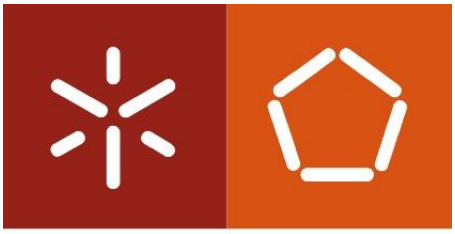

Universidade do Minho
Mestrado Integrado em Engenharia Informática

Processamento e Representação de Informação

Relatório do Trabalho Prático

João Cunha (A84775)
Luis Ramos (A83930)

7 de fevereiro de 2021

Conteúdo

Lista de Figuras

1 Introdução

1.1 Enquadramento e Contextualização

O presente relatório desenvolve-se no âmbito da Unidade Curricular de Processamento e Representação de Informação, do 4º ano do Mestrado Integrado em Engenharia Informática, tendo como principal objectivo o desenvolvimento de uma aplicação web, capaz de gerir e disponiblizar recursos educativos.

1.2 Problema e Objetivos

Os principais objetivos deste trabalho prático são logo dados no início do documento fornecido aos alunos. Por nossas palavras, os principais pontos expostos são os seguintes:

- Disponiblizar recursos educativos de vários tipos;
- Permitir adicionar novos recursos;
- Classificação de recursos por ano,tipo,tema;
- Permitir que um utilizador faça um Post sobre um recurso;
- Permitir que os utilizadores comentem os Posts;
- Permitir a votação no ranking de um recurso;
- Processo de ingestão e disseminação de recurso;

Cumpridos todos os objetivos dados no enunciado, foram ainda adicionados uns objetivos extras pelos próprios elementos do grupo.

2 Diagrama Modelo

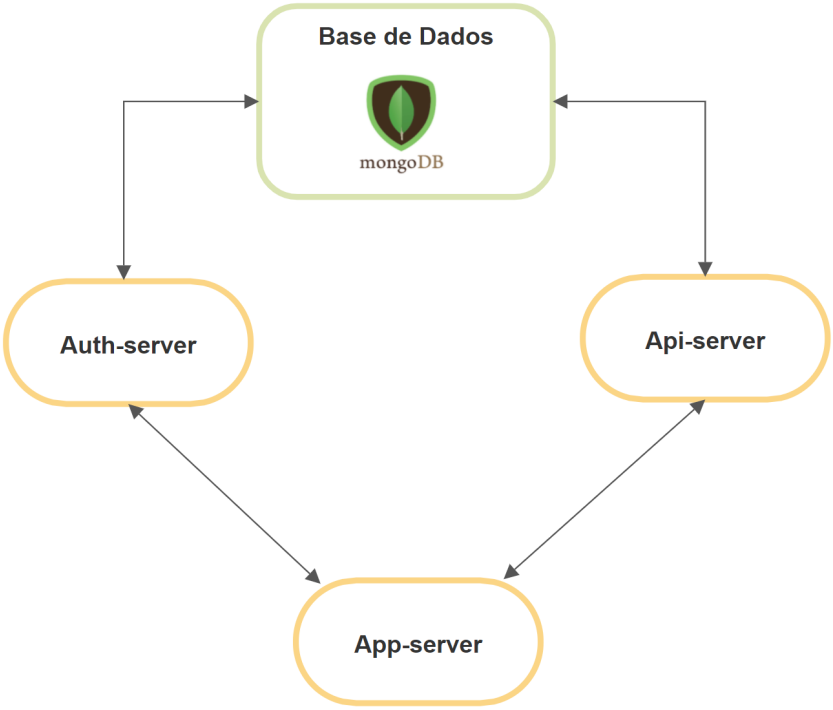

Figura 1: Diagrama Modelo do Sistema

3 Conceção/desenho da Resolução

3.1 Api-server

3.1.1 Utilizadores

O **Utilizador** está protegido com uma autênticação de username+password, e terá 3 níveis de acesso, sendo estes:

- **Administrador**: poderá efetuar todo o tipo de operações, e terá acesso a todos os recursos existentes.
- **Consumidor**: poderá descarregar e consultar recursos, mas apenas os públicos.
- **Produtor**: terá acesso apenas a recursos públicos, e também aos recursos publicados por si, podendo, ou não, realizar operações sobre o mesmo.

Será necessário guardar informação do respetivo utilizador, tal como, nome, email, filiação, password, entre outros. Para isso foi criado um schema do **Utilizador** em *mongoDB*.

```
var utilizadorSchema = new mongoose.Schema({
    nome: { type:  String, required: true },
    email: { type: String, required: true , unique: true},
    password: { type: String, required: true },
    filiacao: String,
    nivel: { type:  String, default: "consumer", required: true },
    dataRegisto: { type : Date, default: Date.now, required: true },
    dataUltimoAcesso: { type : Date },
})
```

Figura 2: Schema do utilizador

De salientar que a password é encriptada antes de ser guardada na base de dados, como iremos referir, em pormenor, mais à frente neste relatório.

3.1.2 Recursos

O **Recurso** irá conter muita informação, tal como, informação do seu produtor, conteudo do próprio recurso, e também o local onde está guardado, para ser possível ao utilizador, efetuar o download do mesmo.

Será então necessário guardar essa informação do recurso na base de dados. Para isso foi criado um schema do **Recurso** em *mongoDB*.

```
var recursoSchema = new mongoose.Schema({
    tipo: { type: String, required: true },
    titulo: { type: String, required: true },
    subtitulo: String,
    dataCriacao: { type : Date},
    dataRegisto: { type : Date, default: Date.now, required: true },
    ranking: {
        rating : Number,
        classf : [mongoose.ObjectId]
        },
    visibilidade: { type: Boolean, required: true },
    produtor: {
        nomeP : { type: String, required: true },
        emailP : { type: String, required: true }
        },
    path:{ type: String, required: true}
})
```

Figura 3: Schema do recurso

De salientar que caso o recurso inserido seja privado, ou seja, a visibilidade seja o valor **false**, apenas terá acesso a esse recurso um utilizador com o nivel de "administrador", ou então o próprio produtor do recurso.

3.1.3 Posts

O **Post** irá conter alguma informação, tal como, informação do post, informação do seu autor, informação do recurso em questão, e também irá conter uma lista de comentários.

Será então necessário guardar essa informação do post na base de dados. Para isso foi criado um schema do **Post** em *mongoDB*.

```
var postSchema = new mongoose.Schema({
    titulo: { type: String, required: true },
    descricao: { type : String, required: true },
    dataRegisto: { type : Date, default: Date.now, required: true },
    autor: {
        nomeA :  { type : String, required: true },
        emailA : { type : String, required: true }
        },
    rec:{
        idRec: { type : mongoose.ObjectId, required: true},
        titRec: {type : String, required: true}
    },
    comentarios: [{
        emailC   :  { type :  String, required: true},
        nomeC : { type :  String, required: true},
        dataC : { type : Date, default: Date.now, required: true },
        comentario: String
        }]
})
```

Figura 4: Schema do post

De salientar que qualquer utilizador poderá comentar o post.

3.2 App-server

O **App-server** é o front-end da aplicação. Esta irá comunicar com o utilizador e irá gerir as páginas da aplicação, consoante o pedido realizado pelo utilizador.

Este comunica sempre com o **Api-server** de modo a efetuar pedidos de busca à base de dados, recebendo assim a informação necessária para mostrar ao utilizador no browser, através de *views*, realizadas em *pug*.

3.3 Auth-server

O **Auth-server** tem como objetivo gerir a autênticação dos utilizadores, e realiza a operação da geração de tokens de sessão para cada um.

O utilizador insere o email e a *password*, de modo a iniciar a sua sessão. O **App-server** comunica com o **Auth-server** enviando lhe os dados. Este verifica se o email existe na base de dados, e caso exista, compara as *passwords*. Se tudo corresponder corretamente, o utilizador é autenticado e é gerado um token de sessão para o mesmo, obtendo assim o acesso à aplicação.

De notar que para ser possível comparar as paswords é necessário aceder à *hash* da *password* guardada base de dados, *hash* essa que foi gerada aquando o registo do utilizador no **App-server**.

4 Conclusão

Culminada a elaboração deste trabalho prático, importa referir que a execução do mesmo permitiu aos elementos do grupo compreender o processo de criação de uma aplicação. A forma como uma app se organiza e a forma como esta funciona, consecutivamente na ligação do back-end com o front-end.

A execução desta fase do trabalho prático permitiu uma melhor consolidação dos construtos teóricos e práticos através da utilização do mongoDB, node.js e pug.

No ímpeto geral o desenvolvimento desta fase do trabalho decorreu como planeado, alcançando os objectivos delineados pelo enunciado.