Processamento de Linguagens (3º ano MIEI)

Trabalho Prático nº2 - Enunciado 4

Relatório de Desenvolvimento-Grupo 39

Henrique Paz (a84372), José Santos (a84288), Pedro Gomes (a84220)

28 de Junho de 2020

Resumo

Este relatorio inicia-se com uma breve contextualização, seguindo-se a declaração dos objetivos deste trabalho e em que é que este consiste. De seguida é feita uma descrição do problema cuja resolução é apresentada no capitulo seguinte, com a especificacao da linguagem desenvolvida,tal como o analizador lexico e as estruturas de dados usadas no processamento do texto.Termina-se com a explicacao de como é formado o ficheiro json e são apresentados alguns testes realizados. Por fim uma analise e as conclusões tiradas sobre o desenvolvimente deste trabalho pratico.

Conteúdo

Capítulo 1

Introdução

1.1 Contexto

O relatorio foi elaborado no ambito do segundo trabalho pratico da UC de Processamento de Linguagens, que se insere no 2º semestre do 3º ano do Mestrado Integrado de Engenharia Informatica.

1.2 Objetivos e Trabalho Proposto

Para este trabalho foi proposta a especificaçao de uma gramatica concreta de um subconjunto da linguagem TOML e tambem um analisador lexico e sintatico com recurso as ferramentas Flex,Yacc.

1.3 Descrição informal do problema

Pretende-se obter um ficheiro JSON com toda a informaçao que existe no ficheiro TOML, para isso é preciso definir:
- Defimir uma gramatica independente de contexto (Yacc).
- Desenvolver um analisador lexico (Flex)
- Associar ações semanticas na gramatica para construir uma estrura de dados auxiliar.
- Percorrer a estrutura e criar o ficheiro JSON correspondente.

Capítulo 2

Conceção/Desenho da Resolução

2.1 Estruturas da linguem TOML escolhidas

O ficheiro de input poderá ter 3 tipos de Estruturas e eventualmente um titulo, não sendo obrigatorio. Os tipos das estruturas aceites são: owner, database, servers.
De seguida vamos dizer quais são os atributos que a nossa gramatica aceita em cada estrutura, de notar que comentarios serão ignorados visto que JSON não permite comentarios.

2.1.1 Atributos do owner

- name - nome do proprietario que é uma string
- date - consiste numa data, ano-mes-dia e é passada posteriomente como uma string
- time - consiste numa hora, hh:mm:ss ou hh:mm, em que nao é obrigatorio especificar os segundos e é passada posteriomente como uma string.

2.1.2 Atributos da database

- server - consiste no ip do server da base de dados no entanto é uma string e por isso tem formato livre.
- ports - array de inteiros, e é passado posteriormente cada elemento do array como um inteiro.
- connection_max - indica a maxima conecçao da base de dados, ou seja, é um inteiro e é passado como tal
- enabled - é tratado como um booleano, ou seja, só pode ser true ou false, e posteriomente é passado como uma string.

2.1.3 Atributos dos servers

Consideramos que a estrutra server no toml só tem 1 atributo que é ter 1 ou mais SubServers,exemplo [servers.alpha].
O subServer esse pode ter varios atributos sendo eles:

- ip - consiste no Ip do server, no entanto é considerado uma string por isso tem formato livre.
- dc - é uma string e por isso tem formato livre.
- host- array de strings, e por isso cada elemento tem formato livre.

2.2 Estrutura de dados

De forma a guarda toda a informação que recolhemos ao longo do ficheiro para depois de forma ordenada formatar o .JSON, nós definimos varias estruras de dados com a ajuda da libraria GLIB.

2.2.1 Estrutura toml

```
typedef struct toml {
        GString * titulo;
        GSList * arrDB;
        GSList * arrOwner;
        GSList * arrSv;
} *Toml;
```

Figura 2.1: Estrutura toml

Esta estrutura é a estrutura principal que no final do yyparse vai conter toda a informação que decimos reconhecer,especificado em cima, do ficheiro TOML. Para isso tem uma string como o titulo do ficheiro e 3 listas com estruturas de databases, owners e servers respetivamente.

2.2.2 struct dataBase

```
typedef struct dataBase {
        GString * serverIp;
        int connect_max;
        GString * enabled;
        int portsNumber;
        int ports[];

}*DataBase;
```

Figura 2.2: struct database

Esta estrutura é a estrutura que contem a informação de uma base de dados. Como é possivel ver no print acima esta estrutura tem uma string com o Ip do server, um int com a connect_max, uma string que só pode ser True of False no enabled,e um array de inteiros com os ports.

2.2.3 Estrutura owner

```
typedef struct owner {
        GString * name;
        GString * date;
        GString * time;

}*Owner;
```

Figura 2.3: struct owner

Esta estrutura é a estrutura que contem a informação de um owner, é bastante simples visto que consiste em 3 strings, uma para o nome outra para a data e por ultimo uma para a hora.

2.2.4 Estrutura servers

```
typedef struct servers {
        GString * nome;
        GString * ip;
        GString * dc;
        GSList * hosts;
}*Servers;
```

Figura 2.4: struct servers

Esta estrutura é na verdade uma estrutura para o que nós consideramos na gramatica de SUBSERVERS,exemplo [servers.alpha], em que tem o nome do mesmo sobre a forma de uma string, o ip e o dc tambem como strings e uma lista com os host que neste caso vai ser uma lista de strings.

2.3 Gramatica independente de Contexto

Tendo em conta o que foi definido anteriormente foi possivel definir uma gramatica para expressar o conhecimento pretendido.
Esta gramatica aceita varias estruturas mesmo que sejam do mesmo tipo, ou seja, duas base de dados, tres owners, etc, e aceita 1 ou 0 titulos sendo que se tiver 2 vai dar um erro sintatico e o seu simbolo inicial é o toml.
Alem disso tem varias palavras reservadas tais como TITLE DATABASE... que em conjunto com o analisador lexico, que está no capitulo 5, percorre todo ficheiro toml. Tambem tem varios simbolos que delimitam atributos como por ex '[' delimita o começo de uma estrutura e ',' que delimita os elementos dos arrays entre outros. Por fim tem tambem simbolos variaveis tais como texto, SUBSERVER e num, que o texto é normalmente um cojunto de caracteres entre parenteses, SUBSERVER corresponde a cada server existente e num a qualquer inteiro encontrado.

```
%}
%union{char* nome; int numero;}

%token TITLE DATABASE SERVERS OWNER

%token NAME DATE TIME

%token SERVER PORTS CONNECT_MAX DB_ENABLED

%token IP DC HOSTS

%token <nome> texto
%token <nome> SUBSERVER
%token <numero> num

%%
Toml : Titulo Body
     | Body
     ;

Titulo : TITLE '=' texto
       ;

Body : Body Struct
     | Struct
     ;
```

```
Struct : '[' ']'
       | '[' OWNER ']' OWNER_Struct
       | '[' DATABASE ']' DBStruct
       | '[' SERVERS ']' SvStruct
       ;

OWNER_Struct : OWNER_Struct OWNER_Elem
             | OWNER_Elem
             ;

OWNER_Elem: NAME '=' texto
          | DATE '=' texto
          | TIME '=' texto
          ;

DBStruct : DBStruct DBElem
         | DBElem
         ;

DBElem : SERVER '=' texto
       | PORTS  '=' Array
       | CONNECT_MAX '='
       | DB_ENABLED '=' texto
       ;

SvStruct : SvStruct SvComp
         | SvComp
         ;
```

```
SvComp : SUBSERVER LSvELem
       | SUBSERVER
       ;

LSvELem : LSvELem SvELem
        | SvELem
        ;

SvELem : IP '=' texto
       | DC '=' texto
       | HOSTS '=' Array
       ;

Array : '[' ']'
      | '[' ArrayElem ']'
      ;


ArrayElem : texto ',' ArrayElem
          | num ',' ArrayElem
          | texto
          | num
          ;
```

Capítulo 3

Geração do JSON e Testes

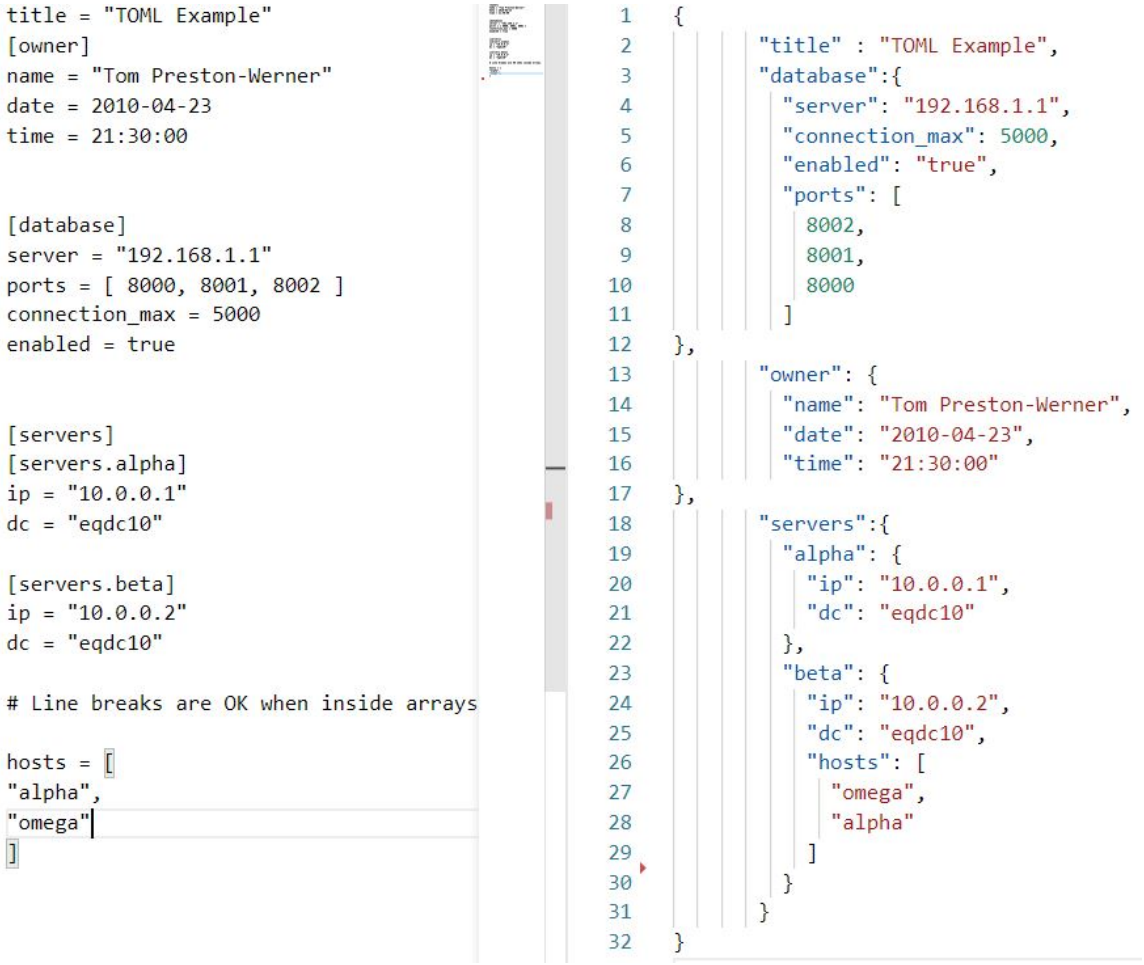
3.1 Geraçao do JSON

Depois de o yyparse concluir é feita a abretura do ficheiro de output e é de seguida é percorrida a estrutura head do tipo toml,ou seja o titulo e os seus arrays, de notar que a ordem de escrita no ficheiro é sempre a mesma primeiro o array de databases, depois o de owners e so no fim o de servers, isto garante uma estrutura mais regular ao ficheiro json.

```
void formatToJson() {
        printOpenJsonBracket();
        printTitle();
        printDatabases();
        printOwners();
        printServers();
        printCloseJsonBracket(1);
}
```

3.2 Testes

Neste secção vamos apresentar alguns testes que fizemos com o input e o seu output resultante.O input é a parte esquerda da imagem e o output, o json correspondente, é a parte direita da imagem. Decidimos apresentar 3 testes, o primeiro é o exemplo base do enunciado, o segundo é um input com varias estrutruras do tipo "servers".Por fim o terceiro teste é um em que o input tem 2 base de dados não sequenciais.



```
title = "TOML Example"
[owner]
name = "Tom Preston-Werner"
date = 2010-04-23
time = 21:30:00


[database]
server = "192.168.1.1"
ports = [ 8000, 8001, 8002 ]
connection_max = 5000
enabled = true


[servers]
[servers.alpha]
ip = "10.0.0.1"
dc = "eqdc10"

[servers.beta]
ip = "10.0.0.2"
dc = "eqdc10"

# Line breaks are OK when inside arrays

hosts = [
"alpha",
"omega"
]
```

```
{
    "title" : "TOML Example",
    "database":{
      "server": "192.168.1.1",
      "connection_max": 5000,
      "enabled": "true",
      "ports": [
        8002,
        8001,
        8000
      ]
},
    "owner": {
      "name": "Tom Preston-Werner",
      "date": "2010-04-23",
      "time": "21:30:00"
},
    "servers":{
      "alpha": {
        "ip": "10.0.0.1",
        "dc": "eqdc10"
      },
      "beta": {
        "ip": "10.0.0.2",
        "dc": "eqdc10",
        "hosts": [
          "omega",
          "alpha"
        ]
      }
    }
}
```

Figura 3.1: teste 1

```
title = "TOML Example"
[owner]
name = "Tom Preston-Werner"
date = 2010-04-23
time = 21:30:00


[database]
server = "192.168.1.1"
ports = [ 8000, 8001, 8002 ]
connection_max = 5000
enabled = true


[servers]
[servers.alpha]
ip = "10.0.0.1"
dc = "eqdc10"

[servers.beta]
ip = "10.0.0.2"
dc = "eqdc10"

hosts = [
"alpha",
"omega"
]

[servers]
[servers.pedro]
ip = "12.0.0.1"
dc = "asdasd"
hosts = [
"teste",
"teste2"
]
```

```
{
    "title" : "TOML Example",
    "database":{
      "server": "192.168.1.1",
      "connection_max": 5000,
      "enabled": "true",
      "ports": [
        8002,
        8001,
        8000
      ]
},
    "owner": {
      "name": "Tom Preston-Werner",
      "date": "2010-04-23",
      "time": "21:30:00"
},
    "servers":{
      "alpha": {
        "ip": "10.0.0.1",
        "dc": "eqdc10"
      },
      "beta": {
        "ip": "10.0.0.2",
        "dc": "eqdc10",
        "hosts": [
          "omega",
          "alpha"
        ]
      },
      "pedro": {
        "ip": "12.0.0.1",
        "dc": "asdasd",
        "hosts": [
          "teste2",
          "teste"
        ]
      }
    }
```

Figura 3.2: teste 2

```toml
title = "TOML Example"
[owner]
name = "Tom Preston-Werner"
date = 2010-04-23
time = 21:30:00

[database]
server = "192.168.1.1"
ports = [ 8000, 8001, 8002 ]
connection_max = 5000
enabled = true

[servers]
[servers.alpha]
ip = "10.0.0.1"
dc = "eqdc10"

[servers.beta]
ip = "10.0.0.2"
dc = "eqdc10"

hosts = [
"alpha",
"omega"
]

[database]
server = "300.212.3.3"
ports = [ 2000, 2001, 2002 ]
connection_max = 2000
enabled = false
```

```json
{
    "title" : "TOML Example",
    "database":{
      "server": "192.168.1.1",
      "connection_max": 5000,
      "enabled": "true",
      "ports": [
        8002,
        8001,
        8000
      ]
    },
    "database":{
      "server": "300.212.3.3",
      "connection_max": 2000,
      "enabled": "false",
      "ports": [
        2002,
        2001,
        2000
      ]
    },
    "owner": {
      "name": "Tom Preston-Werner",
      "date": "2010-04-23",
      "time": "21:30:00"
    },
    "servers":{
      "alpha": {
        "ip": "10.0.0.1",
        "dc": "eqdc10"
      },
      "beta": {
        "ip": "10.0.0.2",
        "dc": "eqdc10",
        "hosts": [
          "omega",
          "alpha"
        ]
```

Figura 3.3: teste 3

Capítulo 4

Conclusão

Para concluir, ao longo do desenvolvimento do trabalho, fomos tomando varias decições nomeadamente na parte da gramatica,o que levou a um resultado que na nossa opinião cumpre o seu proposito e ainda permite a expansão facimente a mais tipos de nodos.Visto isto o nosso grupo ainda tentou adicionar mais um nodo, clientes, mas o tempo mostrou-se pouco para tal. Em relação as ações semanticas foram bastante diretas visto que nós usamos a glib nas estruturas de dados o que tornou o nosso trabalho um pouco mais facil. Depois de ter toda a informação em memoria na estrutura criada por nós a formação do json não impos grandes problemas.
Posto isto achamos que o nosso trabalho cumpre os requesitos propostos e por isso achamos o resultado obtido positivo.

Capítulo 5

Apendice

5.1 analisador.l

```
%option noyywrap
%%
(?i:title)                          { return (TITLE); }
(?i:owner)                          { return (OWNER); }
(?i:database)                       { return (DATABASE);}
(?i:servers)                        { return (SERVERS);}
(?i:name)                           { return (NAME); }
(?i:date)                           { return (DATE); }
(?i:time)                           { return (TIME); }
(?i:server)                         { return (SERVER); }
(?i:ports)                          { return (PORTS); }
(?i:connection_max)                 { return (CONNECT_MAX); }
(?i:enabled)                        { return (DB_ENABLED); }
(?i:ip)                             { return (IP); }
(?i:dc)                             { return (DC); }
(?i:hosts)                          { return (HOSTS);}
(?i:(\[servers\.[a-zA-Z]+\]))       { yytext[yyleng-1] = '\0'; yylval.nome = strdup(yytex
[0-9]+                              { yylval.numero = atoi(yytext); return num; }
[0-9]{4}\-[0-9]{2}\-[0-9]{2}        { yylval.nome = strdup(yytext); return texto;}
[0-9]{2}\:[0-9]{2}(\:[0-9]{2})?     { yylval.nome = strdup(yytext); return texto;}
[Tt][Rr][Uu][Ee]                    { yylval.nome = strdup(yytext); return texto;}
[Ff][Aa][Ll][Ss][Ee]                { yylval.nome = strdup(yytext); return texto;}
#(.*)\n                             {printf("%s\n",yytext);}
\"[^"]+\"                           { yytext[yyleng-1] = '\0'; yylval.nome = strdup(yytex
[\=\[\]\,]                          { return yytext[0];}
.|\n                                {;}
```

5.2 gramatica.y

```
typedef struct dataBase {
	GString * serverIp;
	int connect_max;
	GString * enabled;
	int portsNumber;
	int ports[];

}*DataBase;

typedef struct owner {
	GString * name;
	GString * date;
	GString * time;

}*Owner;

typedef struct servers {
	GString * nome;
	GString * ip;
	GString * dc;
	GSList * hosts;
}*Servers;

typedef struct toml {
	GString * titulo;
	GSList * arrDB;
	GSList * arrOwner;
	GSList * arrSv;
} *Toml;
```

```
Toml head;
Servers currSv;
Owner currOwner;
DataBase currDB;

Toml newToml(){
	Toml tl = malloc(sizeof(struct toml));
	tl->titulo = g_string_new("");
	tl->arrDB = NULL;
	tl->arrOwner = NULL;
	tl->arrSv = NULL;
	return tl;
}

void initCurrOwner(){
	currOwner = malloc(sizeof(struct owner));
	currOwner->name = g_string_new("");
	currOwner->date = g_string_new("");
	currOwner->time = g_string_new("");
}

void addNewOwner(){
	Owner l = currOwner;
	head->arrOwner = g_slist_append(head->arrOwner,l);
	initCurrOwner();
}
```

```
void addOwnerName(char * str){
	currOwner->name = g_string_append(currOwner->name,str);
}

void addOwnerDate(char * str){
	currOwner->date = g_string_append(currOwner->date,str);
}

void addOwnerTime(char * str){
	currOwner->time = g_string_append(currOwner->time,str);
}

// dataBase
void initCurrDB(){
	currDB = malloc(sizeof(struct dataBase));
	currDB->serverIp = g_string_new("");
	currDB->connect_max = -1;
	currDB->enabled =  g_string_new("");
	currDB->portsNumber = 0;
	//currDB->ports
}

void addNewDB(){
	head->arrDB = g_slist_append(head->arrDB,currDB);
	initCurrDB();
}
```

```
void addOwnerName(char * str){
	currOwner->name = g_string_append(currOwner->name,str);
}

void addOwnerDate(char * str){
	currOwner->date = g_string_append(currOwner->date,str);
}

void addOwnerTime(char * str){
	currOwner->time = g_string_append(currOwner->time,str);
}

// dataBase
void initCurrDB(){
	currDB = malloc(sizeof(struct dataBase));
	currDB->serverIp = g_string_new("");
	currDB->connect_max = -1;
	currDB->enabled =  g_string_new("");
	currDB->portsNumber = 0;
	//currDB->ports
}

void addNewDB(){
	head->arrDB = g_slist_append(head->arrDB,currDB);
	initCurrDB();
}
```

```
void addDbSvIp(char * str){
	currDB->serverIp = g_string_append(currDB->serverIp,str);
}

void addDbPort(int port){
	currDB->ports[currDB->portsNumber] = port;
	currDB->portsNumber++;
}

void addDbConnectMax(int i){
	currDB->connect_max = i;
}

void addDbEnabled(char * str){
	currDB->enabled = g_string_append(currDB->enabled,str);
}

//servers.
void initCurrSvs(){
	currSv = malloc(sizeof(struct servers));
	currSv->nome = g_string_new("");
	currSv->ip = g_string_new("");
	currSv->dc = g_string_new("");
	currSv->hosts = NULL;
}

void addNewServidor(){
    head->arrSv = g_slist_append(head->arrSv,currSv);
    initCurrSvs();
}
```

```
void addSvNome(char * str){

	currSv->nome = g_string_append(currSv->nome,str);
}

void addSvIP(char * str){
	currSv->ip = g_string_append(currSv->ip,str);
}

void addSvDC(char * str){
	currSv->dc = g_string_append(currSv->dc,str);
}

void addSvHost(char * str){
	currSv->hosts = g_slist_append(currSv->hosts,str);
}
FILE* fp;
// json
void printComma() {
	fputs(",\n",fp);
}

void printCloseJsonBracket(int end) {
	if(end == 1)
	fputs("}\n",fp);
	else fputs("},\n",fp);
}
```

```
void printTitle() {
	char *cat;
	cat = g_string_free(head->titulo, FALSE);
	fputs("	\"title\" : \"",fp);
	fputs(cat,fp);
	fputs("\",\n",fp);

}

void printDatabase(DataBase db) {
	fputs("	\"database\":",fp);
	printOpenJsonBracket();
	char *cat;
	cat = g_string_free(db->serverIp, FALSE);
	fputs("	\"server\": \"",fp);
	fputs(cat,fp);
	fputs("\"",fp);
	printComma();
	fputs("	\"connection_max\": ",fp);
	fprintf(fp,"%d",db->connect_max);
	printComma();
	cat = g_string_free(db->enabled, FALSE);
	fputs("	\"enabled\": \"",fp);
	fputs(cat,fp);
	fputs("\"",fp);
	printComma();
	fputs("	\"ports\": [\n",fp);
	for(int i=0; i < db->portsNumber; i++){
	if(i == db->portsNumber-1) {
	fputs("	  ",fp);
```

```
void printOwner(Owner owner) {
	fputs("	\"owner\": ",fp);
	printOpenJsonBracket();
	char *cat;
	cat = g_string_free(owner->name, FALSE);
	fputs("		\"name\": \"",fp);
	fputs(cat,fp);
	fputs("\"",fp);
	printComma();
	cat = g_string_free(owner->date, FALSE);
	fputs("		\"date\": \"",fp);
	fputs(cat,fp);
	fputs("\"",fp);
	printComma();
	cat = g_string_free(owner->time, FALSE);
	fputs("		\"time\": \"",fp);
	fputs(cat,fp);
	fputs("\"\n",fp);
	printCloseJsonBracket(0);

}

void printServer(Servers serv) {
	char *cat;
	cat = g_string_free(serv->ip, FALSE);
	fputs("			\"ip\": \"",fp);
	fputs(cat,fp);
	fputs("\"",fp);
	printComma();
	cat = g_string_free(serv->dc, FALSE);
```

```
        }
        for(int i = 0; i<len; i++) {
                if(len-i == 1){
                fputs("              ",fp);
                fputs("\"",fp);
                fputs((char*)g_slist_nth_data(serv->hosts,i),fp);
                fputs("\"\n",fp);
                }else {
                        fputs("              ",fp);
                        fputs("\"",fp);
                        fputs((char*)g_slist_nth_data(serv->hosts,i),fp);
                        fputs("\",\n",fp);
                }
        }
        if(len>0)
        fputs("           ]\n",fp);
        else
        fputs("\n",fp);

}


void printDatabases() {

        int len = g_slist_length(head->arrDB);
        for(int i = 0; i<len; i++) {
                printDatabase(g_slist_nth_data(head->arrDB,i));
        }
}
```

```
        }
        for(int i = 0; i<len; i++) {
                if(len-i == 1){
                fputs("              ",fp);
                fputs("\"",fp);
                fputs((char*)g_slist_nth_data(serv->hosts,i),fp);
                fputs("\"\n",fp);
                }else {
                        fputs("              ",fp);
                        fputs("\"",fp);
                        fputs((char*)g_slist_nth_data(serv->hosts,i),fp);
                        fputs("\",\n",fp);
                }
        }
        if(len>0)
        fputs("           ]\n",fp);
        else
        fputs("\n",fp);

}


void printDatabases() {

        int len = g_slist_length(head->arrDB);
        for(int i = 0; i<len; i++) {
                printDatabase(g_slist_nth_data(head->arrDB,i));
        }
}
```

```
void openFile(char * f){
    fp = fopen(f,"w+");
}


void formatToJson() {
       printOpenJsonBracket();
       printTitle();
       printDatabases();
       printOwners();
       printServers();
       printCloseJsonBracket(1);
}
```

```
%union{char* nome; int numero;}

%token TITLE DATABASE SERVERS OWNER

%token NAME DATE TIME

%token SERVER PORTS CONNECT_MAX DB_ENABLED

%token IP DC HOSTS

%token <nome> texto
%token <nome> SUBSERVER
%token <numero> num

%%
Toml : Titulo Body
     | Body
     ;

Titulo : TITLE '=' texto {head->titulo = g_string_append(head->titulo,$3);}
       ;

Body : Body Struct
     | Struct
     ;


Struct : '[' ']'
       | '[' OWNER ']' OWNER_Struct {addNewOwner();}
       | '[' DATABASE ']' DBStruct {addNewDB();}
       | '[' SERVERS ']' SvStruct
       ;
```

```
OWNER_Struct : OWNER_Struct OWNER_Elem
             | OWNER_Elem
             ;

OWNER_Elem: NAME '=' texto {addOwnerName($3);}
          | DATE '=' texto {addOwnerDate($3);}
          | TIME '=' texto {addOwnerTime($3);}
          ;

DBStruct : DBStruct DBElem
         | DBElem
         ;

DBElem : SERVER '=' texto {addDbSvIp($3);}
       | PORTS  '=' Array {;}
       | CONNECT_MAX '=' num {addDbConnectMax($3);}
       | DB_ENABLED '=' texto {addDbEnabled($3);}
       ;

SvStruct : SvStruct SvComp
         | SvComp
         ;

SvComp : SUBSERVER LSvELem {addSvNome($1);addNewServidor();}
       | SUBSERVER {addSvNome($1);addNewServidor();}
       ;

LSvELem : LSvELem SvELem
        | SvELem
        ;
```

```
SvELem : IP '=' texto {addSvIP($3);}
       | DC '=' texto {addSvDC($3);}
       | HOSTS '=' Array
       ;

Array : '[' ']'
      | '[' ArrayElem ']'
      ;


ArrayElem : texto ',' ArrayElem {addSvHost($1);}
          | num ',' ArrayElem {addDbPort($1);}
          | texto {addSvHost($1);}
          | num  {addDbPort($1);}
          ;

%%
```

```c
int yyerror(char *s) {
    printf("ERRO: %s\n",s); return(0);

}

int main(){
    head = newToml();
    initCurrOwner();
    initCurrDB();
    initCurrSvs();
    yyparse();

    openFile("output.json");
    formatToJson();
    return 0;
}
```