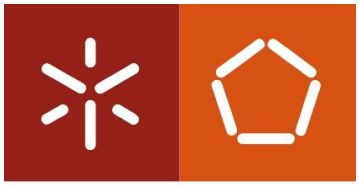

Universidade do Minho

Mestrado Integrado de Engenharia Informática

Processamento e Representação de Informação (1ºSemestre - 2020/21)

Relatório PRI - TP

A85813 António Alexandre Carvalho Lindo

A85400 Nuno Azevedo Alves da Cunha

A85919 Pedro Dias Parente

Braga

7 de fevereiro de 2021

Conteúdo

Capítulo 1

Introdução

Este trabalho teve como objetivo o desenvolvimento de um sistema de gestão de recursos que permitisse disponibilizar vários tipos de recursos educativos (upload e download), adicionar mais tipos e recursos, fazer posts de modo a os utilizadores poderem comentar os vários recursos, um sistema de ranking para os vários recursos etc.

Capítulo 2

Arquitetura de Dados

Neste trabalho utilizamos 4 estruturas de dados para armazenar as informações do sistema: User, Recurso, Post, Comment. Nas seguintes secções são analisadas em detalhe todas estas estruturas.

2.1 User

A estrutura de dados User, guarda informações sobre um dado utilizador do sistema e tem a seguinte estrutura:

```
var userSchema = new mongoose.Schema({
    id: { type: String, required: true, unique: true},
    password: { type: String, required: true},
    nome: { type: String, required: true},
    email: { type: String, required: true},
    filiacao: String,
    age: { type: Number, required: true},
    bio: String,
    access: { type: String, required: true},
    dataRegisto: { type: Date, required:true},
    dataUltimoAcesso: Date
})
```

Figura 2.1: Estrutura de dados User

- Id: Guarda o username do utilizador (o que ele utiliza para o login);
- Password: Guarda a password do utilizador;
- Nome: Guarda o nome do utilizador;
- Email: Guarda o email do utilizador;
- Filiacao: Guarda a filiação do utilizador (professor, docente etc);
- Age: Guarda a idade do utilizador;
- Bio: Guarda uma biografia do utilizador que o mesmo pode escolher escrever;
- Access: Guarda o nível de acesso do utilizador (administrador ou utilizador);
- DataRegisto: Guarda a data de registo do utilizador no sistema;
- DataUltimoAcesso: Guarda a data de último acesso do utilizador ao sistema;

2.2 Recurso

A estrutura de dados Recurso, guarda informações sobre um dado recurso do sistema e tem a seguinte estrutura:

```
var recursoSchema = new mongoose.Schema({
    tipo: { type:String, required:true },
    titulo: { type: String, required: true},
    dateCreation: { type: Number, required: true},
    dataRegisto: { type: Date, required: true},
    visibilidade: { type: String, required: true},
    autor: { type: String, required: true},
    ratings: [
        {
            rating: Number,
            user: String
        }
    ]
})
```

Figura 2.2: Estrutura de dados Recurso

- Tipo: Guarda o tipo do recurso;
- Titulo: Guarda o título do recurso;
- dateCreation: Guarda o ano de criação do recurso;
- dataRegisto: Guarda a data de inserção no sistema do recurso;
- Visibilidade: Guarda o tipo de visibilidade do recurso (público ou privado);
- Autor: Guarda o username do autor do recurso;
- Ratings: Array que guarda objetos com o username do utilizador que deu rate ao recurso e o respetivo valor númerico desse rate;

2.3 Post

A estrutura de dados Post, guarda informações sobre um dado post do sistema e tem a seguinte estrutura:

```
var postSchema = new mongoose.Schema({
    titulo: { type: String, required: true},
    autor: { type:String, require: true},
    conteudo: { type:String, require:true},
    visibilidade: { type: String, required:true},
    upvotes: [String],
    dataRegisto: { type: Date, required: true},
    recursoID: { type: mongoose.Schema.Types.ObjectId, require:true},
    tipo: { type: String, required:true },
    recTitle: { type:String, required:true }
})
```

Figura 2.3: Estrutura de dados Post

- Titulo: Guarda o título do post;
- Autor: Guarda o username do autor do post;
- Conteudo: Guarda o conteudo do post;
- Visibilidade: Guarda o tipo de visibilidade do post (público ou privado);
- Upvotes: Guarda um array com os usernames dos utilizadores que deram like no post;
- dataRegisto: Guarda a data de inserção do post;
- RecursoID: Guarda o id do recurso ao qual o post é relativo;
- Tipo: Guarda o tipo de recurso ao qual o post é relativo;
- RecTitle: Guarda o título do recurso ao qual o post é relativo;

2.4 Comment

A estrutura de dados Comment, guarda informações sobre um dado comentário do sistema e tem a seguinte estrutura:

```
var commentSchema = new mongoose.Schema({
    user: {type:String, required:true},
    upvotes: [String],
    comment: { type:String, required:true},
    dataComment: { type: Date, required: true},
    postID: { type: mongoose.Schema.Types.ObjectId, require:true}
})
```

Figura 2.4: Estrutura de dados Comment

- User: Guarda o username do autor do comentário;
- Upvotes: Guarda um array com os usernames dos utilizadores que deram like no comentário;
- Comment: Guarda o comentário em si (conteúdo);
- DataComment: Guarda a data em que o comentário foi postado;
- PostID: Guarda o o id do post ao qual o comentário é relativo;

Capítulo 3

Arquitetura de Sistema

Na sua implementação deste trabalho o grupo implementou a seguinte arquitetura de Sistema:

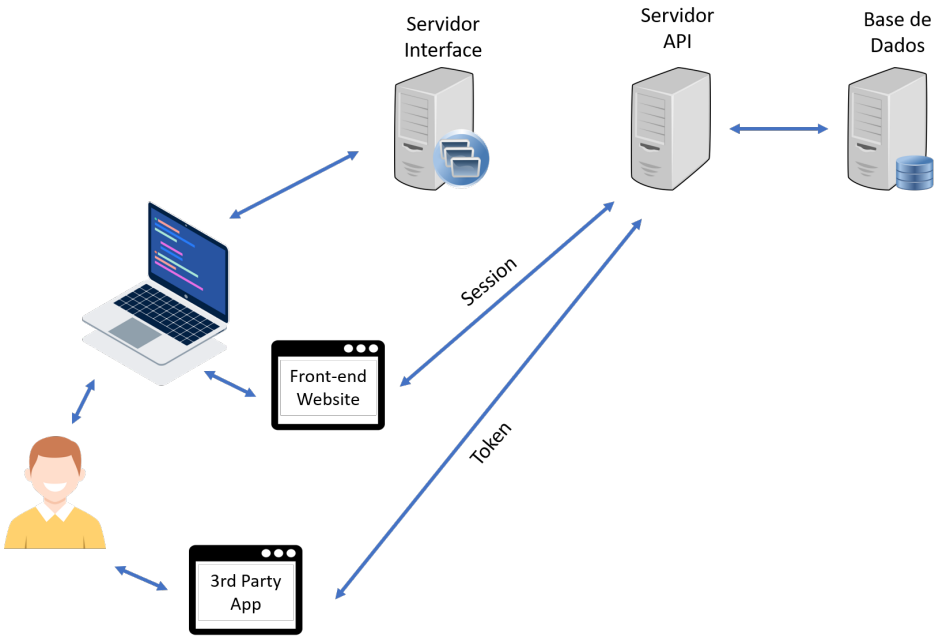

Figura 3.1: Arquitetura Aplicacional

Como se pode ver na imagem é disponibilizado um servidor de interfaces que devolve o front-end da aplicação e um servidor de API que lida com as operações sobre os dados da mesma.

Ambos os servidores existem independentemente um do outro, sendo que o servidor de API pode ser utilizadado por aplicações 3rd party através de uma autenticação por JWT. O servidor de interface gera páginas web que fazem chamadas para a API de dados autorizadas através de um sistema de sessões, os tokens podem ser gerados nestas páginas através de uma conta que esteja logged in por sessão.

Ou seja, há duas formas de aceder aos dados da aplicação, é possivel utilizar o website para servir de intermédio com a api ou obter um token e chamar a API diretamente.

Capítulo 4

Conclusão

Em conclusão, pensamos que construímos uma gestor de recursos com todas as funcionalidades essenciais para o funcionamento do mesmo. As principais dificuldades que enfrentamos foram a organização e escolha da arquitetura dos dados e também o upload dos ficheiros com geração de zip, manifesto e metadata.

Em futuras iterações do trabalho, gostaríamos de desenvolver mais funcionalidades, tais como login com Facebook e Google ou a colocação de fotos nos perfis dos utilizadores.