Universidade do Minho

Departamento de Informática

MEI - Mestrado em Engenharia de Informática

Representação e Processamento de Conhecimento na Web

Maria Beatriz Lacerda A89535

Maria Barros PG47488

Renata Teixeira PG47603

30 de Junho
2021/2022

Conteúdo

Parte I
Introdução

No âmbito da unidade curricular de Representação e Processamento de Conhecimento na Web foi nos proposto o desenvolvimento de um projeto final que consiste num sistema de gestão e armazenamento de recursos didáticos. Deste modo, definifo como objetivos a criação de uma aplicação *web* na qual:

- Seja possível um utilizador se registar e autenticar
- Seja suportados três tipos de utilizadores com diferentes níveis de acesso
- Seja possível a inserção de recursos
- Seja possível a consulta, filtragem e edição desses mesmos recursos
- Seja possível o *download* e a *preview* dos recursos
- Haja um painel de adminstração através do qual haja gestão de utilizadores, recursos e estatísticas
- Seja possível a interação entre utilizadores e recursos através de comentários e atribuições de *likes*

Para atingir estes objetivos foram utilizadas as várias ferramentas com as quais o grupo se foi familiarizando ao longo do semestre tais como Postman, a *framework* Express.js e Node.js. Adicionalmente os dados da aplicação foram armazenados recorrendo ao MongoDB.

1 Estrutura do Relatório

O presente relatório encontra-se dividido em 4 capítulos. No **primeiro capítulo**, abordamos o problema de forma geral e apresentamos os objetivos que procuramos cumprir ao longo da realização deste projeto. No **segundo capítulo** é apresentada a estrutura da aplicação desenvolvida, bem como os atores desta aplicação. No **terceiro capítulo** expomos as funcionalidades implementadas e, finalmente, no **quarto capítulo** fazemos uma breve conclusão e análise crítica do trabalho conseguido.

Parte II
Arquitetura Aplicacional

2 Arquitetura do Sistema

A nossa aplicação *web* é formada por 3 servidores:

- **App server**
  O *app server*, que corre na porta 3002, é responsável pela interação da aplicação com o cliente. Assim sendo, é neste servidor em que se encontram as *views* em *pug*. Será este o servidor que receberá os pedidos dos clientes e que irá comunicar com os outros servidores de modo a obter a informação necessário para o processamento do pedido.
- **Api Server**
  É o servidor responsável pela gestão e armazenamento dos ficheiros bem como dos comentários do aplicação. Este armazenamento é feito recorrendo à base de dados MongoDB.
- **Auth Server**
  Servidor responsável pela gestão e armazenamento dos utilizadores da aplicação utilizando, novamente, o MongoDB.

Relativamente à base de dados, será, portanto conectada quer ao *api-server* quer ao *auth server* e terá três *collections*: users, ficheiros e comentarios.

3 Atores do Sistema

A aplicação compreende três tipos de utilizadores diferentes aos quais correspondem diferentes níveis de acesso às diferentes funcionalidades. Este nível é definido pelo campo "level", presente em cada utilizador guardado na base de dados.

Aquando o registo na aplicação, todos os utilizadores assumem automáticamente um nível de "Consumidor". Mais tarde, o administrador porderá, na gestão de utilizadores, alterar os vários níveis de permissão de cada utilizador, isto é, tornar um "ConsumidorProdutor"e vice-versa.

3.1 Consumidor

Este utilizador pode apenas consultar os recursos já pré-existentes no repositório, tendo apenas acesso às páginas e funcionalidades de consulta e à sua página de perfil. Poderá, ainda, interagir com os recursos, comentando, atribuindo *likes* aos vários recursos e fazendo *download* destes.

3.2 Produtor

O produtor para além das funcionalidades de pesquisa e consulta às quais o "Consumidor"tem acesso, terá ainda a possibilidade de acrescentar novos recursos e ainda de editar os recursos que submeteu na plataforma.

3.3 Administrador

Um utilizador com "level"equivalente a "Admin"terá acesso a todas as funcionalidades anteriores, mas será, ainda, responsável pela gestão da aplicação. Relativamente à gestão de recursos, o administrador poderá eliminar ou editar todos os recursos da plataforma, bem como comentários em recursos. Poderá ainda alterar os níveis de permissão ou eliminar qualquer utilizador da aplicação. Adicionalmente, terá acesso ao ficheiros dos *logs*, bem como permissão para dar *reset* a este ficheiro.

4 Autenticação e Autorização

O processo de autenticação foi implementando no *app server*, recorrendo ao *middleware passport*. Assim, através da *LocalStrategy*, confirmamos se as credenciais introduzidas pelo utilizador correspondem aos dados de alguma conta existente na base de dados. Caso os dados existam, o utilizador é autenticado.

Esta estratégia é invocada na rota de *login*. Se o utilizador for autenticado, será então, gerado um *token*. Este *token* será enviado na *query string* dos pedidos feitos à API. O utilizador terá acesso à informação se o *token* enviado for verificado. Esta verificação é, por sua vez, feita no *api server*.

Parte III
Funcionalidades do Projeto

5 Homepage

A *homepage* do nosso projeto fornece a informação ao utilizador dos recursos adicionados desde o seu último *logout*.

Para este propósito, é feita uma recolha na base de dados de todos os recursos adicionados por qualquer utilizador, exceto ele próprio, desde a data do seu último *logout*.

Isto permite que enquanto o utilizador estiver com a sessão iniciada, sempre que este visitar a *homepage*, poderá ser informado de recursos que tenham sido adicionados por outros utilizadores não só antes, mas também durante a sessão.

6 Inserção de Ficheiros

A nossa aplicação está preparada para receber ficheiros comprimidos em formato ZIP. Estes ficheiros devem cumprir certas regras específicas, nomeadamente a inclusão de, pelo menos, estes dois ficheiros:

- Um ficheiro (manifestFile.txt) que contem todos os ficheiros existentes no ZIP.
- Um ficheiro RRD-SIP.json que contem os metadados do ficheiro submetido. Estes metadados incluem o título, o tipo de recurso, o criador e a data de criação do recurso.

Assim, são necessárias algumas verificações para ter a certeza que os ficheiros têm todos os elementos para que possam ser aceites e armazenados na aplicação. A primeira verificação passa por verificar se o ficheiro RRD-SIP.json existe de facto e se, caso exista, tem todos os metadados necessários já mencionados anteriormente. Caso o ficheiro passe esta primeira verificação, passamos a *flag* "aceite"para *true*.

De seguida, confirmamos a existência do ficheiro manifestoFile. Se este ficheiro existir, então passamos a *flag* "manifesto"para true. Verificamos, ainda, se o ficheiro se encontra completo, isto é, se os ficheiros que ele referencia de facto existem dentro do ZIP submetido.

Se todas estas condições forem cumpridas passamos o conteudo do RRD-SIP.json para a base de dados através de um pedido POST para a API. Adicionalmente, guardamos o ficheiro no *filesystem* da aplicação. O *filesystem* está organizado por tipos sendo as categorias existentes "Manuais", "Testes", "Fichas"e "Slides".

7 Disponibilização de Recursos

7.1 Listagem de Recursos

Quando um utilizador visita a página de recursos, são-lhe apresentados todos os ficheiros armazenados na aplicação naquele momento.

Para tal, é feito um pedido dos recursos à API, onde são recolhidos os dados de cada ficheiro ordenados pela data de submissão, sendo estes, o seu título, a sua data de criação, a data de submissão, o seu produtor, o nome do utilizador que o submeteu, o seu tipo e o seu número de *likes*.

É também possível fazer **listagem** de recursos **por tipo**, sendo que neste caso é feito um pedido à API, sendo recolhida apenas a informação relativa aos recursos do tipo especificado e sendo estes listados.

Por último, é ainda praticável **pesquisar por título na *search bar***. Neste caso, é feito um pedido à API que devolve a informação dos recursos que contenham o pesquisado no seu título, sendo apenas estes listados.

7.2 Edição de um Recurso

A edição de um recurso é apenas permitida ao produtor que o submeteu ou a um administrador, permitindo alterar a data de criação, o produtor, o título e o tipo.

Para a edição do recurso é feito um pedido à API que utiliza os dados escritos no form (que podem ou não ter sido alterados) para atualizar o respetivo recurso.

Durante a edição do recurso é também alterado o RRD-SIP com a nova metainformação de forma a que ao se fazer download deste recurso, tudo esteja atualizado.

7.3 Eliminar um Recurso

De modo a eliminarmos um ficheiro é, primeiramente, feita a verificação de que se trata de um administrador a realizar a ação. Posteriormente, através do "id"do ficheiro fazemos um pedido GET à API. Este pedido tem o propósito de obter o tipo e o nome do ficheiro que queremos eliminar, dado que precisamos dessa informação para o identificarmos no *filesystem*. Assim sendo, e já com o ficheiro devidamente identificado, eliminamos o recurso do *filesystem* e apagamos, ainda, os seus dados da base de dados bem como todos os comentários feitos a este recurso.

7.4 Download um Recurso

Ao fazer *download* de um recurso, o utilizador descarrega todo o conteúdo do ZIP. Para isto existe associado a cada recurso um *link* que processa esse mesmo download na rota adequada. Esse *link* já contém os dados necessários para a localização do recurso no *filesystem*.

7.5 Consulta de um Recurso

De modo a consultar um recurso, é possível clicar no respetivo título deste na página de listagem de todos os recursos, que é um *link* que irá redirecionar para a página desse mesmo recurso onde são listados, a partir de pedidos à API, não só a metainformação, mas também o conteúdo do ficheiro, que contém *links* para os vários ficheiros do ZIP, os *likes* dados nesse recurso, os comentários feitos e onde é também possível comentar e dar *like* no recurso.

7.5.1 Atribuição de *Likes* e Comentários

Qualquer utilizador pode interagir com um recurso atribuindo *likes* bem como comentários. Relativamente à atribuição de *likes*, cada ficheiro na base de dados tem, como membro, uma lista "likedby". Nesta lista estarão presentes os *usernames* dos utilizadores que gostaram desse recurso. Assim sendo, quando clicamos no botão com a intenção de gostar de um recurso fazemos a verificação de se o utilizador se encontra nessa lista. Caso se encontre, então eliminamos o seu *username* dessa lista e removemos o *like*. Caso contrário, adicionamos um like à base de dados e o *username* do utilizador à lista.

Quanto aos comentários, cada página de recurso terá uma caixa de texto destinada para o fazer. Após a escrita do comentário, o utilizador terá que clicar no botão para o submeter. Nesta submissão, adicionamos o comentário introduzido na base de dados, na coleção dos comentários através de um pedido POST à API. Estes comentários poderão ser eliminados por um administrador.

7.5.2 Visualização do Conteúdo de um Recurso

Quando um utilizador pretende ver algum ficheiro pertencente ao conteúdo dum certo recurso sem fazer o download do recurso, é possível, na página do respetivo recurso, selecionar o respetivo *link* do ficheiro que pretende ver, que o redireciona para uma página com o conteúdo desse ficheiro.

Para tal, são extraídos do *zip* todos os ficheiros para uma pasta, e é selecionado dessa pasta o ficheiro pretendido.

8 Página de Perfil do Utilizador

Cada utilizador terá acesso a uma página de perfil onde poderá consultar a informação dele presente na base de dados, o que inclui, por exemplo, o seu nível de acesso à aplicação e às suas funcionalidades.

Para além disto, poderá também alterar a sua *password* e consultar os recursos aos quais atribuiu um *like*.

9 Funcionalidades de Administrador

9.1 Gestão de Utilizadores

O administrador pode alterar as permissões de visualização dos utilizadores (produtor ou consumidor), e eliminar utilizadores. Para o efeito, são feitos pedidos ao *auth-server*, de forma a alterar o nível dos respetivos utilizadores ou eliminar o registo do utilizador.

9.2 Ficheiro de Logs

De forma a coletar a informação que pretendemos utilizamos o módulo *winston*. Este módulo, permite a personalização de *logs* bem como a sua recolha para, por exemplo, um ficheiro, que é no nosso caso o "UserInteraction.log".

Com isto em vista, os eventos que procuramos marcar são os de início de sessão, a visualização de um recurso ou o download de um recurso por parte de um utilizador. Assim sendo, sempre que um utilizador acede à rota "/login", "/recursos/see/:id"e "/download/:tipo/:fnome", será acresentado ao ficheiro um objeto JSON com a data da ação, o utilizador que a realizou e o recurso envolvido. O administrador poderá ter acesso a estes *logs* através do *download* do ficheiro "UserInteraction.log"e poderá, ainda, dar *reset* a este mesmo ficheiro.

Parte IV
Conclusão

Este projeto teve como propósito o desenvolvimento de uma aplicação *web* que consiste num repositório de documentos didáticos.

Durante a realização deste trabalho tivemos a oportunidade de aplicar os vários conhecimentos adquiridos durante este semestre como autenticação, MongoDB e Postman bem como expandimos ainda mais esse conhecimento de modo a ultrapassar eventuais obstáculos.

Apesar de nem todas as funcionalidades terem sido implementadas da forma como pretendíamos, pensamos que os maiores objetivos do trabalho forma alcançados e, portanto, terminamos este projeto com um balanço geral positivo.