# Comunicações Por Computador

Trabalho Prático 2

Grupo PL6-03:

Uma imagem com céu, exterior, pessoa, terra

Descrição gerada automaticamente Uma imagem com pessoa, exterior, céu, homem

Descrição gerada automaticamente

A95460

A96726

Ivo Miguel Alves Ribeiro A96726

Diogo Luís Almeida Costa A95460

Tópicos:

-introdução ao trabalho

- decisões de distribuição de trabalho

-estrategias a seguir

- arquitetura do sistema

- funcionalidades de cada elemento

- explicar o metodo de comunicaçao entre eles

- modelo comunicativo

- explcar as PDU criadas

- formas de intereçao possiveis entre elementos da arquitetura

- como o sistema lida com as situaçoes de erro

- implementaçao de cada elemento

- opçoes de execuçao e as funcionalidas dos seu scomponentes

- quais os requisitos funcionais enumerados no enunciado são implementados e como

- anlise dos testes de sistema e verificaçao do funcionamento de todos os elementos num amabiente de teste pre definido

- explicar o fincionamnetos dos componentes com exemplos de testes

- fazer um ficheiro proprio apenas com os resultados detalhados das experiencias executadas no ambiente de teste, assim como especificaçao exaustiva do proprio ambinete de teste

-avaliaçao do grupo

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Construçao do ambiente de teste | Relatorio |  |  |  |  |  |  |  |
| Ivo | 50% | 30% |  |  |  |  |  |  |  |
| Diogo | 50% | 70% |  |  |  |  |  |  |  |

- conclusao

**INTRODUÇÃO AO TRABALHO**

**ARQUITETURA DO SISTEMA**

**TOPOLOGIA**

Na arquitetura do sistema, acabamos por construir uma topologia com um dominio de topo, dois dominios(.bacalhau;.robalo), dois subdominios(batata.bacalhau;arroz.robalo) e um dominio reverse.

//imagens

Na nossa topologia, o dominio de topo tem dois servidores de topo, nos quais estes são ambos primarios. Para os dois dominios(.bacalhau;.robalo), estes são compostos por um servidor primario, dois servidores secundarios, um servidor de resposta, dois mxs, um webserver e um cliente. Para os dois subdominios(batata.bacalhau;arroz.robalo) estes são composto de igual maneira aos dois dominios referidos anteriormente. Para finalizar, o dominio reverse é constituido por um servidor primario.

**COMPONENTES DO SISTEMA**

**SERVIDOR PRIMARIO(SP)**

O nosso modelo de servidor primario é um servidor DNS que irá responder e efetuar queries DNS, para alem disso estes serao a autoridade so seu proprio dominio DNS e por consequente irão ter acesso direto á base de dados do dominio. Qualquer informaçao importante que deva ser adicionada ou alterada num dominio DNS terá de ser feita diretamente na base de dados do SP. Aditivamente, para alem dos dados relativos ao seu dominio, o SP terá acesso a uma configuraçao especifica na qual sabe todos os dominios para qual é SP, todas as portas de atendimento, identificaçao do ficheiro da base de dados, identificaçao do ficheiro de log, informaçao de segurança para acesso ás bases de dados, identificaçao dos SS respetivos e dos SP dos ssubdominios e endereços dos servidores de topo. Para este projeto, cada SP terá um imput de um ficheiro de configuraçao, um ficheiro de base de dados para cada dominio gerido e um ficheiro com a lista de servidores de topo.Como output terá apenas um ficheiro log.

**SERVIDOR SECUNDARIO(SS)**

O nosso modelo de servidor secundario será um servidor DNS que irá reponder e efetuar queries DNS, este não terá acesso direto á base de dados, mas terá autoridade para ter uma replica da base de dados original do SP do seu dominio, e esta poderá ser atualizada quando necessario. Para alem disso, o SS tem de ter acesso a uma configuraçao especifica na qual este terá a informaçao dos dominios para qual o servidor é secundario, das porta de atendiento, da identificaçao do SP dos dominios para qual é SS, da identificaçao do ficheiro de log, da informaçao de segurança para acesso ao SP, dos endereços de servidores se topo. Para este projeto, cada SS terá um imput de um ficheiro de configuraçao e um ficheiro com a lista de servidores de topo.Como output terá apenas um ficheiro log. Aditivamente, neste projeto foi decidido que a informaçao da base de dados replicada do SP será armazenada apenas em memoria volatil.

**SERVIDOR DE RESPOSTA(SR)**

O nosso modelo de servidor de resposta será um servidor DNS que irá responder e efetuar queries DNS sobre qualquer dominio, este servirá apenas de intermediario, pois este não ira ter autoridade sobre nenhum dominio. Um SR poderá ser mplementado a muitos niveis da rede, desde um processo a cada aplicaçao cliente até um servidor DNS que responde aos clientes duma rede IP local, aos clientes dos provedores de serviçoes ou aos clientes de uma instituiçao.Um SR terá acesso a uma configuraçao especifica na qual irá ter a informaçao dos dominios e lista de servidores DNS que deve contactar, das portas de atendimentos, da indentificaçao do ficheiro lod, dos endereços dos servidores de topo. Para este projeto, cada SR terá um imput de um ficheiro de configuraçao e um ficheiro com a lista de servidores de topo.Como output terá apenas um ficheiro log.

**CLIENTE(CL)**

O nosso modelo de cliente será um processo que vai precisar da informaçao da base de dados de DNS dum determinado dominio. Para obter essa informaçao, irá realizar queries DNS a um SR(para já na 2º fase irá perguntar diretamente a um SP ou SS sobre a base de dados do dominio na qual pretenderá obter informaçoes). Habitualmente, um CL teria um ficheiro de configuraçao com uma lista de SR, mas, para este projeto, cada cliente terá o imput e o output atraves da linha de comando, sem necessidade de ficheiro de configuraçao. ´

**SERVIDORES DE DOMINIOS DE TOPO(SDT)**

O nosso modelo de serviodor de dominio de topo é basicamente igual aos SP ou aos SS (ou seja, um SP ou SS autoritativos para um domínio de topo é um SDT) ainda que estes não tenham domínios hierarquicamente acima na árvore DNS.

**SERVIDORES DE TOPO(ST)**

O nosso modelo de servidor de topo será como um SP, mas terão apenas uma base de dados para cada dominio de topo, onde inclui informação dos SDT respetivos ( os nomes e os endereços IP dos seus SS e do seu SP).

**FICHEIROS DO SISTEMA**

Neste projeto foram definidos alguns ficheiros de configuração, de dados e de log com uma sintaxe predefinida. Os ficheiros de configuração serão apenas lidos e processados no início do componente de software a que dizem respeito e moldam o seu comportamento. Os ficheiros de dados também são consultados apenas no início e a sua informação deverá ter de ficar armazenada em memória. Para alterar ou atualizar o comportamento dos servidores com informação modificada nos ficheiros de configuração será necessário reiniciar os servidores.

**FICHEIROS DE CONFIGURAÇAO DOS SERVIDORES SP, SS E SR**

Estes ficheiros terão uma sintaxe onde terão de respeitar algumas regras:

* As linhas começadas por ‘#’ são consideradas comentários e são ignoradas
* As linhas em branco também devem ser ignoradas;
* Deve existir uma definição de parâmetro de configuração por cada linha seguindo esta sintaxe:

{parâmetro} {tipo do valor} {valor associado ao parâmetro}

Nesta sintaxe os tipos de valores aceites no ficheiros de configuraçao serao:

* DB - o valor indica o ficheiro da base de dados com a informação do domínio indicado no parâmetro (o servidor assume o papel de SP para este domínio);
* SP - o valor indica o endereço IP[:porta] do SP do domínio indicado no parâmetro (o servidor assume o papel de SS para este domínio);
* SS – o valor indica o endereço IP[:porta] dum SS do domínio indicado no parâmetro (o servidor assume o papel de SP para este domínio) e que passa a ter autorização para pedir a transmissão da informação da base de dados (transferência de zona); podem existir várias entradas para o mesmo parâmetro (uma por cada SS do domínio);
* DD – o valor indica o endereço IP[:porta] dum SR, dum SS ou dum SP do domínio por defeito indicado no parâmetro; quando os servidores que assumem o papel de SR usam este parâmetro é para indicar quais os domínios para os quais devem contactar diretamente os servidores indicados se receberem queries sobre estes, em vez de contactarem um dos ST; podem existir várias entradas para o mesmo parâmetro (uma por cada servidor do domínio por defeito); Normalmente, quando os servidores que assumem o papel de SP ou SS usam este parâmetro é para indicar os únicos domínios para os quais respondem, mas nestes casos, o parâmetro serve para restringir o funcionamento dos SP ou SS a responderem apenas a queries sobre os domínios indicados neste parâmetro;
* ST – o valor indica o ficheiro com a lista dos ST (o parâmetro deve ser igual a “root”);~
* LG – o valor indica o ficheiro de log que o servidor deve utilizar para registar a atividade do servidor associada ao domínio indicado no parâmetro; só podem ser indicados domínios para o qual o servidor é SP ou SS; tem de existir pelo menos uma entrada a referir um ficheiro de log para toda a atividade que não seja diretamente referente aos domínios especificados noutras entradas LG (neste caso o parâmetro deve ser igual a “all”).

//IMAGES Exemplo de um dos nosso ficheiros de configuração

**FICHEIRO DE LOG**

Os ficheiros de log irão registar toda a atividade relevante do componente, deve sempre existir uma entrada de log por cada linha do ficheiro. Para alem disso, sempre que um componente arranca deve verificar a existencia dos ficheiros log indicados no seu ficheiro de configuraçao. Se não existir algum deve ser criado e se já existirem as novas entradas devem ser registadas a partir da última entrada já existente no ficheiro. A sintaxe de cada entrada é:

{etiqueta temporal} {tipo de entrada} {endereço IP[:porta]} {dados da entrada}

Tipos de entradas aceites:

* QR/QE – foi recebida/enviada uma query do/para o endereço indicado; os dados da entrada devem ser os dados relevantes incluídos na query; a sintaxe dos dados de entrada é a mesma que é usada no PDU de query no modo debug de comunicação entre os elementos;
* RP/RR – foi enviada/recebida uma resposta a uma query para o/do endereço indicado; os dados da entrada devem ser os dados relevantes incluídos na resposta à query; a sintaxe dos dados de entrada é a mesma que é usada no PDU de resposta às queries no modo debug de comunicação entre os elementos;
* ZT – foi iniciado e concluído corretamente um processo de transferência de zona; o endereço deve indicar o servidor na outra ponta da transferência; os dados da entrada devem indicar qual o papel do servidor local na transferência (SP ou SS) e, opcionalmente, a duração em milissegundos da transferência e o total de bytes transferidos;
* EV – foi detetado um evento/atividade interna no componente; o endereço deve indicar 127.0.0.1 (ou localhost ou @); os dados da entrada devem incluir informação adicional sobre a atividade reportada (por exemplo, ficheiro de configuração/dados/ST lido, criado ficheiro de log, etc.);
* ER – foi recebido um PDU do endereço indicado que não foi possível descodificar corretamente; opcionalmente, os dados da entrada podem ser usados para indicar informação adicional (como, por exemplo, o que foi possível descodificar corretamente e em que parte/byte aconteceu o erro);
* EZ – foi detetado um erro num processo de transferência de zona que não foi concluída corretamente; o endereço deve indicar o servidor na outra ponta da transferência; os dados da entrada devem indicar qual o papel do servidor local na transferência (SP ou SS);
* FL – foi detetado um erro no funcionamento interno do componente; o endereço deve indicar 127.0.0.1; os dados da entrada devem incluir informação adicional sobre a situação de erro (por exemplo, um erro na descodificação ou incoerência dos parâmetros de algum ficheiro de configuração ou de base de dados);
* TO – foi detetado um timeout na interação com o servidor no endereço indicado; os dados da entrada devem especificar que tipo de timeout ocorreu (resposta a uma query ou tentativa de contato com um SP para saber informações sobre a versão da base de dados ou para iniciar uma transferência de zona);
* SP – a execução do componente foi parada; o endereço deve indicar 127.0.0.1; os dados da entrada devem incluir informação adicional sobre a razão da paragem se for possível obtê-la;
* ST – a execução do componente foi iniciada; o endereço deve indicar 127.0.0.1; os dados da entrada devem incluir informação sobre a porta de atendimento, sobre o valor do timeout usado (em milissegundos) e sobre o modo de funcionamento (modo “shy” ou modo debug).

//images

**FICHEIRO DE DADOS DO SP**

O ficheiro da base de dados do SP terá uma sintaxe com as seguintes regras:

* As linhas começadas por ‘#’ são consideradas comentários e são ignoradas;
* As linhas em branco também devem ser ignoradas;
* Deve existir uma definição de parâmetro de dados por cada linha seguindo esta sintaxe:

{parâmetro} {tipo do valor} {valor} {tempo de validade} {prioridade}

O tempo de validade(TTL) representa o tempo maximo em segundos que os dados podem existir numa cache. Quando um determinado tipo não consegue suportar o TTL o seu valor será zero.

O campo de prioridade é um valor inteiro menor que 256 que define uma ordem de prioridade para respetivos parametros onde quanto maior for o numero menor irá ser a prioridade. Para parametros de valor único ou com prioridade igual este campo não deverá existir. Este campo irá ajudar em algumas situaçoes de backups ou de balanceamento de carga de serviços.

Os nomes completos de e-mails, dominios, servidores e hosts devem terminar com um ‘.’(como exemplo: example.com.). Quando os nomes não terminam com ‘.’ Subentende se que estes estao concatenados por um prefixo defenido atraves de um parametro @ do tipo DEFAULT

* DEFAULT\* – define um nome (ou um conjunto de um ou mais símbolos) como uma macro que deve ser substituída pelo valor literal associado (não pode conter espaços nem o valor dum qualquer parâmetro DEFAULT); o parâmetro @ é reservado para identificar um prefixo por defeito que é acrescentado sempre que um nome não apareça na forma completa (termina com ‘.’); o valor de TTL deve ser zero;
* SOASP – o valor indica o nome completo do SP do domínio (ou zona) indicado no parâmetro;
* SOAADMIN – o valor indica o endereço de e-mail completo do administrador do domínio (ou zona) indicado no parâmetro; o símbolo ‘@’ deve ser substituído por ‘.’ e ‘.’ no lado esquerdo do ‘@’ devem ser antecedidos de ‘\’;
* SOASERIAL – o valor indica o número de série da base de dados do SP do domínio (ou zona) indicado no parâmetro; sempres que a base de dados é alterada este número deve ser incrementado;
* SOAREFRESH – o valor indica o intervalo temporal em segundos para um SS perguntar ao SP do domínio indicado no parâmetro qual o número de série da base de dados dessa zona;
* SOARETRY – o valor indica o intervalo temporal para um SS voltar a perguntar ao SP do domínio indicado no parâmetro qual o número de série da base de dados dessa zona, após um timeout;
* SOAEXPIRE – o valor indica o intervalo temporal para um SS deixar de considerar a sua réplica da base de dados da zona indicada no parâmetro como válida, deixando de responder a perguntas sobre a zona em causa, mesmo que continue a tentar contactar o SP respetivo;
* NS – o valor indica o nome dum servidor que é autoritativo para o domínio indicado no parâmetro, ou seja, o nome do SP ou dum dos SS do domínio; este tipo de parâmetro suporta prioridades;
* A – o valor indica o endereço IPv4 dum host/servidor indicado no parâmetro como nome; este tipo de parâmetro suporta prioridades;
* CNAME – o valor indica um nome canónico associado ao nome indicado no parâmetro; um nome canónico não deve apontar para um outro nome canónico nem podem existir outros parâmetros com o mesmo valor do nome canónico;
* MX – o valor indica o nome dum servidor de e-mail para o domínio indicado no parâmetro; este tipo de parâmetro suporta prioridades;
* PTR – o valor indica o nome dum servidor/host que usa o endereço IPv4 indicado no parâmetro; a indicação do IPv4 é feita como nos domínios de DNS reverso (rDNS) quando se implementa reverse-mapping;

//IMAGE

**MODELO COMUNICATIVO**

Na terceira secção do relatório vamos detalhar o modelo comunicativo, explicando os PDU criados e todas as formas de interação possíveis entre os elementos da arquitetura. Além disso, vamos tentar indicar todas as situações de erro e como o sistema lida com elas.

**Modelo Comunicacional do Sistema**

O modelo comunicacional é baseado em interaçoes assíncronas feitas através de mensagens encapsuladas no protocolo UDP. Esta mensagem DNS terá um cabeçalho fixo e os dados(este ocupam ate 1 kbyte). Os dados estão divididos em quatro partes:

* Os dados da query original
* Os resultados dessa query
* Informaçao dos servidores com autoridade sobre os dados da resposta
* Informaçao adicional indiretamente ligada aos resultados dessa query

//IMAGE FORMATO MENSAGEM

Esta mensagem assincrona será responsavel pelo transporte da query do cliente para os servidores e pelo transporte da resposta da query para o cliente. Esta mensagens está dividida em varios campos, terão de ser implementados da seguinte maneira:

* MESSAGE ID – identificador de mensagem que irá ser usado para relacionar as respostas recebidas com a query original;
* FLAGS – devem ser suportadas as flags Q, R e A; a flag Q ativa indica que a mensagem é uma query, senão é uma resposta a uma query; se a flag R estiver ativa na query indica que se deseja que o processo opere de forma recursiva e não iterativa; se a flag R estiver ativa na resposta indica que o servidor que respondeu suporta o modo recursivo; se a flag A estiver ativa na resposta indica que a resposta é autoritativa ;
* RESPONSE CODE – indica o código de erro na resposta a uma query; se o valor for zero então não existe qualquer tipo de erro e a resposta contém informação que responde diretamente à query; essa resposta deve ser guardada em cache; se houver erros, o sistema deve suportar os seguintes códigos de erro: 1, o domínio incluído em NAME existe mas não foi encontrada qualquer informação direta com um tipo de valor igual a TYPE OF VALUE; este caso é identificado como resposta negativa e pode ser guardada em cache; 2, o domínio incluído em NAME não; este caso também é identificado como resposta negativa e pode ser guardada em cache; 3, a mensagem DNS não foi descodificada corretamente;
* NUMBER OF VALUES – número de entradas relevantes que respondem diretamente à query e que fazem parte da lista de entradas incluídas no campo RESPONSE VALUES;
* NUMBER OF AUTHORITIES – número de entradas que identificam os servidores autoritativos para o domínio incluído no RESULT VALUES;
* NUMBER OF EXTRA VALUES – número de entradas com informação adicional relacionada com os resultados da query ou com os servidores da lista de autoridades;
* QUERY INFO – informação do parâmetro da query (NAME) e o tipo de valor associado ao parâmetro (TYPE OF VALUE); os tipos suportados são os mesmos suportados na sintaxe dos ficheiros de base de dados dos SP; na resposta a queries, os servidores devem copiar a informação do QUERY INFO e incluí-la na mensagem de resposta;
* RESPONSE VALUES – lista das entradas que fazem match no NAME e TYPE OF VALUE incluídos na cache ou na base de dados do servidor autoritativo; cada entrada deve ter a informação completa tal como é definida na base de dados DNS do SP do domínio referente ao NAME;
* AUTHORITIES VALUES – lista das entradas que fazem match com o NAME e com o tipo de valor igual a NS incluídos na cache ou na base de dados do servidor autoritativo; cada entrada deve ter a informação completa tal como é definida na base de dados DNS do SP do domínio referente ao NAME;
* EXTRA VALUES – lista das entradas do tipo A e que fazem match no parâmetro com todos os valores no campo RESPONSE VALUES e no campo AUTHORITIES VALUES de forma a que o elemento que o CL ou servidor que recebe a resposta não tenha que fazer novas queries para saber os endereços IP dos parâmetros que vêm como valores nos outros dois campos; cada entrada deve ter a informação completa tal como é definida na base de dados DNS do SP do domínio referente ao NAME.

Segue se entao o exemplo da interaçao assincrona de um CL e um SP:

//IMAGE FIG 4 ENUNCUADO

De seguida, encontra se a resposta do SP ao CL:

//IMAGEM FIG6 ENUNCIADO

//Codificaçao binaria das menagens--------------------------------------------------------------------------

Neste trabalho, todas as interaçoes começam com o envio de uma query, como já sabemos, essa query vem de um CL ou de um Servidor DNS.Em baixo, segue se o exemplo de todos os passos desde o envio de uma query de um cliente para um servidor SP ou SP:

//IMAGEM PAINT

Estes passos passam resumidamente por o cliente enviar a query para o SP, o servidor descodificar corretamente a query, de seguida este irá primeiro aceder á sua cache para procurar a informaçao necessaria para a resoluçao da query. Se este não for bem sucedido, este irá procurar na sua base de dados, se este for bem sucedido irá retornar a query resolvida para o cliente com todos os campos devidamente preenchidos.

O mesmo vai acontecer com o SS, a única diferença será a ocorrencia da transferencia de zona caso o SS não tenha a copia atualizada da base de dados do SP.

**TRANSFERENCIA DE ZONA**

Todas as interaçoes numa operaçao de transferencia de zona deverao ser feitas utilizando uma conexao TCP. O SS vai tentar a transferencia de zona se a base de dados do respetivo SP for mais atual que a sua copia. A transferencia ocorrerá com o SS a mandar o nome do dominio da qual quer receber a base de dados. De seguida, o SP vai verificar a validade desse dominio e se SS tem autorizaçao para receber a copia da sua base de dados. Se isto acontecer, o SP vai mandar o numero de entradas do ficheiro da base de dados primeiro, depois o SS vai verificar o numero de linhas de entrada que vai receber. Quando o SS verificar e aceitar esse numero, o SP vai mandar todas as entradas em formato de texto, em ordem crescente, uma de cada vez, enquanto isso, o SS vai recebendo as linhas e vai verificar se recebeu as entradas esperadas. Quando o tempo predefenido da transferencia acabar, a conexao TCP vai terminar e com isso a transferencia tambem. Um SP também não deve aceitar pedidos consecutivos de transferência de zona do mesmo SS com intervalo menor de SOARETRY segundos.

// imagem tranferencia de zona

**ESTRATEGIAS DE IMPLEMENTAÇAO**

O nosso trabalho irá ser composto por 6 classes, nomeadamente a class ServerP, Server S, Cliente, Query, Cache e Logs. De seguida, irei explicar cada uma das suas funcionalidades e opções de execução.

**ServerP**

Na classe ServerP, temos como objetivo a implementação de um servidor do tipo primário. Este servidor ao inicio ira incializar duas classes(cache e log), depois irá receber um ficheiro de configuraçao ”teste.txt” no qual vai guardar a sua informação nas variáveis locais com a ajuda da função ParserSP.De seguida, vai buscar o nome do ficheiro da sua base de dados e irá usar a função ParserCacheServer da class Cache para extrair a informação para a sua cache. Estas duas funções vão ser declaradas no ficheiro log.

Depois desta fase, o servidor estará apto para receber uma ligação, neste caso, so haverão duas possiveis por agora, uma ligaçao de um cliente onde ira receber uma query caso receba um “QE” , ou um pedido de transferência de zona por parte de um SS,caso receba um “domain:”. Estas duas ligações também vao ser declaradas no ficheiro logs.

Para a ligação com o cliente, o servidor vai receber a mensagem do CL num formato conciso e depois de já estar formulada a resposta este irá envia la de volta para o cliente. A rececao da querie e o envio da resposta da querie vao estar notificadas na logs.

//prints da alguma cena possivelmente

Para a ligação com o SS , o servidor vai receber como tínhamos referido anteriormente “domain:”. De seguida o SP vai enviar o numero de entradas da sua cache, espera a mensagem de confirmação so SS e depois envia um socket por cada linha de entrada retirada da cache.Cada mensagem enviada ou recebida neste processo de transferência vai ser notificada no logs.

**ServerS**

Na classe ServerS, temos como objetivo a implementação de um servidor do tipo primário. Este servidor vai ser quase igual ao SP no que toca á criação de uma cache, do log e da ligação com o cliente. As queries recebidas por este servidor vao ser descodificadas e resolvidas da mesma forma que fizemos com o SP.

A transferência de zona também vai ser feita quando iniciamos o servidor SS, mandando assim a mensagem”domain:” para inciar o processo de tranferencia. Depois de receber o numero de linhas, vai mandar mensagem a “ok + o numero de linhas que recebeu do SP”. Quando o SP começar a mandar as linhas de entrada da base de dados, o SS vai receber e guardar na cache uma de cada vez. Cada mensagem recebida ou enviada neste processo de transferência vai ser notificada no logs.

**Cliente**

Na classe Cliente, temos como objetivo a implementação de uma aplicação cliente, este cliente ira ter como variável local apenas o seu Ip. Este cliente vai mandar socket para o SP ou SS, com uma querie em formato conciso já predefinida. Depois vai receber a resposta através de um socket.Tanto o envio da querie como a resposta recebida vao estar notificados nos logs.

**Logs**

Na classe Logs, pretendemos guardar informações de todas as açoes ocorridas nestes processos. Criando um ficheiro ao incio e ir acrescentando linhas com definidas pelo seu tipo de log, pela hora e pela informação revelada.

**Query**

A classe query vai ser a classe que vai trabalhar na descodificação da query e formulação da sua resposta. Para isso, as variáveis locais desta classe vão ser o id, Flags, nResponse, n Values, nAutho, nExtravalues(estes são os campos do HEADER Fields de uma mensagem DNS), mais o InfoName (nome do dominio), e Type(vai ser o tipo de dados que vao querer extrair da base de dados).

Primeiramente, esta classe recebe uma string(a mensagem concisa da query que chegou ao servidor), depois descodifica e faz o parse da mesma. Caso ocorra algum erro durante esta fase, o processo query termina e devolve a query com response code igual a 3. De seguida, verifica-se se o domínio da query é igual ao da base dados(para termos a certeza que esta a ser procurada a resposta no domínio certo).Caso não seja