Escola de Engenharia
Universidade do Minho

DEPARTAMENTO DE ENGENHARIA INFORMÁTICA
**Mestrado Integrado em Engenharia Informática**

# Trabalho Final - Rede Social

*Processamento e Representação de Conhecimento*

Rui Vieira - A74658

Tiago Baptista - A75328

Braga, 31 de Janeiro de 2020

# Conteúdo

**1 Introdução** **2**

**2 Opções de Desenvolvimento** **3**

**3 Base de Dados - MongoDb** **4**
3.1 Caraterização das Coleções e Modelos . . . . . 4

**4 Autenticação** **6**

**5 Gramática** **7**

**6 Conclusão** **8**

# 1. Introdução

O trabalho descrito no presente documento foi realizado no âmbito da unidade curricular de Processamento e Representação de Conhecimento teve como principal objetivo a criação de uma rede social para os estudantes de Engenharia Informática da Universidade do Minho. A referida aplicação *web* deve funcionar *online* contendo um perfil de utilizador, interações entre os utilizadores (eventos, amizades, entre outros), um *feed* de *posts*. Todos estes componentes possuem operações CRUD (create, read, update and delete), facilmente detetáveis pelo utilizador. A arquitetura da aplicação foi baseada numa arquitetura MERN (Mongo, Express, Reactjs, Nodejs), tal como apresentado na Figura 5.1.

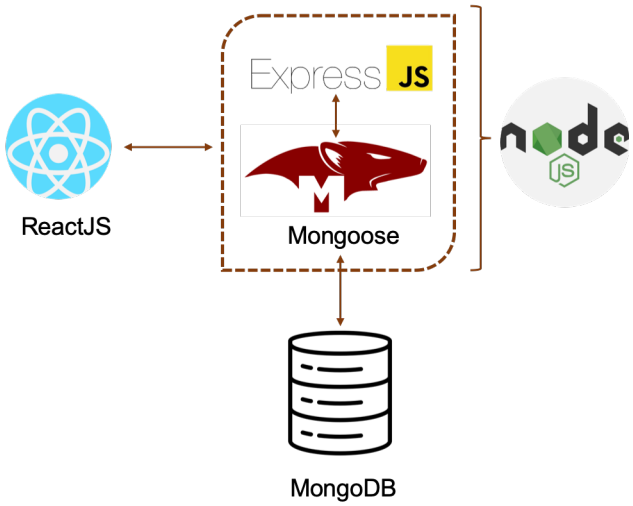


**Figura 1.1:** Arquitetura da Aplicação Desenvolvida.

# 2. Opções de Desenvolvimento

Aquando o inicio do projeto existiram algumas decisão importantes, com necessidade de serem tomadas, dentro do grupo de trabalho. Assim sendo, o grupo optou por desenvolver o *frontend* em ReactJs com o auxilio do antD possibilitando a aprendizagem de ferramentas não utilizadas durante a unidade curricular. As restantes decisões tomadas foram relativas ao *backend*, desenvolvido em Node Js com auxílio da *frameword* Express Js, e à base de dados, optando-se pela solução do MongoDB. Para terminar, foi definido que existiriam dois tipos de utilizadores, Docentes e Alunos, com duas diferenças primordiais. A primeira é a reputação dada a um *post* que seja efetuado pelo docente, a segunda e que não foi possível de realizar seria a permissão exclusiva de criar alguns tipos de eventos (testes, aulas de dúvidas,...).

# 3. Base de Dados - MongoDb

Tal como já referido, a aplicação relatada no presente documento teve por base uma base de dados em *MongoDB*. Vários modelos foram definidos e guardados em coleções distintas. Resumindo, existe uma coleção por cada tipo de dados. Para garantir a integridade dos dados, cada coleção possui um controlador único.

## 3.1 Caraterização das Coleções e Modelos

- USERS - Contém todas as informações necessária para o utilizador. É composto por:
  - email;
  - password - password encriptada por bcrypt;
  - username;
  - age;
  - position - descrição do cargo, por exemplo : Aluno de Mestrado de Engenharia Informática (4$^{o}$ano);
  - type - tipo de utilizador;
  - photo - fotografia representativa do utilizador;
  - friends - amizades aceites pelos 2 utilizadores envolvidos;
  - unconfirmed - amizades em que um dos 2 utilizadores em questão ainda não aceitou.
- POSTS
  - title;
  - owner - quem realizou o post;
  - content - conteúdo de texto que o post contém;
  - date - referente ao momento da publicação;
  - ownerPhoto - foto do criador do post;
  - likes - número de gostos que possui;
  - hashList - lista de hastags associada ao post;
  - file;
  - imagem.
- EVENTS
  - participation;

  - type;
  - owner;
  - date;
  - description;
  - photo - fotografia ilustrativa do evento;
  - file;
  - title;
  - UC;
- ALBUM
  - owner;
  - date;
  - description;
  - photos;
  - identification;
  - likes.
- COMENT
  - parentId - identificador do post a que o comentário está associado;
  - owner
  - description
  - date
  - likes
  - file
  - image

# 4. Autenticação

Para a autenticação utilizamos a estratégia de sessões por tokens, usando jwt, aprofundado conhecimentos para além daquilo que foi leccionado na unidade curricular. A quando do inicio da sessão é definida a duração da sessão e guardado no token o tipo de utilizador, permitindo desta forma controlar o acesso a rotas por tipo de utilizador. Necessitando apenas de acrescentar o metodo *(*verifier) como middleware nas rotas.

# 5. Gramática

Foi indicado como objetivo a criação de um método de importação de utilizadores utilizando um ficheiro.Para tal foi desenvolvida uma DSL que permite que um utilizador da plataforma efetue o seu registo preenchendo os campos de username,password e email.

De maneira a efetuar o reconhecimento e o *parsing* da informação necessária foi desenvolvida uma gramática e um visitor com recurso ao *antlr4*. O *visitor* criado limita-se a extrair os campos necessários para efetuar o registo e a guarda-los num ficheiro que posteriormente vai ser lido pelo servidor de maneira a usar esses campos para criar e guardar um novo utilizador na base de dados

Todos o processamento ocorrre em *run time* pois o *antlr* possui um *package* específico para o *nodeJS*.

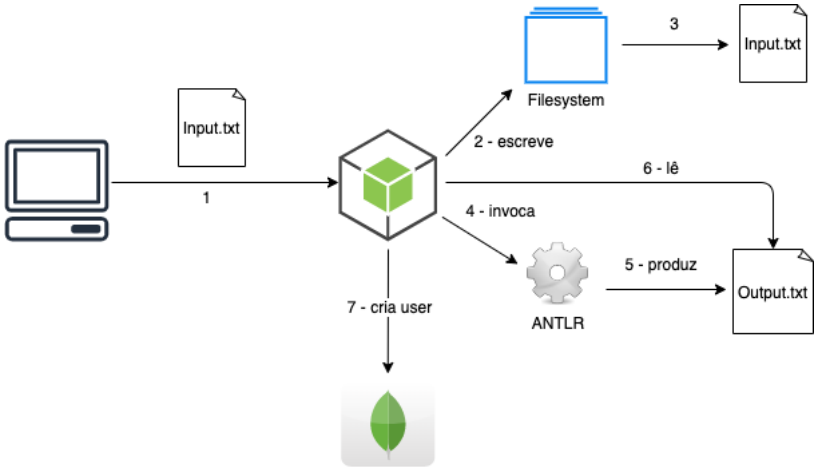


**Figura 5.1:** Funcionamento do parser.

# 6. Conclusão

Este trabalho permitiu o desenvolvimento de capacidades a nível de desenvolvimento *web* nas duas vertentes de *front end* e *back end*.

A nível de trabalho desenvolvido ficou para trás um objetivo a que nos tínhamos proposto, a criação de um *chat* de maneira a pemitir a comunicação entrer utilizadores. Não se chegou a essa meta pois decidiu-se dar prioridade a funcionalidades do *core* da aplicação, como a criação de *posts*, eventos, pedidos de amizade e o perfil do utilizador em si.