Processamento de Linguagens (3º ano do MiEI)

Trabalho Prático 1

Transformador Publico2NetLang

Eduardo Costa	José Pinto	Luís Lopes	Ricardo Carvalho
(A85735)	(A84590)	(A85367)	(A84261)

5 de Abril de 2020

Resumo

Neste relatório mostra-se o trabalho realizado para desenvolver um filtro de texto, recorrendo ao gerador Flex, que permita estudar o conteúdo dos comentários feitos numa notícia publicada no site do jornal O Público, cujos dados relevantes à análise pretendida devem ser extraídos do ficheiro HTML fornecido e transformados no formato JSON apresentado no enunciado.

Conteúdo

Capítulo 1

Introdução

1.1 Introdução

O trabalho de seguida apresentado tem como principais objetivos aumentar a experiência de uso do ambiente Linux e de algumas ferramentas de apoio à programação, aumentar capacidade de escrever Expressões Regulares (ER) para descrição de padrões de frases, desenvolver, a partir destas, sistemática e automaticamente, processadores de Linguagens Regulares que filtrem ou transformem textos com base no conceito de regras de produção Condição-Ação e utilizar o Flex para gerar filtros de texto em C.

De entre as sete propostas de trabalhos apresentadas no enunciado fornecido pelos docentes, optamos pelo 2.4, referente ao "Transformador Publico2NetLang", apresentado abaixo. A escolha deste tema não teve qualquer motivo relevante para além do facto de nos parecer, à primeira vista, o tema mais interessante e que imediatamente nos fez pensar como poderíamos abordá-lo.

1.2 Seleção de enunciados

De entre os 7 temas propostos, o enunciado do escolhido é o seguinte:

Transformador Publico2NetLang (2.4)

Analise com todo o cuidado o ficheiro http://natura.di.uminho.pt/ jj/pl-20/TP1/Publico_extraction_portuguese_-comments_4.html o qual contém os comentários (85 neste exemplo) a uma noticia publicada no jornal O Público, extraídos da página HTML da versão online do dito jornal.
Para se fazer um estudo sócio-linguístico de forma e conteúdo dos comentários que a notícia suscitou, os dados relevantes à análise pretendida devem ser extraídos do ficheiro HTML fornecido e devem ser transformados no formato JSON a seguir mostrado.

```
"commentThread": [
    {
    "id": "STRING",
    "user": "STRING",
    "date": "STRING",
    "timestamp": NUMBER,
    "commentText": "STRING",
    "likes": NUMBER,
    "hasReplies": TRUE/FALSE,
    "numberOfReplies": NUMBER
    "replies": [ ]
    },.......
]
```

Construa então um filtro de texto, recorrendo ao gerador FLex, que realize o processamento explicado, tendo em consideração que as respostas1 que surjam a um dado comentário devem ser aninhadas, na forma de uma lista, dentro do campo "replies"do dito comentário, seguindo evidentemente o mesmo formato apresentado.

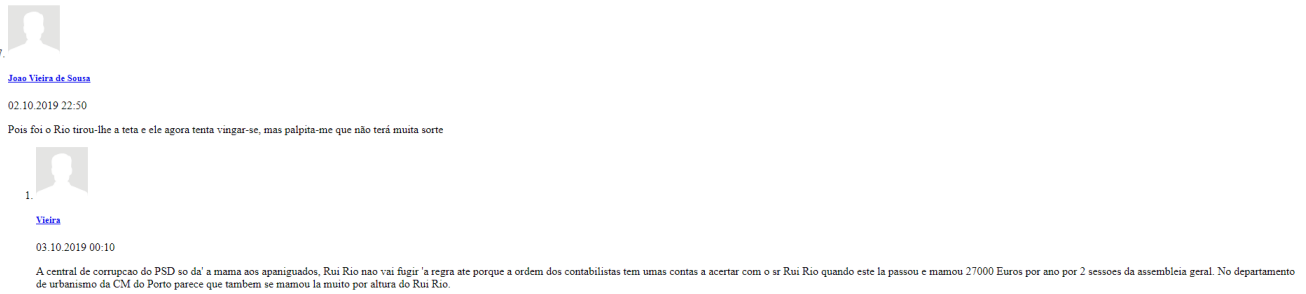
1.3 Estrutura do Relatório

Neste relatório iremos começar por explicar o problema, as características dos dados fornecidos no ficheiro HTML e os padrões de frase por nós encontrados, que levaram à resolução dos problemas inicialmente propostos. Numa segunda fase apresentamos a nossa proposta de solução, a nossa abordagem e a forma como a implementamos, com partes relevantes de código nos casos que considerarmos necessários. Por fim, apresentamos, também, alguns testes para verificar a solução encontrada e a funcionalidade do trabalho realizado.

Capítulo 2

Análise e Descrição do Problema

Como já foi referido, o problema consiste, essencialmente, na transformação de um ficheiro HTML num ficheiro JSON que permita uma análise dos comentários feitos numa notícia publicada no site do jornal O Público, como os ilustrados abaixo, na figura.


7.

Joao Vieira de Sousa

02.10.2019 22:50

Pois foi o Rio tirou-lhe a teta e ele agora tenta vingar-se, mas palpita-me que não terá muita sorte

1.

Vieira

03.10.2019 00:10

A central de corrupcao do PSD so da' a mama aos apaniguados, Rui Rio nao vai fugir 'a regra ate porque a ordem dos contabilistas tem umas contas a acertar com o sr Rui Rio quando este la passou e mamou 27000 Euros por ano por 2 sessoes da assembleia geral. No departamento de urbanismo da CM do Porto parece que tambem se mamou la muito por altura do Rui Rio.

O ficheiro que se encontra na imagem seguinte, apresenta o mesmo comentário e respetiva resposta de um outro utilizador, desta vez em HTML. A partir do momento em que começamos a analisar este ficheiro e a sua organização, chegamos a várias conclusões e começamos a pensar na forma como poderíamos encarará-la. Partimos, então, numa fase inicial, para a separação dos vários tipos considerados relevantes:

- ID do comentário ("comment");
- Autor do comentário ("comment__author");
- Data e hora do comentário ("dateline comment__dateline");
- Conteúdo do comentário ("comment__content");
- Respostas ao comentário, que são, por sua vez, também elas comentários ("comments__list");

```
<li class="comment" data-comment-id="8f949889-2606-4749-1c42-08d7471cb23d">
<div class="comment__inner">
<div class="comment__meta">
<span class="avatar comment__avatar">
<span class="avatar__pad">
<img alt="" data-interchange="https://static.publicocdn.com/files/homepage/img/avatar_74x74.png, small],
[https://static.publicocdn.com/files/homepage/img/avatar_74x74.png, retina]" data-resize="kmou55-interchange" data-t="2nlz4i-
t" id="kmou55-interchange" src="https://static.publicocdn.com/files/homepage/img/avatar_74x74.png"/>
</span>
</span>
<h5 class="comment__author">
<a href="/utilizador/perfil/275d9ced-3b68-4dc4-bd47-441c76edf850" rel="nofollow">Joao Vieira de Sousa </a>
</h5>
<span class="comment__reputation comment__reputation-r2" title="Experiente"><i aria-hidden="true" class="i-check"></i></span>
<!--<span class="comment__location">Cruz Quebrada Dafundo</span>-->
<time class="dateline comment__dateline" datetime="2019-10-02T22:50:07.08">
<a class="comment__permalink">02.10.2019 22:50</a>
</time>
</div>
<div class="comment__content">
<p>
                Pois foi o Rio tirou-lhe a teta e ele agora tenta vingar-se, mas palpita-me que não terá muita sorte
            </p>
</div>
</div>
<ol class="comments__list">
<li class="comment" data-comment-id="93506ff9-5244-4c37-455e-08d7471e83ae">
<div class="comment__inner">
<div class="comment__meta">
<span class="avatar comment__avatar">
<span class="avatar__pad">
<img alt="" data-interchange="https://static.publicocdn.com/files/homepage/img/avatar_74x74.png, small],
[https://static.publicocdn.com/files/homepage/img/avatar_74x74.png, retina]" data-resize="7qi934-interchange" data-t="qj7imo-
t" id="7qi934-interchange" src="https://static.publicocdn.com/files/homepage/img/avatar_74x74.png"/>
</span>
</span>
<h5 class="comment__author">
<a href="/utilizador/perfil/ad91ec31-3d7e-45f5-afd3-01905b95c359" rel="nofollow">Vieira </a>
</h5>
<span class="comment__reputation comment__reputation-r4" title="Moderador"><i aria-hidden="true" class="i-check"></i></span>
<!--<span class="comment__location"></span>-->
<time class="dateline comment__dateline" datetime="2019-10-03T00:10:15.203">
<a class="comment__permalink">03.10.2019 00:10</a>
</time>
</div>
<div class="comment__content">
<p>
                    A central de corrupcao do PSD so da' a mama aos apaniguados, Rui Rio nao vai fugir 'a regra ate
porque a ordem dos contabilistas tem umas contas a acertar com o sr Rui Rio quando este la passou e mamou 27000 Euros por ano
por 2 sessoes da assembleia geral. No departamento de urbanismo da CM do Porto parece que tambem se mamou la muito por altura
do Rui Rio.
                </p>
</div>
</div>
</li>
</ol>
<form class="form comments__form expanded" data-abide="" data-t="hagfba-t" style="display:none"></form>
</li>
```

Pelo que vimos no ficheiro HTML, exemplificado na imagem anterior, percebemos que teríamos de filtrar cada um destes elementos individualmente de forma a podermos colocar apenas a informação relevante no ficheiro JSON, eliminando, assim, a parte do código HTML e obtendo os dados fundamentais acerca do comentário. Pudemos ver, também, que, tanto no site fornecido, como no respetivo ficheiro HTML, não existe qualquer referência a "likes", como apresenta o enunciado, pelo que esse atributo foi por nós ignorado na posterior realização do trabalho, que mais à frente neste relatório se apresenta.

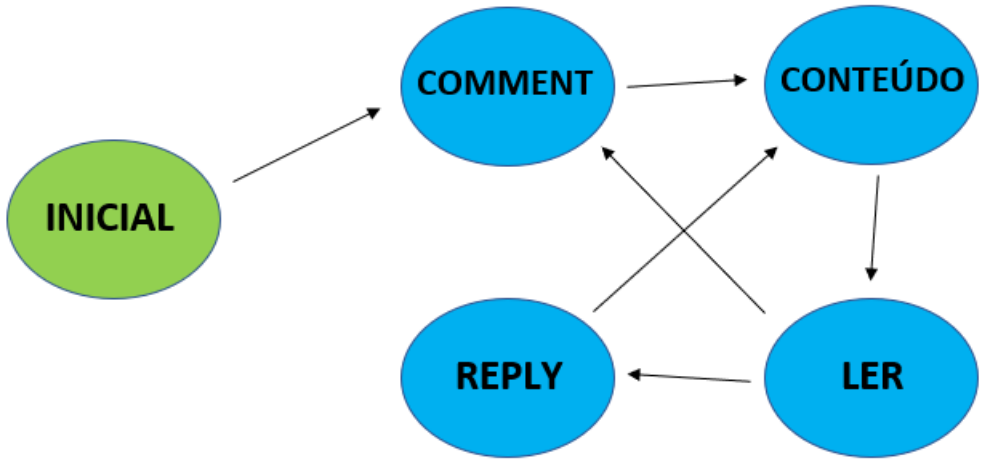
Capítulo 3

Desenho da Solução e sua Implementação

Como já mencionado anteriormente, o ficheiro HTML distingue cada um dos tipos de informação apontados no ponto anterior pelo que a nossa abordagem à resolução do problema passou pela elaboração de filtros Flex e funções essenciais em C para extração dos dados necessários para colocar no ficheiro JSON. Daqui surgiram 3 ficheiros principais: *thread.c*, *thread.h* e *filtro.l*. Sendo o primeiro o filtro propriamente dito e o *thread.c* contém funções importantes para a filtragem e armazenamento de informação.
O filtro por nós construído tem os estados "ler", "comment", "reply"e "conteúdo".

```
%x ler comment reply conteudo
```

O autómato finito determinístico que representa a nossa abordagem deste problema representa-se da seguinte forma:



Estado ler

O estado "ler"implementado, como mostra a imagem abaixo, distingue um novo comentário de uma secção de respostas e retira o *id* do comentário através da função "retiraId"definida em *thread.c*. Este estado é também responsável por iniciar o estado "comment"ou o estado "reply", dependendo da situação.

```
{commentList}\<li.+   {BEGIN comment; fprintf(json, "\"id\" : \"%s\"\n", retiraID(yytext));};

<ler>\<li.+ {if(hasReplies==1){sprintf(idR, "%s", retiraID(yytext));BEGIN reply;numberOfReplies++;}
            else{BEGIN comment; fprintf(json, "\"id\" : \"%s\",\n", retiraID(yytext));};};

<ler>{commentList} {hasReplies=1;numberOfReplies=1; replys = malloc(sizeof(CommentThread**));
                    BEGIN reply; fprintf(json,"\"hasReplies\" : \"TRUE\",\n");}

<ler>{commentExpanded} { if(hasReplies!=1){
                           fprintf(json,"\"hasReplies\" : \"FALSE\",\n");}
                         fprintf(json,"\"numberOfReplies\" : %d,\n",numberOfReplies);
                         writeCommentThread(replys, json, numberOfReplies);
                         if(comentarios<84){
                           fprintf(json,"],\n\n");
                           }
                         else{fprintf(json,"]\n"); }
                         hasReplies = 0; numberOfReplies =0;};


<ler>.+ ;
<ler>\n ;

<ler><<EOF>>   {fprintf(json,"\n}\n]"); exit(0);};
```

Estado comment

O estado "comment"retira o nome do utilizador, data e *timestamp*, utilizando funções definidas no *thread.c* que através do "strtok"filtram apenas a informação essencial. Para além disso, é também responsável por iniciar o estado "conteúdo".

```
<comment>\<(h5).+\>[ \n\r]*(Conta) {fprintf(json, "\"user\" : \"Conta desactivada\",\n");};

<comment>{commentAuthor} { fprintf(json, "\"user\" : \"%s\",\n", retiraUser(yytext));};

<comment>{data} {fprintf(json, "\"date\" : \"%s\",\n", retiraData(yytext));};

<comment>{timeStamp} {fprintf(json, "\"timestamp\" : \"%s\",\n",retirarTimeStamp(yytext));};

<comment>{content} {texto = "";BEGIN conteudo;};

<comment>{branco} ;
<comment>.+ ;
```

Estado conteúdo

Para o estado "conteúdo", inicialmente tentamos ler o conteúdo dos comentários do início ao fim, mas constatamos que tal não era possível, pelo que tivemos que alterar a nossa abordagem neste ponto, passando, então, a ler parágrafo a parágrafo, concatenando-os até chegar ao final do comentário que é identificado por </p>. Mesmo depois de uma filtragem bem sucedida, chegamos à conclusão que o ficheiro JSON gerado não era válido, por isso definimos uma função "retiraCommentText", em *thread.c*, responsável por retirar informação desnecessária ao comentário filtrado. Esta, sempre que encontra uma mudança de parágrafo, escreve /p e sempre que encontra aspas escreve \". No final do "conteúdo"o filtro volta ao estado "ler".

```
<conteudo>{paragrafo} {
                        junta(yytext);
                      };
<conteudo>\<\/p> {
                  comentarios++;
                  junta(yytext);
                  texto=retiraCommentText(texto);
                  if(hasReplies ==1){
                  replys[numberOfReplies-1]= newCommentThread(idR,userR,dateR,timestampR,texto,0);
                  BEGIN ler;
                  }
                  else{
                  BEGIN ler;
                  fprintf(json,"\"commentText\" : \"%s\",\n", texto);
                  fprintf(json,"\"likes\" : \"%d\",\n",likes);
                  }};

<conteudo>{branco} ;
```

Função que concatena parágrafos, como referido acima:

```
void junta(char* yytext){
          char* result = (char*) malloc(sizeof(char*)*(strlen(yytext) + strlen(texto) + 1));
          strcpy(result, texto);
          strcat(result, yytext);
          texto = result;
}
```

Função "retiraCommentText", que retira informação desnecessária do conteúdo, também esta referida anteriormente:

```
//char* retiraCommentText
char* retiraCommentText(char* str) {
    char* token;char* token2;char* tok;
    token = strtok(str, ">");
    token = strtok(NULL, " ");
    token = strtok(NULL, "<");
    while (token[0] == ' ') token++;
    int i=0;int i2=0;
    token2 = (char*) malloc(sizeof(char*)*(strlen(token)));
    while(token[i+1]!='\0'){
        if(token[i]=='\"'){
            token2[i2]='\\';
            i2++;
            token2[i2]=token[i];
        }
        if(token[i]=='\n'){
            token2[i2]='\\';
            i2++;
            token2[i2]='p';
            i2++;
            token[i]=' ';
        }
        token2[i2]=token[i];
        i++;i2++;}
    int k =0;
    tok = (char*) malloc(sizeof(char*)*i);
    while(token2[k]!='\0'){
        for(int f=k; f<200+k;f++){
            if(token2[f]!='\0'&& token2[f]!=' ' && token2[f]!='\n'){
                break;}
            if(token2[f]=='\0')
                return tok;}
        tok[k]=token2[k];
        k++;}
//printf(token2);
    return(tok);
}
```

Estado reply

Quanto ao estado "reply", são processadas as secções de respostas do ficheiro. Este é iniciado pelo estado "ler"e, neste estado, os comentários vão sendo armazenados em memória numa estrutura "CommentThread** replys", por nós definida.

```
<reply>\<li.+ {sprintf(idR, "%s", retiraID(yytext));};

<reply>\<(h5).+\>[ \n\r]*(Conta) {sprintf(userR, "%s", "Conta desactivada");};

<reply>{commentAuthor}  {sprintf(userR, "%s", retiraUser(yytext));};

<reply>{data} {sprintf(dateR, "%s", retiraData(yytext));};

<reply>{timeStamp} {sprintf(timestampR, "%s", retirarTimeStamp(yytext));};

<reply>{content} {texto = ""; BEGIN conteudo;};


<reply>{branco} ;
<reply>.+ ;
```

Como podemos ver, este estado inicia o estado "conteúdo"e só aí é que a resposta é armazenada em memória. De seguida, é iniciado o estado "ler"onde existe uma condição "if"no filtro <ler>\<li.+ para continuar a escrever *replys*, caso seja necessário. Desta forma, para depois escrever as respostas armazenadas e todas as informações necessárias no ficheiro JSON, utilizamos a seguinte função:

```
void writeCommentThread(CommentThread** c, FILE* json, int size) {
    fprintf(json, "\"replys\": [");
    for (int i = 0; i < size; i++) {
        fprintf(json, "{\n");
        fprintf(json, "\"id\" : \"%s\",\n", c[i]->id);
        fprintf(json, "\"user\" : \"%s\",\n", c[i]->user);
        fprintf(json, "\"date\" : \"%s\",\n", c[i]->date);
        fprintf(json, "\"timestamp\" : \"%s\",\n", c[i]->timestamp);
        fprintf(json, "\"commentText\" : \"%s\",\n", c[i]->commentText);
        if (size == 1) { fprintf(json, "\"likes\" : \"%d\"}", c[i]->likes); }
        else {
            if (i == size - 1) fprintf(json, "\"likes\" : \"%d\"}", c[i]->likes);
            else { fprintf(json, "\"likes\" : %d},", c[i]->likes); }
        }
    }
}
```

A estrutura "CommentThread"referida encontra-se definida em *thread.h* e apresenta-se da seguinte forma:

```
typedef struct CommentThread {
    char id[1024];
    char user[1024];
    char date[50];
    char timestamp[50];
    char* commentText;
    int likes;
    int hasReplies;
    int numberOfReplies;
    char** replies;
}CommentThread;
```

Definições auxiliares

Utilizamos, também, algumas definições para os filtros, de forma a simplificar e organizar a nossa implementação e raciocínio, como mostra a imagem seguinte:

```
branco [ \t\n]+
pL    [A-Za-zçÇáÁàÀéÉêÊõÃú]

commentList \<(ol).+
commentExpanded \<(form).+
commentAuthor \<(h5).+\>[\n\r]*\<(a).+\<\/(a)\>
data \<(time).+
timeStamp \<a{branco}(class)\=\"(comment__permalink).+
content \<(div){branco}(class)\=\"(comment__content).+
paragrafo <p[ "!?,;\-.\n\t-(A-Za-zçÇáÁàÀéÉêÊõÃÚÔúíóÉ)0-9:/=>'ª°%?“–]+[ \t\n]+
```

Main

Por fim, a definição da nossa "main"passou por abrir o ficheiro HTML fornecido e o nosso ficheiro JSON, e, de seguida, chamar o filtro mostrado acima para que, juntamente com as funções implementadas, preencham o ficheiro JSON de forma adequada, como pedido no enunciado.

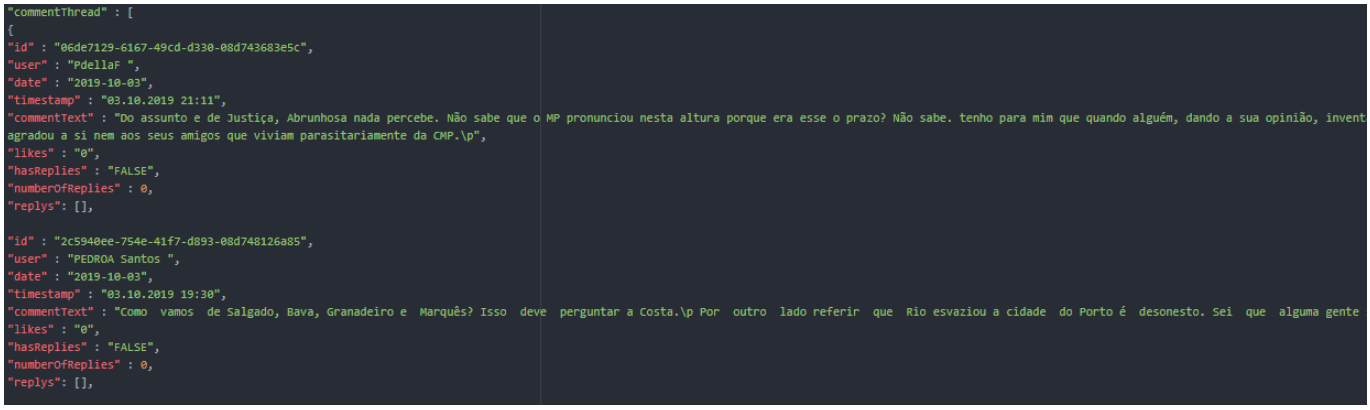
```
int main(int argc, char **argv){
  yyin = fopen("Publico_extraction_portuguese_comments_4.html","r");
  json = fopen("commentThread.json","w");
  fprintf(json, "\"commentThread\" : [\n{\n");
  yylex();
  texto = " ";

  fclose(yyin);
  return 0;
}
```

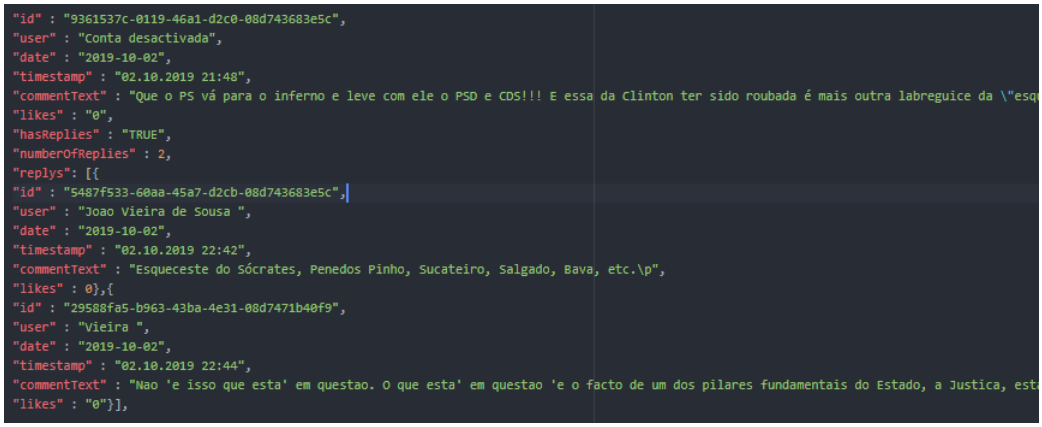
Capítulo 4

Resultados Finais

Os resultados do projeto, que se encontram representados nas imagens abaixo, mostram a forma como o ficheiro JSON é construído. Podemos ver, então, que todos os campos são corretamente completos, apenas com a informação necessária e formando um ficheio JSON válido. É importante reforçar, tal como dito inicialmente, que o número de likes foi por nós ignorado durante a implementação, pois não existe qualquer referência a este atributo no site da notícia nem no ficheiro HTML associado. Assim, ao longo do ficheiro JSON, este número foi mantido a 0.



Nesta segunda imagem mostra-se, pela sua importância e dificuldade de implementação, que as respostas aos comentários estão a ser devidamente introduzias, assim como as suas informações. É de notar que o número de respostas ao comentário principal está a ser devidamente incrementado e que não há referência a número de respostas nas respostas, pois, neste caso, não é possível essa situação ocorrer.



Capítulo 5

Conclusão

Finalmente, como conclusão, consideramos ter concluído com sucesso o proposto no enunciado escolhido, cumprindo todos os requisitos e funcionalidades nele presentes. Consideramos, também, que este projeto nos ajudou a consolidar, de uma forma mais prática, os conceitos abordados nas aulas. Gostaríamos também de destacar que, devido à situação em que nos encontramos e por não sabermos como se procederá a defesa deste trabalho, tentamos elaborar um relatório o mais explícito possível, com imagens, e respetivas descrições, referentes a todas ou quase todas as fases do processo, funções e filtros implementados.