – Tolerância a faltas:** O sistema deve tolerar a falha de qualquer dos seus componentes, através de principalmente oferecer redundância ativa.

Estado: Concluído

Percentagem concluída: 100%

Aspetos implementados:

* É utilizada redundância **ativa-passiva** no *cluster* de balanceadores de carga. Através do ***keepalived***, a máquina de reserva assume funções caso a primeira falhe, e vice-versa;
* É utilizada redundância **ativa-ativa** nas instâncias das bases de dados. Através da replicação do **PostgreSQL**, a máquina secundária regista todas as alterações feitas à primeira.

**RNF-08 – Disponibilidade:** O sistema deve responder a qualquer pedido dentro de um tempo razoável pré-definido.

Estado: Em desenvolvimento

Percentagem concluída: 50%

Aspetos implementados:

* Foram definidas **métricas** para tempos de resposta razoáveis, da API, do servidor e da base de dados;
* O sistema foi construído utilizando principalmente comunicação por **redes internas**, para melhorar a performance e **diminuir a latência**.
* As métricas foram comparadas com os tempos de referência através de testes individuais ou através do browser, por exemplo, verificando os logs das máquinas virtuais ou a consola na secção de rede.

Aspetos em falta:

* Testar o cumprimento das métricas com o Apache Jmeter

**RNF-09 – Tempo de resposta e desempenho:** O sistema deve ter um tempo de resposta e um desempenho adequados à expectativa dos utilizadores.

Estado: Concluído

Percentagem concluída: 100%

Aspetos implementados:

* Foram definidos valores de referência em comparação com websites de referência, por exemplo de serviço semelhantes ou de acesso recorrente por parte dos utilizadores, além dos valores referidos no RNF-08.
* Foram feitos testes com utilizadores para a averiguar o desempenho e como se compara às expectativas dos utilizadores.

**RNF-10 – Nome de domínio:** Deve ser criado e registado um nome de domínio para obtenção do certificado.

Estado: Concluído

Percentagem concluída: 100%

Aspetos implementados:

* Foi registado o domínio **tesourosperdidos.me**, através da plataforma **amen.pt**, e de forma gratuita.

**RNF-11 – Segurança – Canais seguros:** Obtenção de um certificado.

Estado: Concluído

Percentagem concluída: 100%

Aspetos implementados:

* Foi obtido um certificado para a comunicação direta com os balanceadores de carga, utilizando a ferramenta ***certbot***. A autoridade de certificação é a *Let’s encrypt*.

**RNF-12 – Segurança – Autenticação e autorização:** O sistema computacional e as redes devem estar cobertos por políticas de segurança que fazem uso dos mecanismos de proteção adequados de modo a impedir o acesso não-autorizado a recursos e dados e também impedir a disrupção do serviço.

Estado: Concluído

Percentagem concluída: 100%

Aspetos implementados:

* As máquinas virtuais não têm IPs externos diretamente associados, e o acesso ssh é restringido para os administradores do sistema;
* Apenas aqueles que configuram máquinas virtuais têm acesso às mesmas.

**RNF-13 – Privacidade:** O sistema deve estar de acordo com os regulamentos para a privacidade dos dados (RGPD).

Estado: Concluído

Percentagem concluída: 100%

Aspetos implementados:

* Foi criada uma página com discriminação do tratamento e recolha dos dados dos utilizadores, de acordo com os princípios do RGPD (<https://en.wikipedia.org/wiki/General_Data_Protection_Regulation>);
* Foi implementado um formulário de consentimento para a recolha e utilização dos dados, necessário para o registo dos utilizadores, onde são descritas as políticas de privacidade e recolhas de dados do website.

**RNF-14 – Segurança - Configuração de firewall:** O sistema computacional e as redes devem estar cobertas por políticas de segurança que fazem uso dos mecanismos de proteção adequados de modo a impedir o acesso não-autorizado a recursos e dados e também impedir a disrupção do serviço.

Estado: Concluído

Percentagem concluída: 100%

Aspetos implementados:

* Foi implementada segurança ***estática***, sob a forma da configuração de regras das ***firewalls*** de cada instância com o software ***iptables***.

**RNF-15 – Integridade dos dados:** A infraestrutura deve assegurar a consistência dos dados.

Estado: Concluído

Percentagem concluída: 100%

Aspetos implementados:

* A bases de dados conta com uma segunda instância em replicação e backup, garantindo não só que os dados não são perdidos, mas também que a leitura e escrita dos dados são uniformes.
* As duas instâncias de API e Servidores Web permitem escritas e leituras concorrentes, com a garantia de que leêm ou atualizam sempre os dados mais recentes.
* Também podemos dizer que obtivemos consistência perfeita, sendo que sempre que se faz uma leitura, a versão lida é a mais recente.

**RNF-16 – Salvaguarda de dados:** Os dados devem ser salvaguardados regularmente através de uma política adequada à importância dos dados e configurações. Adicionalmente deve estar definido um procedimento de restauro de qualquer parte dos dados.

Estado: Em desenvolvimento

Percentagem concluída: 50%

Aspetos implementados:

* Os dados dos utilizadores são armazenados em instâncias apenas comunicáveis por máquinas na sua sub-rede, por endereços internos. Em conjunto com regras das bases de dados, são dificultados ataques diretamente aos dados armazenados no sistema
* As palavras-passe dos utilizadores são armazenadas no Auth0, de forma que não estejam na base de dados em si, reduzindo ainda mais a probabilidade de serem descobertas.

Aspetos em falta:

* Fazer um plano para um procedimento de restauro de dados em caso de falha no sistema.

**RNF-17 – Custo:** Ponderando o custo das várias soluções, deve ser escolhida uma configuração que preste o melhor serviço pelo preço mais baixo. A escolha deve ser justificada tendo por base as várias opções.

Estado: Concluído

Percentagem concluída: 100%

Aspetos implementados:

* Todas as máquinas foram configuradas com a configuração de menor custo, que ao mesmo tempo suportasse as necessidades de projeto.

**RNF-18 – Testes - carga e vulnerabilidade:** O sistema deve ser desenvolvido com uma política integrada de testes que permitem ir testando à medida que se adicionam novos componentes ou que

se alteram componentes existentes.

Estado: Não desenvolvido

## Design da Arquitetura distribuída – (10.0%)

Uma imagem com captura de ecrã, espaço

Descrição gerada automaticamente

Uma imagem com captura de ecrã, diagrama

Descrição gerada automaticamente

Uma imagem com captura de ecrã, círculo, diagrama

Descrição gerada automaticamente

Uma imagem com captura de ecrã, diagrama, texto, círculo

Descrição gerada automaticamente

## Especificação API – (10.0%)

O nome do ficheiro é o tesourosperdidos.yaml.

No nosso YAML obtemos um erro ao qual não conseguimos dar solução:  
Uma imagem com texto, captura de ecrã, Tipo de letra, file

Descrição gerada automaticamente

Na descrição do erro é apontada a seguinte linha:  
Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

E nesta não há referências ao esclarecimento do erro.

Na descrição do erro é referido o caminho:

path./foundobjects/{id}/change-owner.put.requestBody.content.application/json.schema.$ref

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

Descrição gerada automaticamente

## Especificação dos Requisitos NF - escalabilidade – (1.0%)

* Devem ser usados load balancers para distribuir o volume de tráfego equitativamente pelos containers em uso e/ou ajustar o número de servidores em uso;
* Devem ser simuladas situações como spikes de tráfego ou falhas de servidores para garantir o pleno funcionamento dos load balancers.
* Devem ser monitorizados os níveis de saúde dos sistemas, como o tempo resposta e a taxa de transferência de dados de cada servidor através de scripts.
* Devem ser definidos limites para os tempos de resposta, e para a carga dos servidores sob métricas como uso do CPU e da memória.

## Especificação dos Requisitos NF - segurança – (1.0%)

* Devem ser configuradas sub-redes de forma a segregar a comunicação interna e externa, e eliminando a dependência da comunicação com os servidores num IP externo;
* Devem ser implementadas políticas de acesso às componentes do sistema como o princípio do menor privilégio, utilizando um sistema hierárquico para reduzir o acesso de dados transversalmente;
* Devem ser gerados e configurados certificados TLS de forma a encriptar a comunicação com os servidores;
* Além da encriptação dos dados, deve ser minimizada a recolha de dados dos utilizadores, e expostas as políticas de recolha e consentimento de dados;
* Devem ser realizados scans e revistas as políticas de acesso semanalmente, de forma a monitorizar a integridade dos sistemas.

## Especificação dos Requisitos NF - tolerância a faltas – (1.0%)

* Devem ser isoladas as várias funções do sistema de forma a evitar “cascading failures”;
* Deve ser estabelecido um plano de resposta a fugas de dados, falhas nos componentes ou falhas dos ambientes de trabalho;
* Devem ser configurados alertas através de canais como o email, de fácil acesso, caso exista alguma falha no sistema;
* Deve ser usada uma política de comunicação VRRP (Virtual Router Redundancy Protocol), através de clusters ativo-passivos nos load balancers e bases de dados;
* Devem ser definidos processos de restart automático de componentes em falha.

## GitHub com branches individuais e releases – (4.0%)

O repositório de nome PTI-PTR, disponível em: <https://github.com/MartimEmauz/PTI-PTR>, foi configurado com um *branch* de desenvolvimento para **cada membro do projeto**, e contém **código** da aplicação web desenvolvida. Além disso, contém também os registos das **atas** de reuniões SCRUM, **guiões** e **scripts** para ajudar os membros a configurar e instalar dependências necessárias.

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

Descrição gerada automaticamente

São utilizados ***releases*** com uma **nomenclatura** referente à versão do projeto. No caso de versões de desenvolvimento, utilizamos v0.X.X, e as versões de desenvolvimento começarão por 1 (v1.X.X).

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

Descrição gerada automaticamente

## Implementação da arquitetura distribuída – (12.0%)

< Apresente nesta seção os detalhes de implementação do sistema. Descreva-o detalhando as instâncias de VM e configurações de rede. Detalhe a implementação dos balanceadores de carga (7%), dos web servers (2,5%) e das bases de dados (2,5%). >

## Balanceador de carga (e.g., HAProxy, Nginx) e escalabilidade – (10.0%)

Os balanceadores de carga estão instanciados em máquinas virtuais (VMs) na Google Cloud Platform (GCP). Estes estão configurados no modo ativo-passivo através do Keepalived, instalado nas VMs, sendo a comunicação entre os load balancers feita via VRRP para otimizar o processo de resposta a pedidos. Através do HAProxy, garantimos que os serviços internos dos load balancers estão otimizados, a filtração de pedidos e a preparação do envio dos mesmos em formato HTTP aos webservers, de modo a assegurar maior segurança. Os load balancers têm também os benefícios da configuração atual do iptables, ao restringir os ips com a possibilidade de fazer contacto com as VMs através de determinados portos, sendo liberado os portos para comunicação interna (112 VRRP), e comunicação segura com a rede interna (80 http e 443 https).

## Tolerância a faltas e verificação de saúde – (5.0%)

< Explique os mecanismos de tolerância a falta implementados para os balanceadores de carga, *web servers* e bases de dados.>

## Implementação de mecanismo de Autenticação e Autorização (Auth0) – (5.0%)

Começámos por criar uma conta através do Auth0 Dashboard. Através deste, criámos uma aplicação com os seguintes parâmetros:

* **Nome da Aplicação:** "PTI-PTR"
* **Tipo de Aplicação:** "Single Page Web Applications"
* **URLs de Callback e Logout Configuradas:**
  + - Callback URL: http://localhost:4200
    - Logout URL: <http://localhost:4200>

Ainda no Auth0 Dashboard, foi criada a API Django com os parâmetros:

* + **Nome da API:** "API"
  + **Identificador (audience):** http://127.0.0.1:8000/

Parar a implementação dos mecanismos no Projeto Angular, foram feitas as seguintes instalações e configurações:

1. **Instalação do Auth0 SDK:**
   * O pacote @auth0/auth0-angular foi instalado no projeto Angular:

**npm install @auth0/auth0-angular**

1. **Configuração do Auth0 no Angular:**
   * No arquivo app.module.ts, o módulo Auth0 foi configurado com os detalhes da aplicação Angular:

AuthModule.forRoot({

domain: 'dev-7ety36lrjtchzs5o.us.auth0.com',

clientId: 'QtBbOFF8p3ObAlrfl8NvoARtRysybsAi',

audience: 'http://127.0.0.1:8000/',

redirectUri: 'http://localhost:4200/',

})

1. **Proteção de Rotas com AuthGuard:**
   * O AuthGuard do Auth0 foi utilizado para proteger rotas que requerem autenticação.
2. **Implementação de Componentes de Login e Logout:**
   * Métodos de login e logout foram implementados através do SDK do Auth0 em componentes Angular.
3. **Realização de Chamadas de API Autenticadas:**
   * O token de acesso fornecido pelo Auth0 foi utilizado para fazer chamadas autenticadas à API Django. Isso foi implementado num interceptor HTTP.

Foi também configurado o Backend em Django, através dos seguintes passos:

1. **Instalação de Dependências Python:**

**pip install python-jose pyjwt**

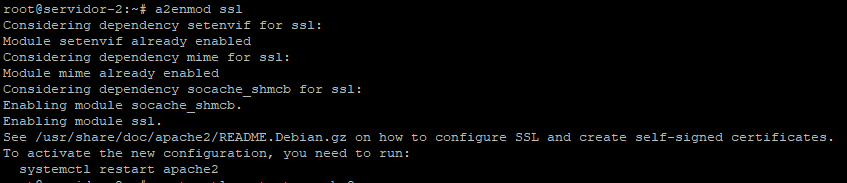
1. **Validação do Token JWT Implementada:**
   * Um middleware de validação foi implementado no backend Django para verificar a validade e as reivindicações do token JWT fornecido pelo Auth0.
2. **Validação da Audiência (audience) Configurada:**
   * Foi configurada a validação da audiência para garantir que o token JWT inclua o identificador correto da API.
3. **Views Seguras:**
   * Utilize a classe de permissão *IsAuthenticated* nas vistas para restringir o acesso. Exemplo: Em uma vista baseada em classe, defina permission\_classes = [IsAuthenticated].

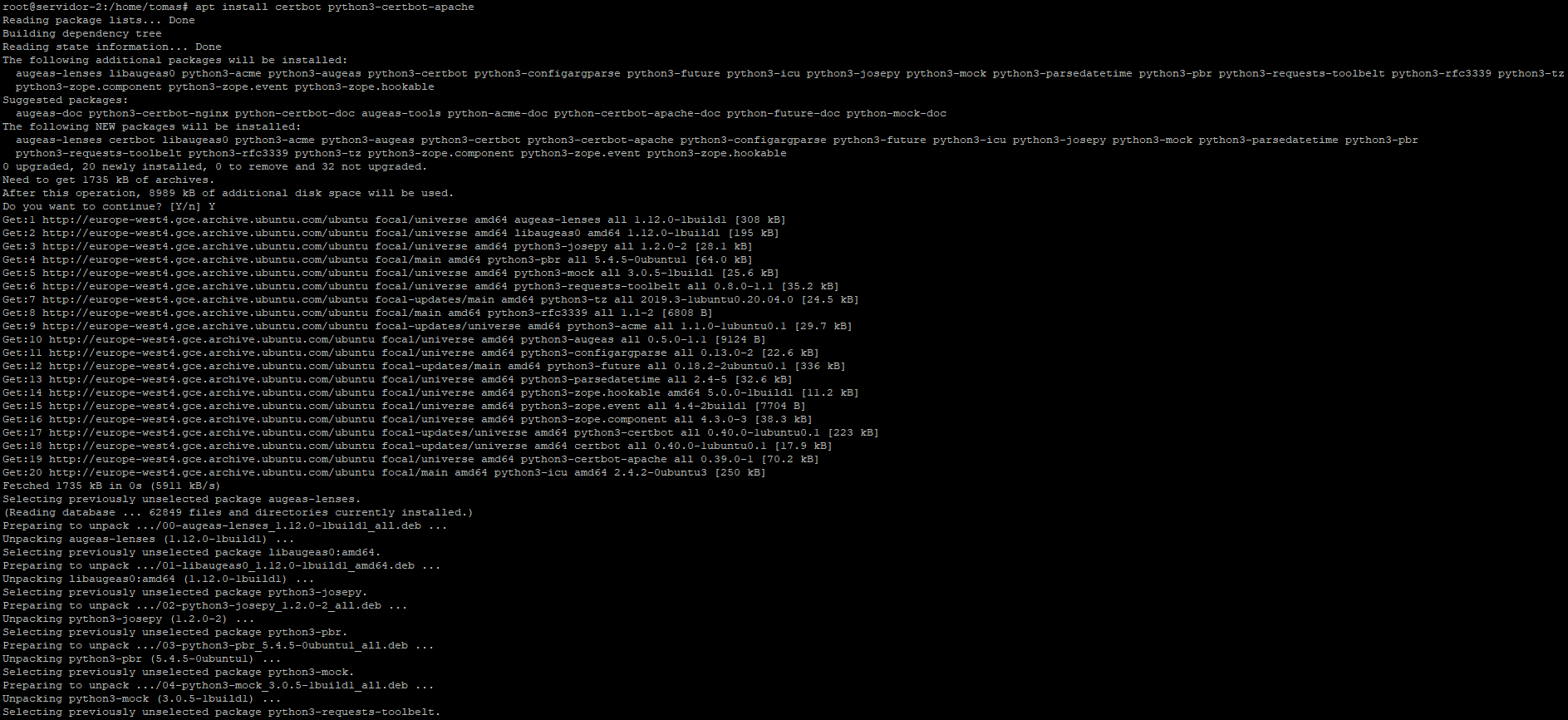
## Canais seguros, DNS e configuração de firewall – (10.0%)

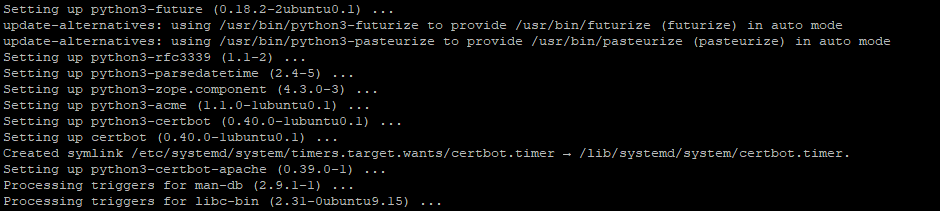
### Uso de TLS com certificado assinado por uma Autoridade Certificadora – (4.0%)

A implementação do TLS foi efetuada nos load balancers no intuito de garantir a escaladbilidade dos webservers, assim garantindo, que os webservers meramente apresentam a informação do backend e os load balancers processam tudo relacionado aos pedidos dos utilizadores.

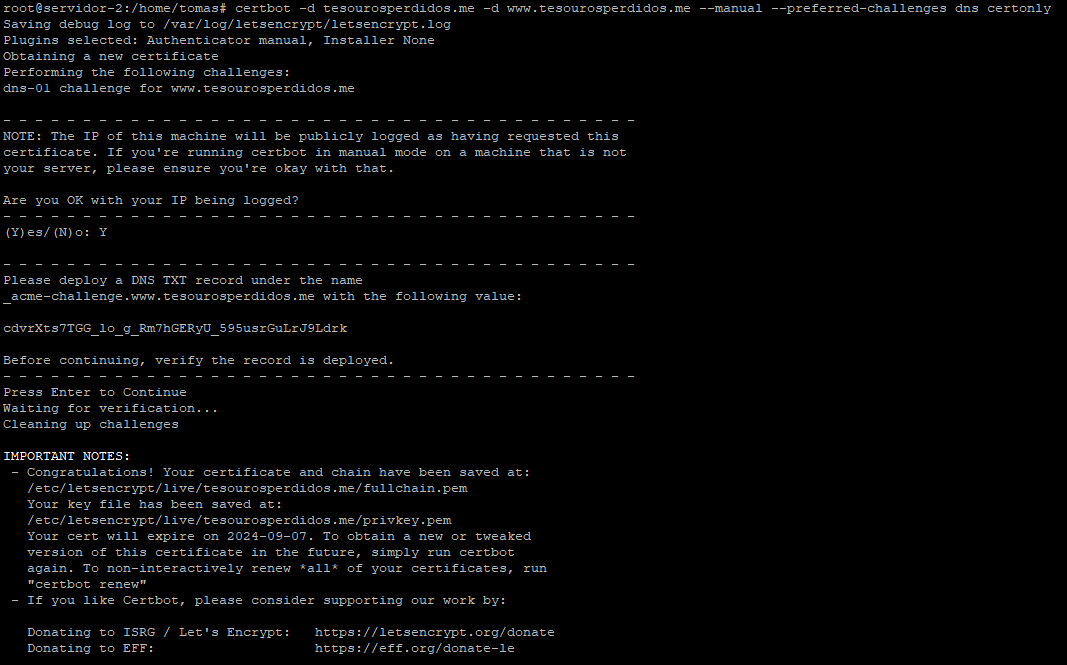
Esta instalação foi feita na seguinte ordem:

* sudo –s
* a2enmod ssl 
* apt install certbot python3-certbot-apache

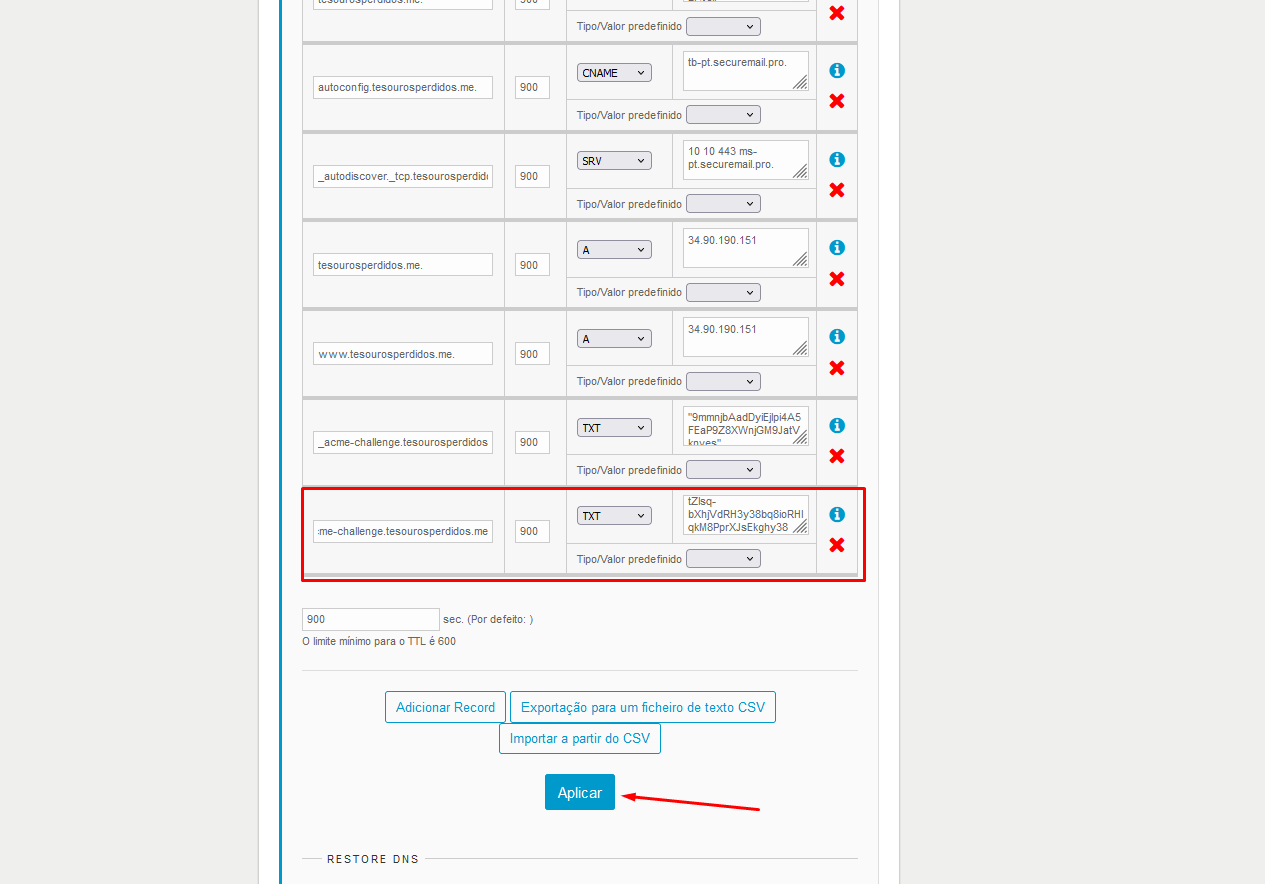




* systemctl status certbot.timer
* certbot -d tesourosperdidos.me -d www.tesourosperdidos.me --manual --preferred-challenges dns certonly



* sudo sh -c 'cat /etc/letsencrypt/live/tesourosperdidos.me/fullchain.pem /etc/letsencrypt/live/tesourosperdidos.me/privkey.pem /etc/haproxy/dhparams.pem > /etc/letsencrypt/live/tesourosperdidos.me/haproxy.pem'
* Ir ao gestor de serviços de DNS, no nosso caso amen.pt, e associar os ficheiros da chave e do certificado ao file que o certificado utiliza para testar a ligação



* nano /etc/haproxy/haproxy.conf

IMAGEM DA CONF ATUAL Do haproxy

* systemctl restart haproxy

Tendo em conta que os certificados estão do lado dos load balancers, os webservers basta decifrarem o cabeçalho enviado pelo load balancer e mostrar o website.

### Nome de domínio registado e associado ao IP estático – (2.0%)

Na plataforma amen.pt, foi verificada a disponibilidade do domínio tesourosperdidos, com várias extensões (“.com”, “.pt”, “.net”). A extensão gratuita mais compacta disponível era “.me”.

Após a reserva do domínio, foram alterados os registos da lista de DNS para que “**tesourosperdidos.me**” e “**www.tesourosperdidos.me**”, ficassem associados ao IP externo dos balanceadores de carga (**34.90.190.151**).

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

Descrição gerada automaticamente

De seguida, foi atualizada a configuração do apache nos servidores web, para definir os nomes do servidor relacionados com o domínio:

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

Descrição gerada automaticamente

### Configurações de Firewall – (4.0%)

< Apresente e explique todas as configurações de firewall realizadas.>

## APIs externas – (2.0%)

**Google Maps API**

* Na secção do produto “APIs e serviços”, na biblioteca foi selecionada e ativada a API do Google Maps, e associada ao projeto PTI-PTR;
* Após a ativação, foi copiada e registada a chave desta para utilização no projeto angular;
* No projeto, foi instalada a biblioteca necessária com o comando:

npm install @agm/core

* Feitas as importações, a seguinte função carrega o mapa no frontend:

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

Descrição gerada automaticamente

**Nota: Não foi implementada a API do Stripe para pagamentos**

## Gestão de credenciais e IPs na instalação – (5.0%)

< Explique como fazem a gestão de credenciais e IPs na instalação.>

## Automação e scripts para construção e lançamento da aplicação – (2.0%)

< Explique os mecanismos de automação e scripts utilizados para construção e lançamento da aplicação.>

## Inclusão de testes unitários integrados ao processo de construção da aplicação (GitHub) – (3.0%)

< Apresente os testes unitários integrados ao processo de construção da aplicação no GitHub. >

## Testes de aceitação com a API – (3.0%)

< Reporte os testes de aceitação realizados na API. >

## Testes de carga – (3.0%)

< Reporte os testes de carga realizados. >

## Testes de vulnerabilidades – (3.0%)

< Reporte os testes de vulnerabilidades realizados. >