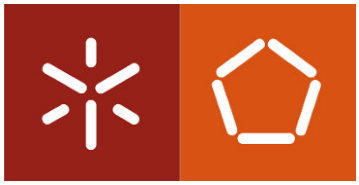

Universidade do Minho

Comunicações por Computador

Departamento de Informática

TP2: Implementação de Sistema DNS

Grupo 5

João Escudeiro (A96075) Hugo Martins (A95125)
Tiago Ribeiro (A76420)

27 de janeiro de 2023

Conteúdo

Lista de Figuras

Lista de Tabelas

1 Introdução

No âmbito do trabalho prático da Unidade Curricular de Comunicações por Computador, o objetivo era a implementação de um **Sistema DNS** um pouco menos complexo do que os sistemas DNS usados no dia à dia, mas que permitisse perceber a forma como os mesmos trabalham, a troca de *Queries* entre cliente e servidor e ainda as comunicações entre Servidores primário e secundário. Nesta segunda fase , os principais objetivos , a par da correção de bugs da fase anterior , seriam a implementação dos componentes que ainda não estavam criados(Servidor de Topo , Servidor de Resolução , Servidor de Zona Reversa).Estes componentes participam , a par dos SDT e subdomínios, no modo iterativo , cujo modo de funcionamento é explicado abaixo .

2 Arquitetura do Sistema

A arquitetura do nosso sistema, como demonstra a figura 1, será composta pelos seguintes componentes: Servidor Primário, Servidor Secundário, Servidor de Resolução,Servidor de Topo , Servidor Reverse e Cliente.

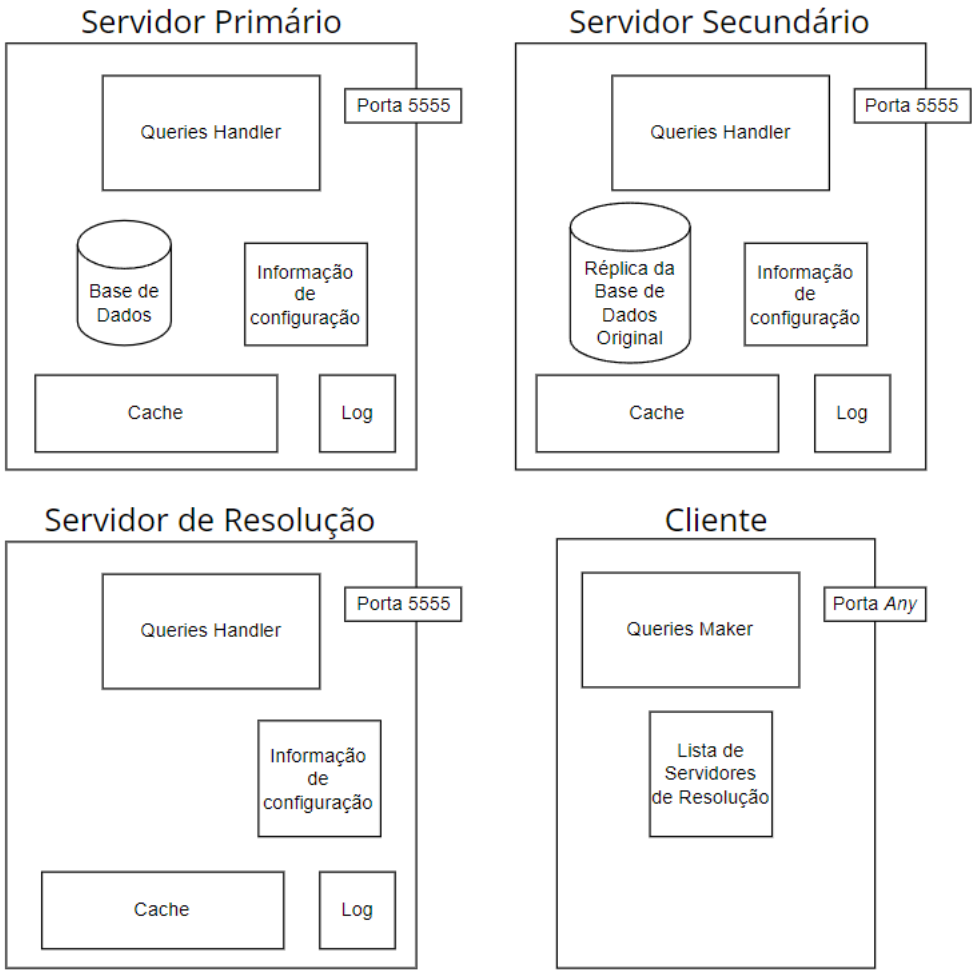

Figura 1: Representação do modelo da Arquitetura do Sistema.

De notar que, para efeitos de simplificação neste contexto académico, será usada a porta 5555 como porta de entrada e de saída, isto é, todos os pedidos para os servidores serão realizados através da porta 5555 , mas as respostas são enviadas através de uma nova porta,

normalmente atribuída aleatoriamente, com a funcionalidade de enviar as respostas aos pedidos através desta nova porta criada, para evitar "congestionamentos".

Os requisitos funcionais para cada componente serão destacados na seguinte tabela 1:

Servidor Primário	**Servidor Secundário**
Responder e efetuar Queries DNS	Responder e efetuar Queries DNS
Acesso direto à base de dados	Acesso a uma réplica da base de dados original
Acesso à informação de configuração	Acesso à informação de configuração
Armazenamento de dados em cache	Armazenamento de dados em cache
Geração de um ficheiro log	Geração de um ficheiro log
Utilização da porta 5555	Utilização da porta 5555
Servidor de Resolução	**Cliente**
Responder e efetuar Queries DNS	Efetuar Queries DNS
Acesso à informação de configuração	Acesso a uma lista de SR
Armazenamento de dados em cache	
Geração de um ficheiro log	
Utilização da porta 5555	

Tabela 1: Requisitos funcionais das componentes do sistema.

Cada componente comunicará entre si através de mensagens DNS encapsuladas no protocolo UDP, à exceção da transferência de zona que será realizada através de uma ligação TCP.

De notar ainda que bastantes componentes do sistema possuem funcionalidades iguais. Isto significa que a nível de implementação, iremos optar por um desenvolvimento de código modular o que irá nos permitir a reutilização de métodos ou funções.

3 Modelo Comunicacional do Sistema

As interações possíveis entre os componentes do sistema são : comunicação entre o cliente e o servidor , para resolução de Queries , que é feita por uma mensagem DNS.

3.1 Modelo de Informação

Esta mensagem DNS, referida anteriormente, é especificada agora com base na seguinte figura 2:

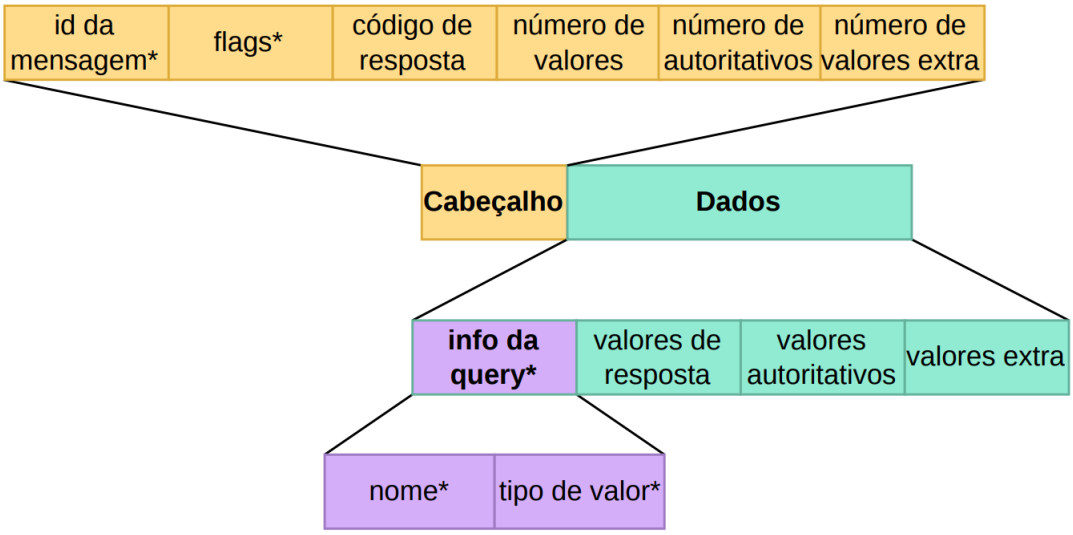

Figura 2: A representação lógica da mensagem de DNS usada no sistema.

Cada campo nesta mensagem DNS será agora descrita nos seguintes módulos:

Um campo **Cabeçalho** que será composto pelos seguintes campos:

- **id da mensagem***: *Integer* (de 1 até 65535 números inteiros), em que cada número é um identificador da mensagem que será usado para relacionar as respostas recebidas com a query original.
- **flags***: *Char*, em que cada char indica o tipo da mensagem. Serão usados os caracteres Q,R e A, sendo que o caractere Q indica que a mensagem é uma query ou uma resposta à mesma; o caractere R indica que se pretende que o processo opere de forma recursiva e não iterativa (que será o modo por defeito) numa query, e numa resposta indica que o servidor que respondeu suporta o modo recursivo; o caractere A indica que a resposta é autoritativa, sendo que este valor A nas Queries é ignorado.
- **código de resposta**: *Integer*, em que indica o código de erro na resposta. Serão usados os códigos de 0 até 3, sendo que o código é igual a 0 sempre que não ocorra nenhum erro. Será igual a 1 sempre que o domínio especificado no campo ***nome*** existir mas não tiver sido encontrado um tipo de valor igual ao especificado no campo ***tipo de valor***. Será igual a 2 sempre que o domínio especificado em ***nome*** não existir. E será igual a 3 sempre que a mensagem DNS não tenha sido descodificada corretamente.
- **número de valores**: *Integer*, em que quantifica o número de entradas relevantes (até um máximo de 255 entradas) à query.
- **número de autoritativos**: *Integer*, que quantifica o número de entradas (até um máximo de 255 entradas) que identificam os servidores autoritativos para o domínio especificado.
- **número de valores extra**: *Integer*, que identifica o número de entradas (até um máximo de 255 entradas) com informação adicional.

Um campo **Dados** que será composto pelos seguintes campos:

- **info da query***: Composta pelos campos:
  - **nome***: *String*, que identifica a informação do parâmetro da query.
  - **tipo de valor***: *String*, que identifica o tipo de valor associado ao parâmetro. Sendo que os tipos suportados serão iguais aos da sintaxde dos ficheiros de base de dados dos SP.
- **valores de resposta**: *Lista de Strings*, em que cada string é um *match* dos campos especificados em ***nome*** e ***tipo de valor***.
- **valores autoritativos**: *Lista de Strings*, em que cada string é um *match* do campo especificado em ***nome*** e em que o campo ***tipo de valor*** é igual a "NS".
- **valores extra**: *Lista de Strings*, em que cada string é uma entrada do tipo A e que fazem *match* no parâmetro com todos os valores nos campos ***valores de resposta*** e ***valores autoritativos***.

Nota: Para os últimos 3 campos, a informação obtida e inserida nestas listas de strings será recolhida da cache ou da base de dados do servidor autoritativo.

É de notar ainda que os campos assinalados com um asterisco serão os únicos usados numa mensagem query. Ou seja, todos os restantes campos serão ignorados, isto é, colocados a zero. A figura 3 seguinte, demonstra o PDU para uma query DNS mencionada anteriormente.

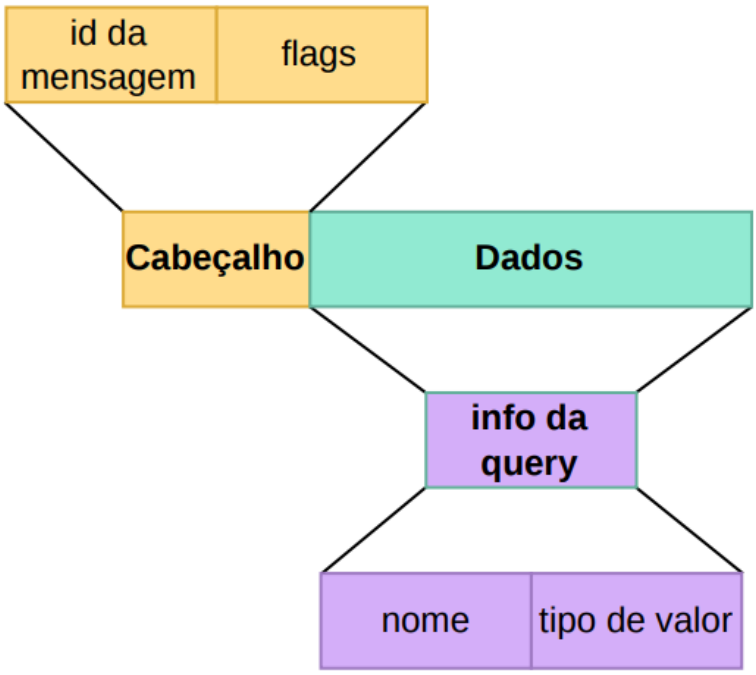

Figura 3: A representação lógica de uma *query* DNS usada no sistema.

3.2 Modelo de Comunicação

Como é mostrado na figura 4 , as diversas iterações entre os componentes (CL,ST,SR,SDT), ajudarão a obter a resposta a pedidos para um dado domínio . O maior "Responsável"é o SR , que é capaz de receber e redirecionar as Queries para os diferentes componentes do sistema , permitindo assim obter a resposta correta às Queries.

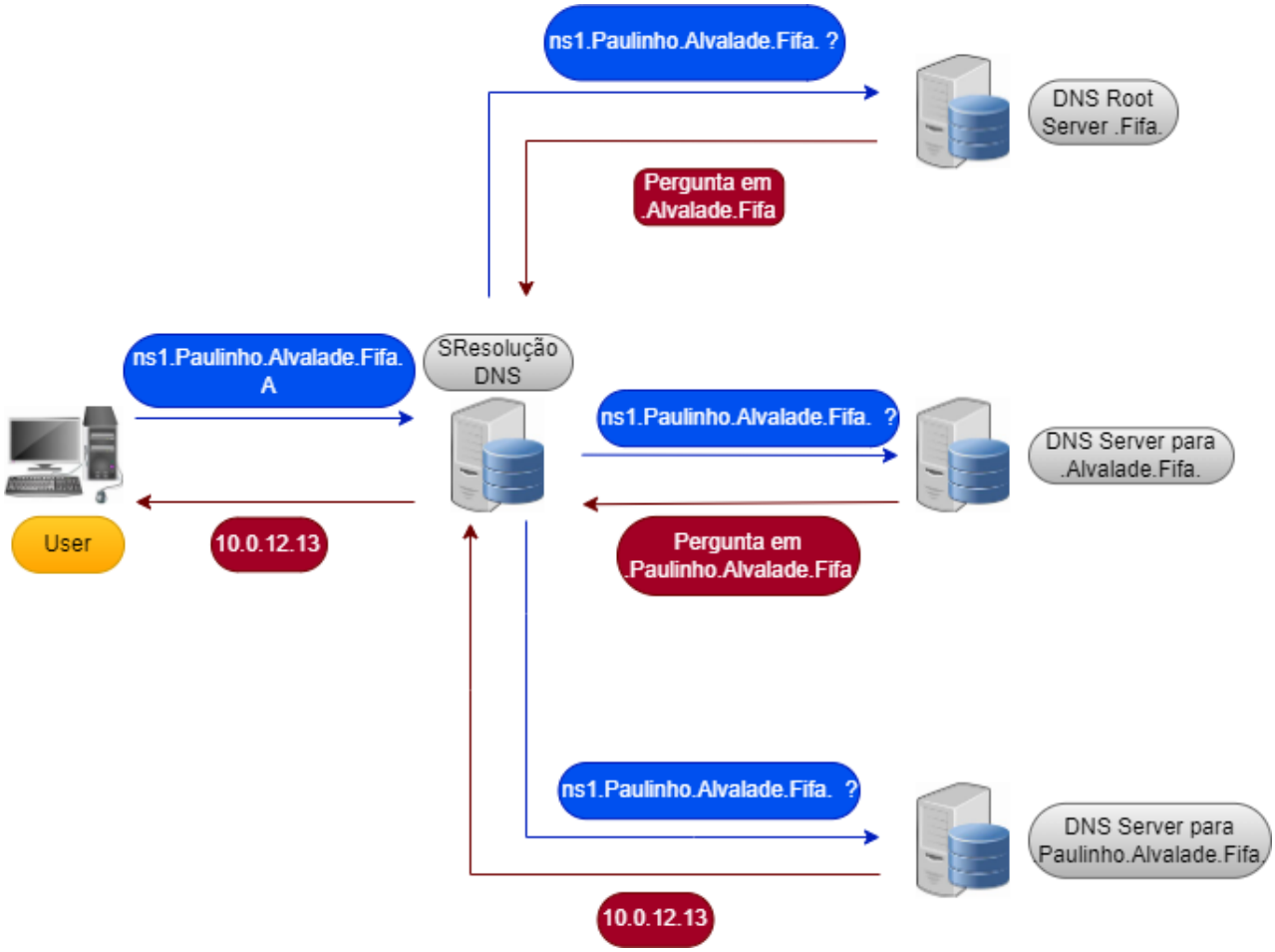

Figura 4: Exemplo de interações do modo iterativo entre os componentes do sistema .

Para a realização do Modelo de Comunicação foram analisadas todas as interações possíveis entre os elementos e qual o comportamento a adotar aquando uma situação de erro. Assim foram analisadas as comunicações feitas entre um SS e um SP durante uma transferência de zona, como se pode observar na figura 6 e as comunicações feitas entre um cliente e um servidor durante um pedido Querie DNS como exemplificado na figura 5. Foi feito ainda, um algoritmo que represente o comportamento do SS numa transferência de zona, um que represente o CL em pedidos de Query DNS e um outro que represente o servidor para todos os casos. Nota: Foi assumido, em todas as situações, que depois de estabelecida a ligação, esta se mantém até ao final do processo.

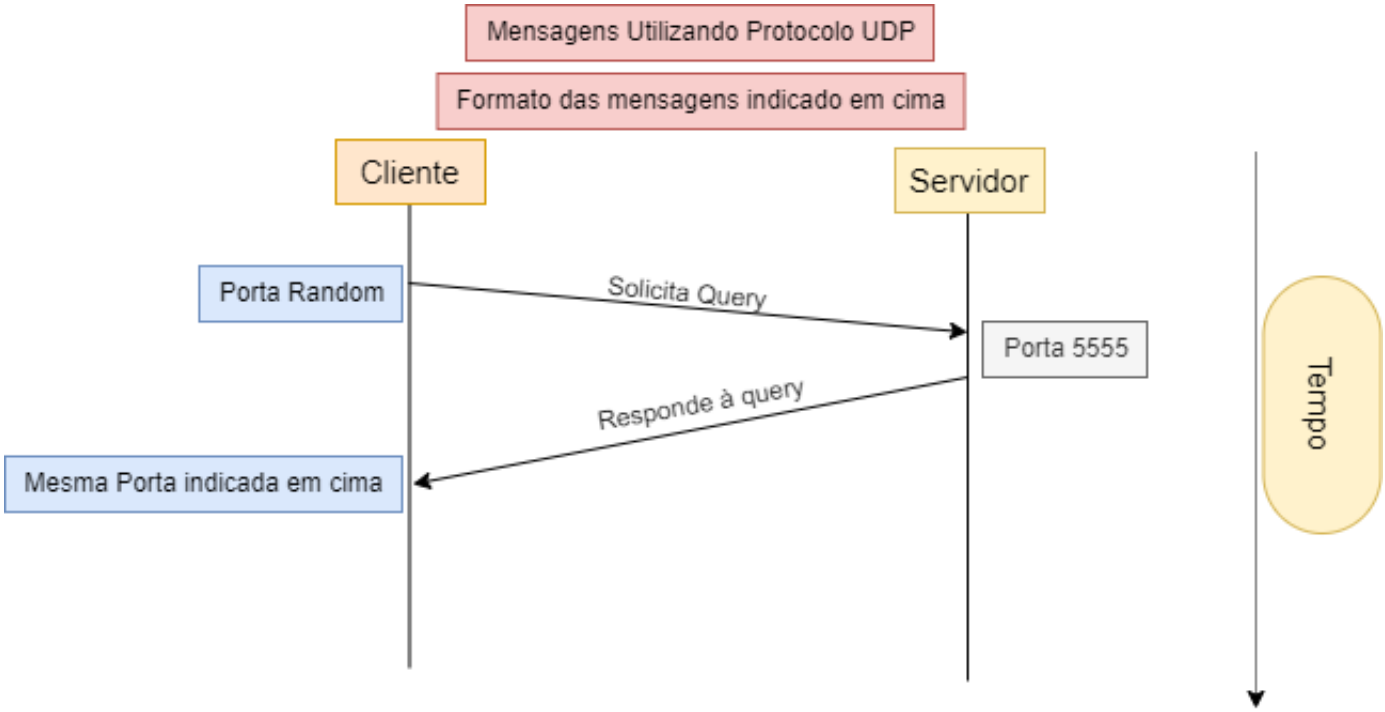

Figura 5: A representação da comunicação entre Cliente/Servidor usada no sistema.

As comunicações representadas , indicam as interações entre cliente-servidor , bem como Servidor Primário -Servidor Secundário . É de realçar que nesta segunda fase foram também incluídas comunicações entre os servidores , através do uso de Queries , para facilitar na realização do modo iterativo . Essas mensagens têm particularidades , e é através da Flag "Response Code"que o servidor que recebe a resposta percebe se encontrou resposta, se o domínio não existe , se tem de fazer forwarding da resposta , ou se ocorreu algum erro na descodificação da resposta.

3.3 Planeamento do Ambiente de Teste

O ambiente de testes será composto por dois domínios denominados de .Alvalade e .Luz, dois subdomínios denominados Paulinho (do .Alvalade) e Enzo (do .Luz),uma zona reverse e dois servidores de topo. Cada um destes domínios terá 1 servidor primário, 2 servidores secundários, 2 servidores de e-mail, 1 servidor web, 1 servidor de resolução e 1 cliente. Cada um destes subdomínios terá 1 servidor primário, 2 servidores secundários, 2 servidores de e-mail, 1 servidor web e 1 servidor de resolução. Na figura 7 podemos observar este nosso ambiente de testes explicado anteriormente.

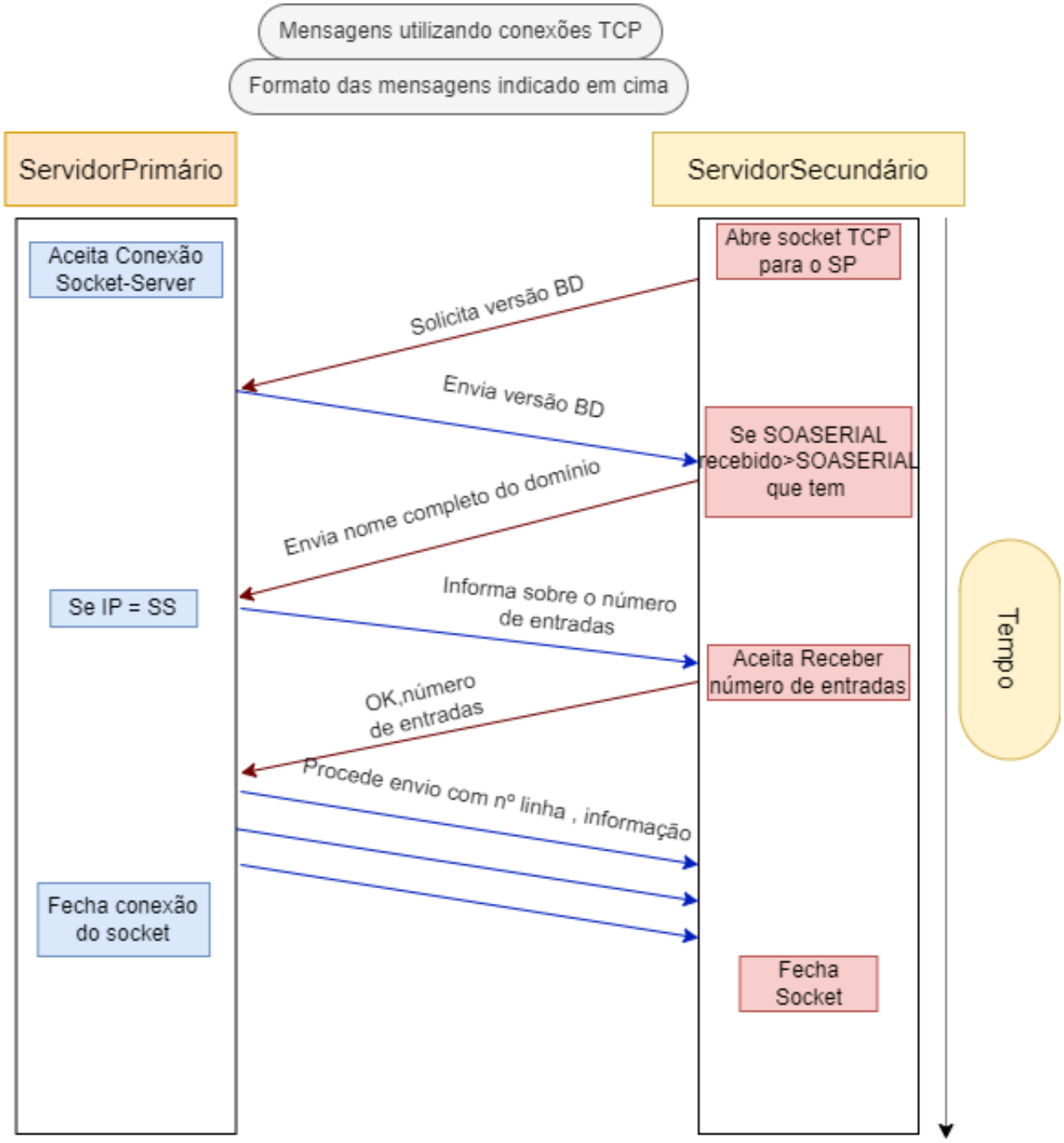

Figura 6: A representação da comunicação entre SP/SS para uma Transferência de Zona.

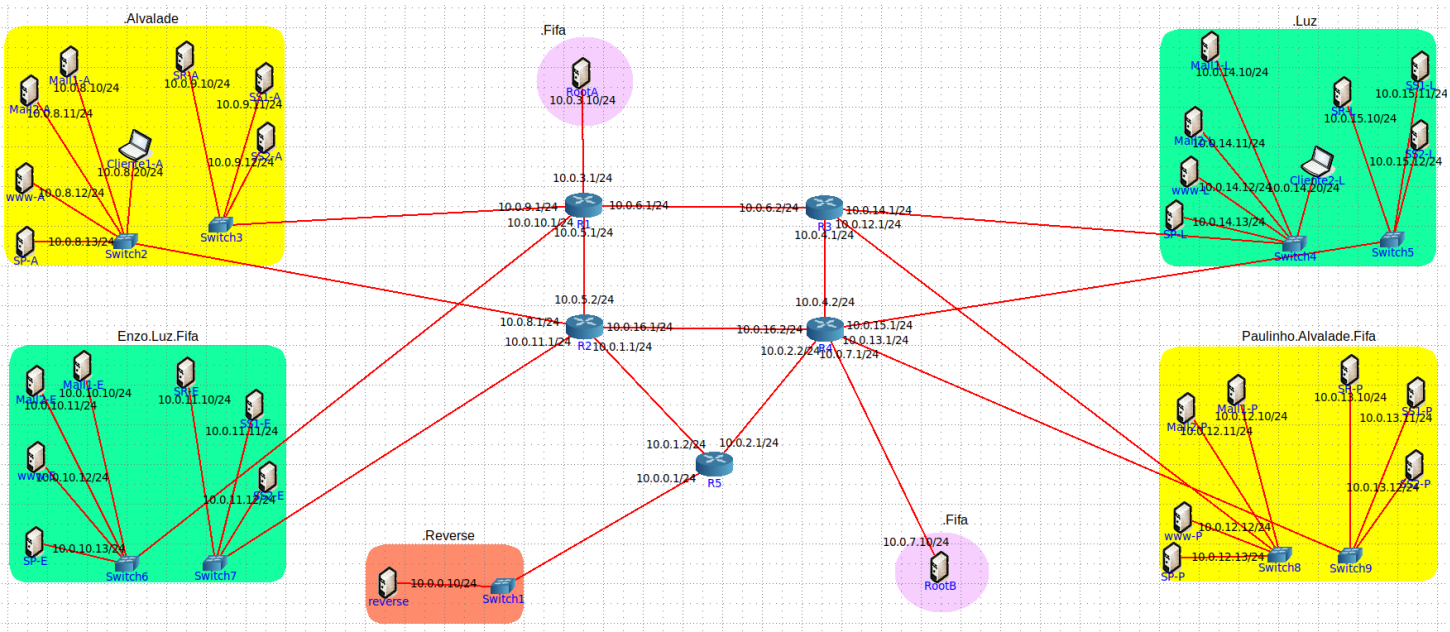

Figura 7: Topologia do ambiente de testes.

4 Estratégias de implementação

4.1 Primeira Fase

Primeiramente optou-se por dar prioridade ao registo dos dados, mais concretamente à leitura e armazenamento do conteúdo do ficheiro de Configs, bem como da Base de dados em memória. Assim, foram criados dois ficheiros (parseDB e parseConfig) que, como o próprio nome indica , servem de apoio ao parse de cada um dos ficheiros. Foram criadas duas Classes que facilitam no armazenamento e procura por campo, utilizando o paradigma de Objetos. Para o ficheiro de Configs, foi criada a classe configLines, que armazena a linha por parametro, tipo e valor . Por exemplo , a linha ("example.com DB files/database.txt") será guardada como : parametro=example.com , tipo=DB,valor =files/database.txt . Após isto , o ficheiro e configs é passado linha a linha para a classe ConfigValues , que contém um array de items , que vai ter em cada posição um objeto do tipo configLines , que terá a informação relativa a cada uma das linhas do ficheiro de configs.

Da mesma forma acontece com a Base de Dados, em que é criada uma classe auxiliar chamada ValParse que guarda as linhas da base de dados em três arrays separando-as pelos seguintes parâmetros: parametro, tipo, valor, tempo, prioridade. Na classe DataBase é guardado o domínio, o TTL, e três listas diferentes. Uma armazena os valores que possam ser passados com @, outra que é a listaUser guarda toda a informação relativamente aos componentes daquele domínio e a listaCname, que como o próprio nome indica guarda todas as linhas que tenham parâmetro do tipo CNAME.

Nota: Um aspeto importante a ter em conta é o facto de o grupo, após uma conversa com um docente, decidiu colocar o IP e a Porta do servidor no seu ficheiro de configs, tornando mais prático a sua obtenção e evitando funções demasiado complexas para esta fase.

Após toda a informação estar tratada e armazenada, espera-se por uma comuninação via TCP do SS para proceder a uma possível transferência de zona como explicado no diagrama da Figura 8 (parte do SS) e Figura 9 (parte do SP).Caso a versão da Base de Dados do SS seja mais desatualizada do que a base de dados do SP, então procede-se ao início da transferência de zona.

Após a conexão TCP ser terminada, o SP, bem como o SS ficam à espera de comunicação via UDP de possíveis clientes que façam as Queries respetivas como é explicado na figura 9 (parte do servidor) e Figura 10 (parte do Cliente). De notar que a parte da implementação que responde ás Queries DNS implementada no SS é idêntica á do SP e está explicada na parte da conexão UDP da Figura 9.

A par deste processo, surge um vasto leque de operações que permite que todas estas comunicações entre SP e SS, bem como Cliente e Servidor funcionem como: criação de sockets, obtenção de IPs e portas, passagem da mensagem para binário, parse das mensagens, utilizando classes auxiliares, entre outros.

4.2 Segunda Fase

Para a segunda fase do TP, optou-se por começar dando prioridade à correção de alguns bugs correspondentes à resposta a Queries por parte do servidor,nomeadamente da Querie A. De seguida, e após estes erros estarem corrigidos, dividimos o trabalho em 4 partes: implementação de um sistema multithreading,implementação do modo iterativo,implementação de cache no servidor de resolução e implementação da zona reversa ou "Reverse Zone". A implementação do sistema multithreading permitiu que os servidores fossem capaz de receber vários pedidos em simultâneo , e enviar as respetivas respostas , sem a necessidade de tempos de espera elevados , da mesma forma que o DNS real . Para efeitos de Controlo de Concorrência , o grupo tomou algumas medidas , sendo feitos os respetivos "locks"e "unlocks"sempre que

eram acedidas estruturas de dados ,como por exemplo a cache e a base de dados de cada um dos servidores. Assim garante-se que há exclusão mútua nestas secções críticas , evitando assim respostas erradas . A implementação de cache no servidor de resolução permitiu-nos evitar muitos ciclos iterativos e tornar o programa mais robusto, uma vez que ao guardar a Querie e a resposta à mesma em cache, permite entregar ao cliente a resposta mais rápido e dimuniundo a probabilidade de erros (caso algum componente falhe), bastando apenas uma comunicação entre o mesmo e o servidor resolução , sem a necessidade de haver comunicação com os outros componentes do sistema , e assim diminuir a latência , que neste projeto e num contexto universitário , em que há um número reduzido de clientes não se nota , mas no DNS real é algo a ter em conta , devido ao elevado número de utilizadores distribuídos por diferentes partes do globo.

Após a construção de todos os ficheiros de configuração e bases de dados necessários tanto para os ST como SR, passamos para a implementação do SR que no modo iterativo é o grande "Protagonista"visto que todas as iterações passam por ele. Após isto e após a construção dos ST (construção que foi facilitada pois apenas se teve de replicar código já realizado, e fazer algumas alterações), chegou-se ao produto iterativo final. Estas implementações estão explicadas em forma mais detalhada em diagrama nas figuras 11 e 12.

Por fim sucedeu-se a implementação da zona reversa do programa.Esta foi implementada em código relativo ao servidor primário, uma vez que, se comporta de maneira semelhante, sendo apenas acrescentado código relativo a Queries "PTR". Por fim foram criados tanto o ficheiros de logs como o de base de dados relativos à mesma.

De seguida deu-se início à construção da base de Dados e do ficheiro de Configs que auxiliará na realização da zona reversa.Após este processo simples , passou-se para a implementação do mesmo. À semelhança de um servidor normal , o servidor da zona reversa , apenas difere na forma como faz a procura na sua base de dados .Um aspeto importante a considerar é que as Queries para o servidor reverso têm uma flag que indica que a mesma é ou nao para a zona reversa,a flag "PTR".

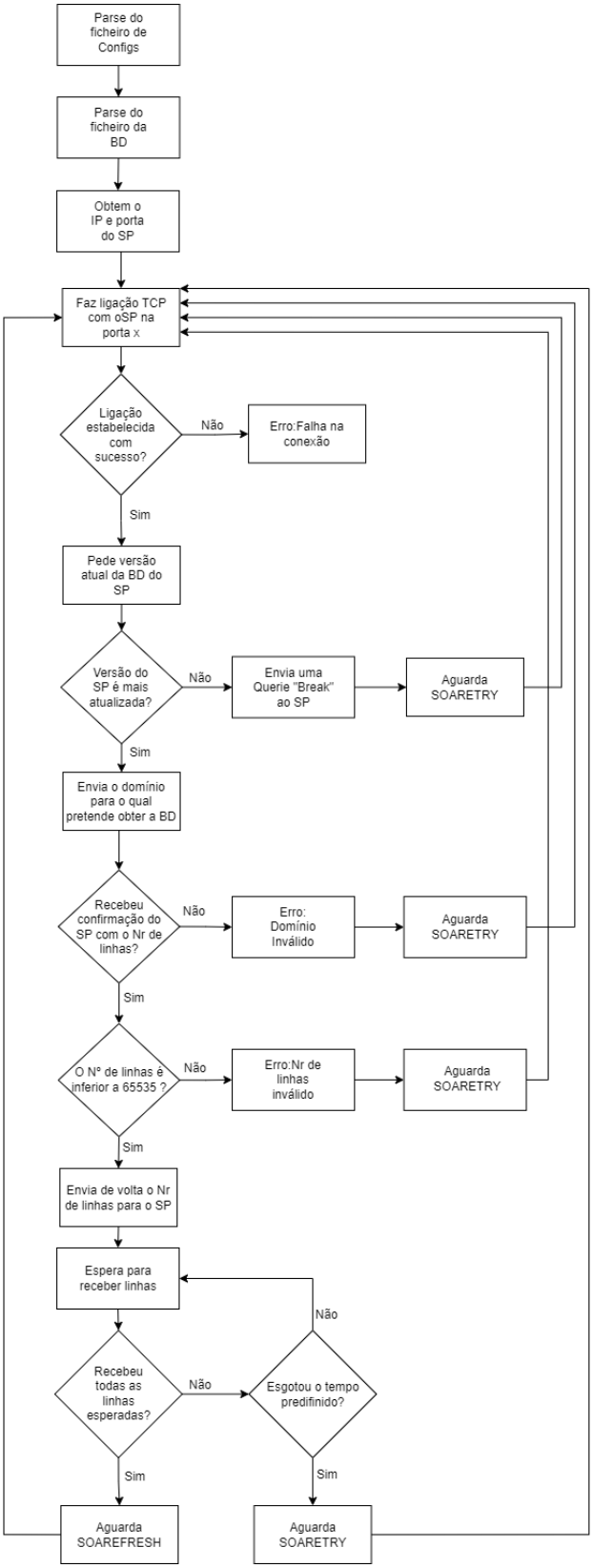

Figura 8: Algoritmo do SS na transferência de zona.

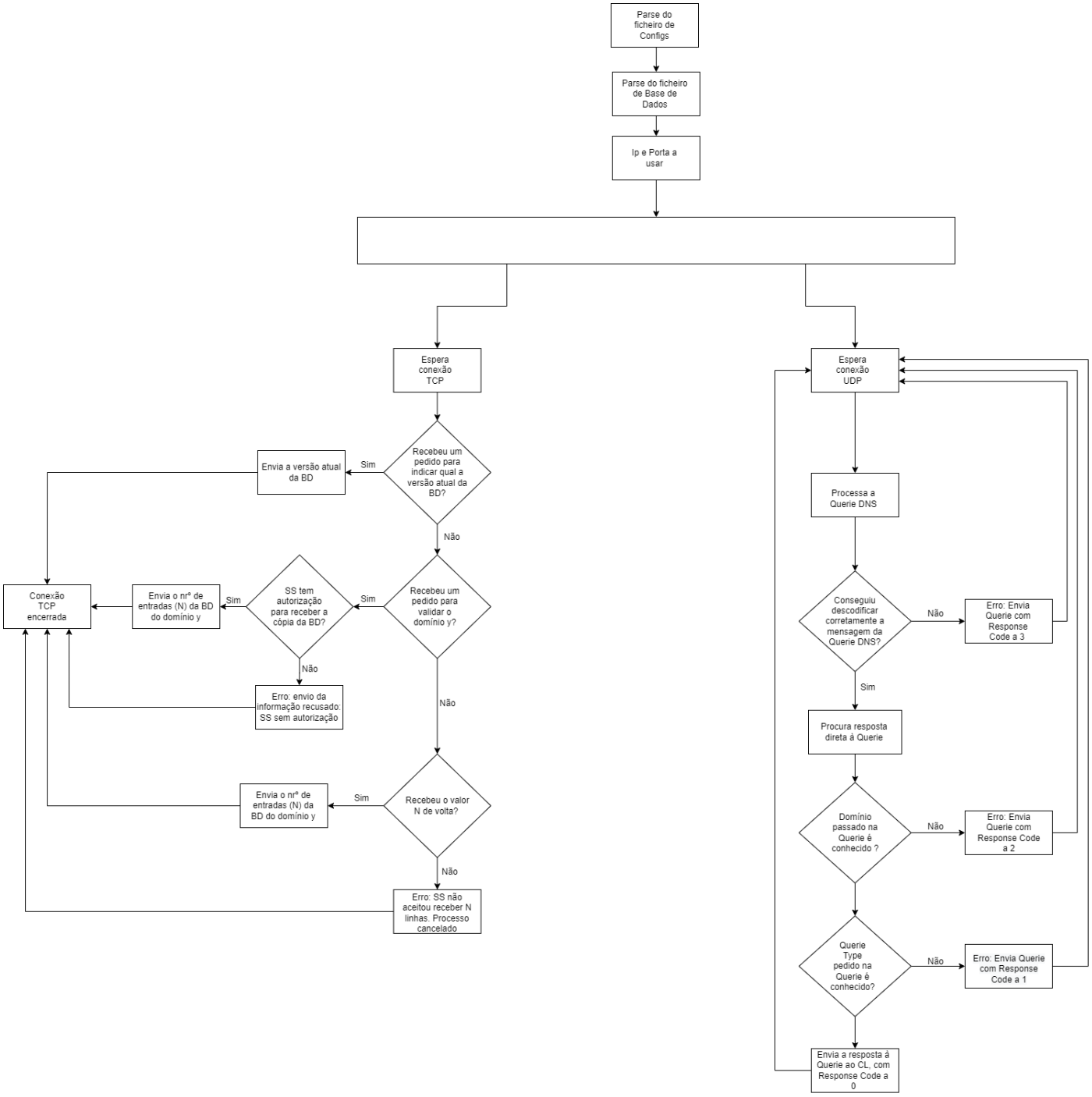

Figura 9: Algoritmo do servidor.

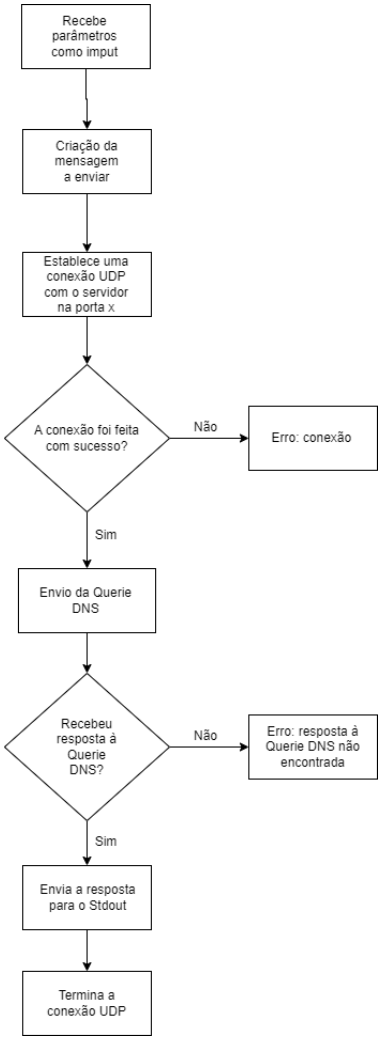

Figura 10: Algoritmo do cliente.

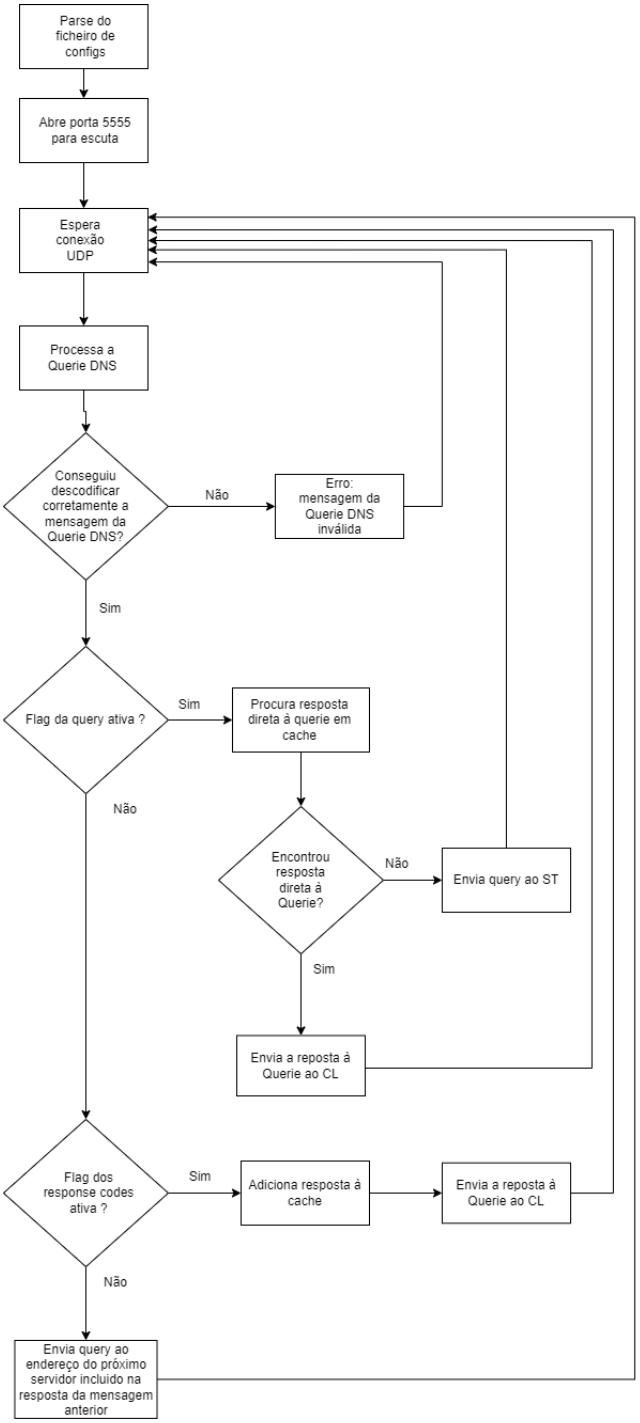

Figura 11: Algoritmo do Servidor de Resolução.

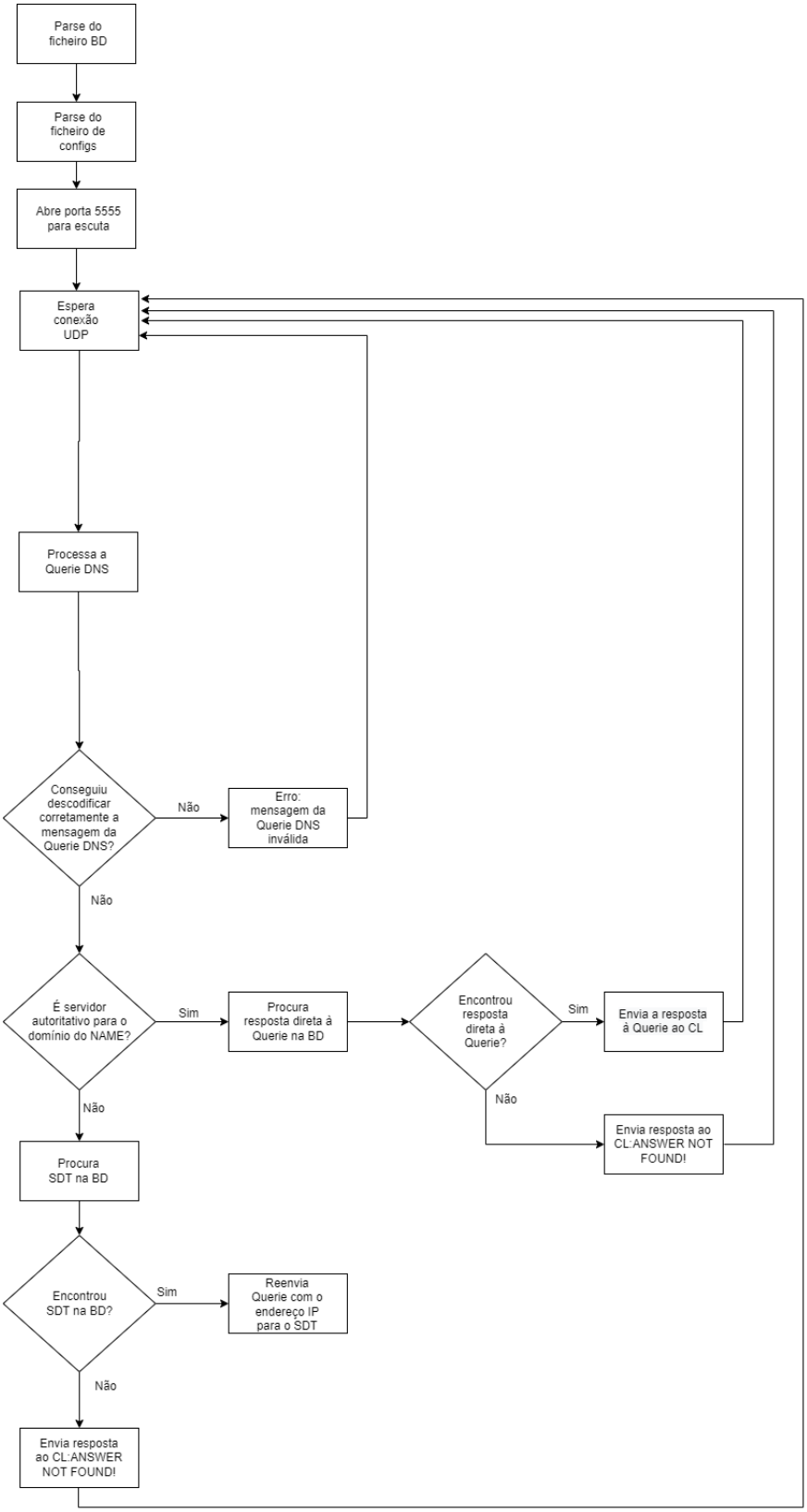

Figura 12: Algoritmo do Servidor de Topo.

5 Análise de testes

De forma a testar o comportamento dos elementos implementados no nosso ambiente de teste, foi realizado e analisado vários testes na nossa topologia e, através do uso da ferramenta *Wireshark*, confirmamos e validamos o tráfego gerado nestes testes.

Começamos primeiro por testar a transferência de zona entre um Servidor Primário e um Servidor Secundário. A figura 13 demonstra o output gerado pelo Servidor Primário e a figura 14 demonstra o output gerado pelo Servidor Secundário.

```
porta:5555
----------------------
TCP server up and listening
----------------------
Mensagem Recebida: 27187,Q,0,0,0,0;SOASERIAL?,;
----------------------
Mensagem Enviada: 27187,,0,0,0,0;,;0117102022,,
----------------------
TCP server up and listening
----------------------
Mensagem Recebida: 30739,Q,0,0,0,0;Dominio,Alvalade.com;
----------------------
Mensagem Enviada: 30739,,0,0,0,0;,;28,,
----------------------
TCP server up and listening
----------------------
Mensagem Recebida: 22329,Q,0,0,0,0;nrLinhas,28;
----------------------
Mensagem Enviada: 22329,,0,0,0,0;,;0;@ DEFAULT Alvalade.com.  ,,
----------------------
Mensagem Enviada: 22329,,0,0,0,0;,;1;TTL DEFAULT 86400  ,,
----------------------
Mensagem Enviada: 22329,,0,0,0,0;,;2;@ SOASP ns1.Alvalade.com. TTL ,,
----------------------
Mensagem Enviada: 22329,,0,0,0,0;,;3;@ SOAADMIN dns\.admin.Alvalade.com. TTL ,,
----------------------
Mensagem Enviada: 22329,,0,0,0,0;,;4;@ SOASERIAL 0117102022 TTL ,,
----------------------
Mensagem Enviada: 22329,,0,0,0,0;,;5;@ SOAREFRESH 20 TTL ,,
----------------------
Mensagem Enviada: 22329,,0,0,0,0;,;6;@ SOARETRY 10 TTL ,,
----------------------
Mensagem Enviada: 22329,,0,0,0,0;,;7;@ SOAEXPIRE 604800 TTL ,,
----------------------
Mensagem Enviada: 22329,,0,0,0,0;,;8;@ NS ns1.Alvalade.com. TTL ,,
----------------------
Mensagem Enviada: 22329,,0,0,0,0;,;9;@ NS ns2.Alvalade.com. TTL ,,
----------------------
Mensagem Enviada: 22329,,0,0,0,0;,;10;@ NS ns3.Alvalade.com. TTL ,,
----------------------
Mensagem Enviada: 22329,,0,0,0,0;,;11;@ NS ns4.Alvalade.com. TTL ,,
----------------------
Mensagem Enviada: 22329,,0,0,0,0;,;12;@ MX mx1.Alvalade.com TTL 10,,
----------------------
Mensagem Enviada: 22329,,0,0,0,0;,;13;@ MX mx2.Alvalade.com TTL 20,,
----------------------
Mensagem Enviada: 22329,,0,0,0,0;,;14;ns1.Alvalade.com. A 10.0.9.12 TTL ,,
----------------------
Mensagem Enviada: 22329,,0,0,0,0;,;15;ns2.Alvalade.com. A 10.0.9.11 TTL ,,
----------------------
Mensagem Enviada: 22329,,0,0,0,0;,;16;ns3.Alvalade.com. A 10.0.9.10 TTL ,,
----------------------
Mensagem Enviada: 22329,,0,0,0,0;,;17;ns4.Alvalade.com. A 10.0.9.12 TTL ,,
----------------------
Mensagem Enviada: 22329,,0,0,0,0;,;18;sp.Paulinho.Alvalade.com. A 10.0.12.13 TTL
 ,,
----------------------
Mensagem Enviada: 22329,,0,0,0,0;,;19;mx1.Alvalade.com. A 10.0.8.10 TTL ,,
----------------------
Mensagem Enviada: 22329,,0,0,0,0;,;20;mx2.Alvalade.com. A 10.0.8.11 TTL ,,
----------------------
Mensagem Enviada: 22329,,0,0,0,0;,;21;www.Alvalade.com. A 10.0.8.12 TTL ,,
----------------------
Mensagem Enviada: 22329,,0,0,0,0;,;22;sp CNAME ns1   ,,
----------------------
Mensagem Enviada: 22329,,0,0,0,0;,;23;ss1 CNAME ns2   ,,
----------------------
```

Figura 13: Output gerado pelo Servidor Primário relativo à transferência de zona.

```
< python3 SS.py files/Alvalade/configSSAlvalade.txt
----------------------
Mensagem Enviada: 27187,Q,0,0,0,0;SOASERIAL?,;
----------------------
----------------------
Mensagem Enviada: 30739,Q,0,0,0,0;Dominio,Alvalade.com;
----------------------
nr de linhas:28
----------------------
Mensagem Enviada: 22329,Q,0,0,0,0;nrLinhas,28;
----------------------
Recebi as linhas da BD e vou atualizar
----------------------
```

Figura 14: Output gerado pelo Servidor Secundário relativo à transferência de zona.

Como podemos observar em ambas figuras 13 e 14, a transferência de zona é efetuada com sucesso e os respeitos ficheiros de base de dados são atualizados e/ou criados no Servidor Secundário.

De maneira a validar o tráfego gerado por cada um dos módulos, utilizámos o *Wireshark* de maneira a capturar os pacotes TCP desta transferência de zona.

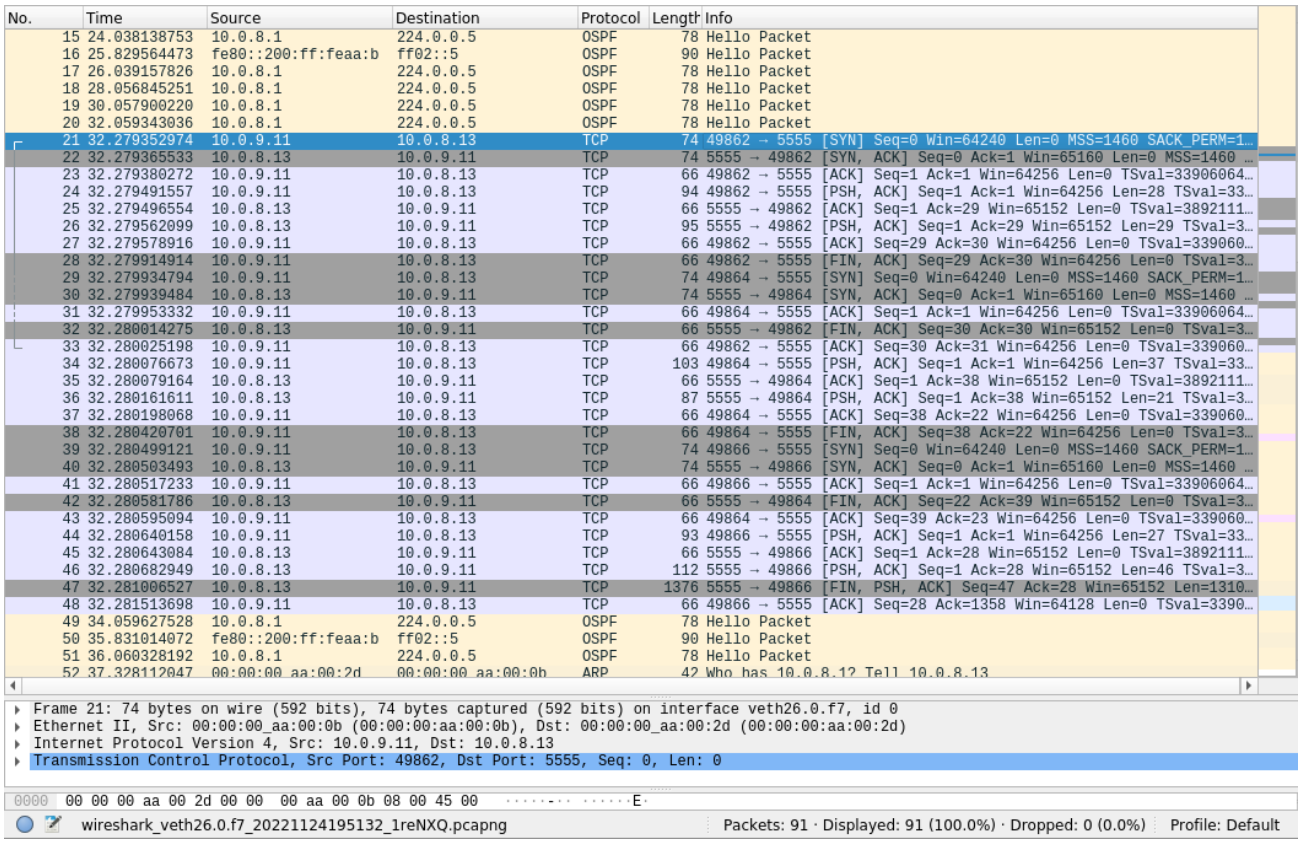

Figura 15: Captura de tráfego do wireshark relativo à transferência de zona.

Como podemos observar na figura 15, nos pacotes número 21 até 48, encontra-se o tráfego TCP relacionado à transferência de zona entre o Servidor Primário (endereço IP 10.0.8.13) e o Servidor Secundário (endereço IP 10.0.9.11).

De seguida, fizemos um teste semelhante para uma query DNS de maneira a observar

o tráfego UDP no *wireshark* e ainda para testar o comportamento do módulo. A figura 16 demonstra o output gerado pelo Cliente no terminal.

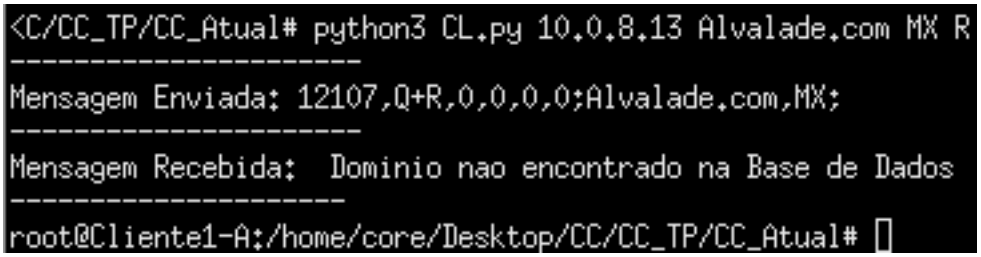

Figura 16: Output gerado pelo Cliente relativo à query DNS.

Analisando agora este tráfego no *wireshark*, obtivemos os seguintes resultados.

82	74.636437606	10.0.8.20	10.0.8.13	UDP	69 47265 → 5555 Len=27
83	74.636496985	10.0.8.13	10.0.8.20	UDP	82 5555 → 47265 Len=40

Figura 17: Captura de tráfego do wireshark relativo à query DNS.

Na figura 17 podemos observar o tráfego UDP relativo à query DNS realizada do cliente (endereço IP 10.0.8.20) ao Servidor Primário (10.0.8.13).

5.1 Fase 2

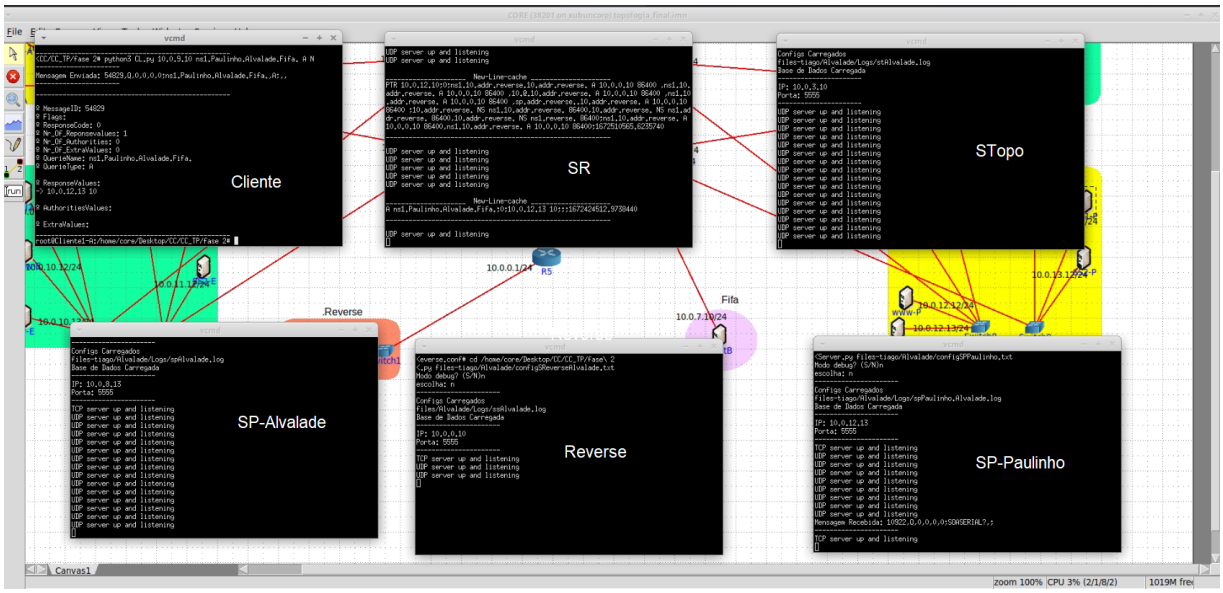

Figura 18: Teste modo iterativo

No.	Time	Source	Destination	Protocol	Length	Info
97	158.193825676	10.0.9.1	224.0.0.5	OSPF	78	Hello Packet
98	160.195279763	10.0.9.1	224.0.0.5	OSPF	78	Hello Packet
99	162.196274234	10.0.9.1	224.0.0.5	OSPF	78	Hello Packet
100	164.197675967	10.0.9.1	224.0.0.5	OSPF	78	Hello Packet
101	164.929417031	10.0.8.20	10.0.9.10	UDP	90	39468 → 5555 Len=48
102	164.930238062	10.0.9.10	10.0.3.10	UDP	90	36991 → 5555 Len=48
103	164.982770885	10.0.3.10	10.0.9.10	UDP	189	60696 → 5555 Len=147
104	164.983976692	10.0.9.10	10.0.8.13	UDP	90	37211 → 5555 Len=48
105	164.984505318	10.0.8.13	10.0.9.10	UDP	392	34407 → 5555 Len=350
106	164.984940812	10.0.9.10	10.0.12.13	UDP	90	60581 → 5555 Len=48
107	164.985348099	10.0.12.13	10.0.9.10	UDP	102	44395 → 5555 Len=60
108	164.985572227	10.0.9.10	10.0.8.20	UDP	102	50199 → 39468 Len=60
109	166.198328687	10.0.9.1	224.0.0.5	OSPF	78	Hello Packet

```
Frame 101: 90 bytes on wire (720 bits), 90 bytes captured (720 bits) on interface veth1.3.ac, id 0
Ethernet II, Src: 00:00:00_aa:00:14 (00:00:00:aa:00:14), Dst: 00:00:00_aa:00:2e (00:00:00:aa:00:2e)
Internet Protocol Version 4, Src: 10.0.8.20, Dst: 10.0.9.10
User Datagram Protocol, Src Port: 39468, Dst Port: 5555
Data (48 bytes)
```

Figura 19: Captura de tráfego do wireshark relativo à query A.

6 Atividades desenvolvidas

Atividades Desenvolvidas	**Tiago**	**Hugo**	**Bernardo**
Desenvolvimento A1	100	0	0
Desenvolvimento A2	0	100	0
Desenvolvimento A3	0	0	100
Relatório	34	33	33
Implementação dos parsers e estruturas de dados	33	33	34
Implementação dos Cliente/Servidor	33	34	33
Implementação do Modo Iterativo	34	33	33
Implementação da Cache	33	33	34
Implementação da Zona Reversa	33	34	33

7 Conclusão

Após a conclusão da segunda fase do trabalho prático, e consequentemente da componente prática da disciplina , consideramos que o trabalho desenvolvido foi positivo, tendo em conta os objetivos propostos no enunciado . Na segunda fase , surgiram alguns problemas que tiveram de ser resolvidos ,que já tinham sido notados na primeira fase .Após a correção de pequenos erros, principalmente relativos às respostas a Queries , a maior dificuldade foi a implementação do Servidor de resolução que tinha de perceber respostas e supostamente direcionar para o devido sítio a resposta.

De notar ainda que não foi implementada uma codificação normal binária ,pois o grupo optou por dar prioridade a outros aspetos mais relevantes.

No geral , este trabalho, ajudou bastante a perceber o modo de funcionamento do DNS , e após a sua implementação desde raiz , conseguimos perceber a interação entre os componentes do mesmo , bem como a importância de cada um deles no funcionamento deste Serviço.

Referências

[1] Nome da organização: geeksforgeeks

Disponível em: https://www.geeksforgeeks.org/python-programming-language/?ref=shm

Data de acesso: Durante as várias fases do projeto

[2] Nome da organização: pythontic

Disponível em: https://pythontic.com/modules/socket/udp-client-server-example

Data de acesso: Durante a construção do socket UDP

Anexos

Anexo 1 - Ficheiro de Base de Dados do domínio Alvalade

```
# DNS database file for domain Alvalade.Fifa
# It also includes a pointer to the primary server
# of the Paulinho.Alvalade.Fifa subdomain

@ DEFAULT Alvalade.Fifa.
TTL DEFAULT 86400

@ SOASP ns1.Alvalade.Fifa. TTL
@ SOAADMIN dns\.admin.Alvalade.Fifa. TTL
@ SOASERIAL 0117102022 TTL
@ SOAREFRESH 14400 TTL
@ SOARETRY 3600 TTL
@ SOAEXPIRE 604800 TTL

@ NS ns1.Alvalade.Fifa. TTL
@ NS ns2.Alvalade.Fifa. TTL
@ NS ns3.Alvalade.Fifa. TTL
@ NS ns4.Alvalade.Fifa. TTL

Paulinho.@ NS sp.Paulinho.Alvalade.Fifa.

@ MX mx1.Alvalade.Fifa. TTL 10
@ MX mx2.Alvalade.Fifa. TTL 20

ns1 A 10.0.8.13 TTL
ns2 A 10.0.9.11 TTL
ns3 A 10.0.9.10 TTL
ns4 A 10.0.9.12 TTL
sp.Paulinho A 10.0.12.13 TTL
ss1.Paulinho A 10.0.13.11 TTL
ss2.Paulinho A 10.0.13.12 TTL
mx1 A 10.0.8.10 TTL
mx2 A 10.0.8.11 TTL
www A 10.0.8.12 TTL

sp CNAME ns1 TTL
ss1 CNAME ns2 TTL
ss2 CNAME ns3 TTL
sr CNAME ns4 TTL
mail1 CNAME mx1 TTL
mail2 CNAME mx2 TTL
```

Anexo 2 - Ficheiro de Base de Dados do domínio Luz

```
# DNS database file for domain Luz.Fifa
# It also includes a pointer to the primary server
```

```
# of the Enzo.Luz.Fifa subdomain

@ DEFAULT Luz.Fifa.
TTL DEFAULT 86400

@ SOASP ns1.Luz.Fifa. TTL
@ SOAADMIN dns\.admin.Luz.Fifa. TTL
@ SOASERIAL 0117102022 TTL
@ SOAREFRESH 14400 TTL
@ SOARETRY 3600 TTL
@ SOAEXPIRE 604800 TTL

@ NS ns1.Luz.Fifa. TTL
@ NS ns2.Luz.Fifa. TTL
@ NS ns3.Luz.Fifa. TTL
@ NS ns4.Luz.Fifa. TTL

Enzo.@ NS sp.Enzo.Luz.Fifa.

@ MX mx1.Luz.Fifa. TTL 10
@ MX mx2.Luz.Fifa. TTL 20

ns1 A 10.0.14.13 TTL
ns2 A 10.0.15.11 TTL
ns3 A 10.0.15.12 TTL
ns4 A 10.0.15.10 TTL
sp.Enzo A 10.0.10.13 TTL
mx1 A 10.0.14.10 TTL
mx2 A 10.0.14.11 TTL
www A 10.0.14.12 TTL

sp CNAME ns1 TTL
ss1 CNAME ns2 TTL
ss2 CNAME ns3 TTL
sr CNAME ns4 TTL
mail1 CNAME mx1 TTL
mail2 CNAME mx2 TTL
```

Anexo 3 - Ficheiro de Base de Dados do subdomínio Paulinho.Alvalade

```
# DNS database file for subdomain Paulinho.Alvalade.Fifa

@ DEFAULT Paulinho.Alvalade.Fifa.
TTL DEFAULT 86400

@ SOASP ns1.Paulinho.Alvalade.Fifa. TTL
@ SOAADMIN dns\.admin.Paulinho.Alvalade.Fifa. TTL
@ SOASERIAL 0117102022 TTL
@ SOAREFRESH 14400 TTL
@ SOARETRY 3600 TTL
@ SOAEXPIRE 604800 TTL
```

```
@ NS ns1.Paulinho.Alvalade.Fifa. TTL
@ NS ns2.Paulinho.Alvalade.Fifa. TTL
@ NS ns3.Paulinho.Alvalade.Fifa. TTL
@ NS ns4.Paulinho.Alvalade.Fifa. TTL


@ MX mx1.Paulinho.Alvalade.Fifa. TTL 10
@ MX mx2.Paulinho.Alvalade.Fifa. TTL 20

ns1 A 10.0.12.13 TTL
ns2 A 10.0.13.11 TTL
ns3 A 10.0.13.12 TTL
ns4 A 10.0.13.10 TTL
mx1 A 10.0.12.10 TTL
mx2 A 10.0.12.11 TTL
www A 10.0.12.12 TTL

sp CNAME ns1 TTL
ss1 CNAME ns2 TTL
ss2 CNAME ns3 TTL
sr CNAME ns4 TTL
mail1 CNAME mx1 TTL
mail2 CNAME mx2 TTL
```

Anexo 4 - Ficheiro de Base de Dados do subdomínio Enzo.Luz

```
# DNS database file for subdomain Enzo.Luz.Fifa

@ DEFAULT Enzo.Luz.Fifa.
TTL DEFAULT 86400

@ SOASP ns1.Enzo.Luz.Fifa. TTL
@ SOAADMIN dns\.admin.Enzo.Luz.Fifa. TTL
@ SOASERIAL 0117102022 TTL
@ SOAREFRESH 14400 TTL
@ SOARETRY 3600 TTL
@ SOAEXPIRE 604800 TTL

@ NS ns1.Enzo.Luz.Fifa. TTL
@ NS ns2.Enzo.Luz.Fifa. TTL
@ NS ns3.Enzo.Luz.Fifa. TTL
@ NS ns4.Enzo.Luz.Fifa. TTL

@ MX mx1.Enzo.Luz.Fifa. TTL 10
@ MX mx2.Enzo.Luz.Fifa. TTL 20

ns1 A 10.0.10.13 TTL
ns2 A 10.0.11.11 TTL
ns3 A 10.0.11.12 TTL
ns4 A 10.0.11.10 TTL
```

```
mx1 A 10.0.10.10 TTL
mx2 A 10.0.10.11 TTL
www A 10.0.10.12 TTL

sp CNAME ns1 TTL
ss1 CNAME ns2 TTL
ss2 CNAME ns3 TTL
sr CNAME ns4 TTL
mail1 CNAME mx1 TTL
mail2 CNAME mx2 TTL
```

Anexo 5- Ficheiro de Base de Dados do domínio .Fifa.

```
# DNS database file for domain Fifa
# It also includes a pointer to the primary server
# of the Alvalade.Fifa subdomain

@ DEFAULT Fifa.
TTL DEFAULT 86400

@ SOASP ns1.Fifa. TTL
@ SOAADMIN dns\.admin.Fifa. TTL
@ SOASERIAL 20 TTL
@ SOAREFRESH 10 TTL
@ SOARETRY 10 TTL
@ SOAEXPIRE 10 TTL

@ NS ns1.Fifa. TTL
@ NS ns2.Fifa. TTL

Alvalade.@ NS sp.Alvalade.Fifa.
Luz.@ NS sp.Luz.Fifa.


ns1 A 10.0.3.10 TTL
ns2 A 10.0.7.10 TTL

sp.Alvalade A 10.0.8.13 TTL
sp.Luz A 10.0.14.13 TTL

reverse NS sp.reverse TTL
sp.reverse A 10.0.0.10 TTL

sp1 CNAME ns1 TTL
sp2 CNAME ns2 TTL
```