



Universidade do Minho
Escola de Engenharia

UNIVERSIDADE DO MINHO

DEPARTAMENTO DE INFORMÁTICA

Representação e Processamento de Conhecimento na
Web

Projeto Final

Relatório de Desenvolvimento

Sara Marques
pg47657

Maria Ramos
pg47483

Ana Ribeiro
pg47841

29 de junho de 2022

Conteúdo

1	Introdução	4
2	Arquitetura da solução	5
2.1	Arquitetura geral da aplicação	5
2.2	<i>Upload</i> de ficheiros	6
2.3	<i>Download</i> e visualização	7
3	Implementação da solução	9
3.1	Utilizadores e Funcionalidades	9
3.1.1	Autenticação	9
3.2	Upload	10
3.2.1	<i>Schemas</i>	10
3.2.2	Metainformação	11
3.3	<i>Download</i> e visualização de ficheiros	12
3.4	Página de um recurso	13
3.4.1	Reviews	13
3.5	Notícias	14
3.6	Gestão de recursos	16
3.7	Gestão de Utilizadores	18
3.8	Estatísticas de utilização	20
4	Conclusão e trabalho futuro	21

Lista de Figuras

2.1	Arquitetura da aplicação.	6
2.2	Upload de ficheiros.	7
3.1	Pedido de metainformação.	11
3.2	Exemplo de conteúdo de um SIP submetido.	12
3.3	Página de um recurso e respetivas reviews	14
3.4	Página de notícias para utilizadores	15
3.5	Página de notícias para administradores	15
3.6	<i>Form</i> para notícias	16
3.7	<i>Slider</i> de notícias	16
3.8	Opções disponíveis para um utilizador registado (sem ser administrador).	17
3.9	Modal com opções de edição de um ZIP.	17
3.10	Erro ao tentar alterar o tipo do ficheiro.	17
3.11	Modal de edição de um recurso.	18
3.12	Eliminação de uma pasta.	18
3.13	Página de gestão de utilizadores	19
3.14	Página de estatísticas de utilização	20

Capítulo 1

Introdução

Este projeto foi desenvolvido no âmbito da UC de Representação e Processamento de Conhecimento na Web do 1º ano do perfil de Engenharia de Linguagens do 1º do Mestrado em Engenharia Informática.

O projeto teve como objetivo o desenvolvimento de um Repositório de Recursos Didáticos. A aplicação desenvolvida possui funcionalidades como: autenticação de utilizadores; visualização, filtragem, edição, *download* e *upload* de recursos, possibilidade de comentar e classificar os recursos disponíveis e disponibilização de notícias. O RRD foi implementado seguindo as orientações do modelo OAIS.

Capítulo 2

Arquitetura da solução

2.1 Arquitetura geral da aplicação

A aplicação é constituída por 3 servidores distintos:

- **Autenticação** - Responsável por verificar as credenciais de acesso e de registo dos clientes. Gera *tokens* caso a autenticação seja bem sucedida. Para a verificação de credenciais e registo de utilizadores acede a uma base de dados.
- **API de dados** - Responsável por pedir, atualizar e eliminar dados da base de dados. Verifica a existência e o nível dos *tokens* e responde aos pedidos com json. Nem sempre os pedidos enviados à API de dados necessitam de *tokens*, pois existem operações disponíveis na aplicação para utilizadores não autenticados.
- **Aplicação** - Intermediário entre cliente e outros servidores. Envia pedidos aos outros servidores e responde ao cliente com HTML, CSS e Javascript.

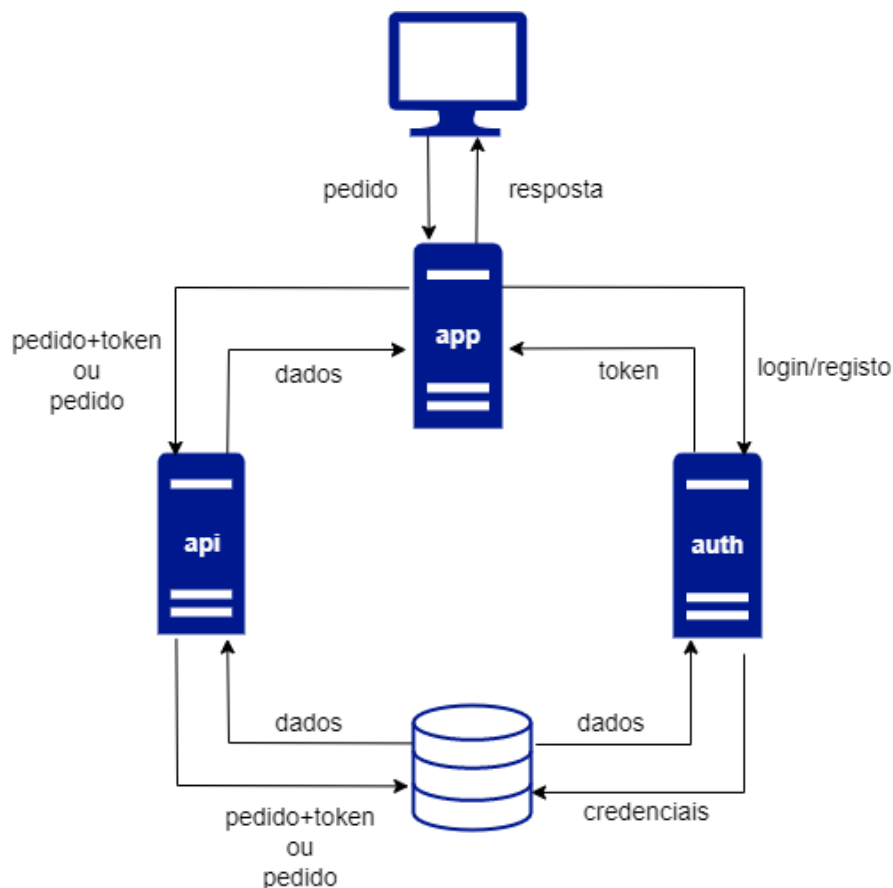


Figura 2.1: Arquitetura da aplicação.

2.2 Upload de ficheiros

Um produtor pode apenas submeter ficheiros no formato ZIP no sistema. Este ZIP deve possuir os requisitos enumerados no enunciado:

- Deve existir um manifesto em formato JSON com o nome `RRD-SIP.json`.
- O manifesto deve seguir a norma Bagit.
- Todos os outros ficheiros do pacote deverão estar ao nível do manifesto ou em subpastas e deverão ser todos referenciados por este.

Quando um ficheiro é submetido por um cliente, o servidor da aplicação irá reencaminhá-lo para o servidor da API, pois será aí que ficará armazenado. No servidor da API será feita uma verificação do *token*, visto que apenas utilizadores registados e administradores podem ser produtores. Caso o pedido de *upload* passe a verificação de *token*, o ZIP será armazenado numa pasta dedicada ao utilizador que fez *upload*. **Todos os ZIPs submetidos serão armazenados no sistema de ficheiros de acordo com o utilizador que fez o *upload*.**

Após o armazenamento do ficheiro ZIP, o servidor procede a fazer um conjunto de verificações:

- Verifica a existência de do manifesto `RRD-SIP.json`.
- Verifica o formato do manifesto.
- Verifica se o manifesto referencia todos os ficheiros do SIP e se todos os ficheiros do SIP são referenciados pelo manifesto.
- Verifica os *checksums* dos ficheiros.

Se o SIP não passar por alguma destas verificações, será enviada uma resposta negativa, com o respetivo erro, ao servidor da aplicação e, posteriormente, ao cliente. Caso contrário, será iniciado o processo de extração de metainformação e validação de ficheiros XML.

Nesta segunda etapa de processamento, é verificado o tipo de cada ficheiro presente no SIP. Caso um ficheiro seja um XML, este será validado de acordo com um *schema*. Caso não seja válido, será retornado ao servidor da aplicação um erro. Caso contrário, é extraída metainformação através da leitura de alguns campos desse XML.

Caso um ficheiro não seja um XML, será necessário pedir ao cliente metainformação relativa a esse ficheiro. Isto significa que o servidor da aplicação terá que retornar ao cliente um formulário a requisitar metainformação de todos os ficheiros presentes no SIP que não são do tipo XML. Será, então, enviada uma resposta do tipo **201 Ok** para informar que o SIP passou por todas as verificações, mas que, no entanto, é necessária informação adicional. Claro está que se o SIP contiver apenas ficheiros do tipo XML, a resposta enviada será um **200 Ok**, porque foi possível extrair e guardar na base de dados toda a metainformação necessária.

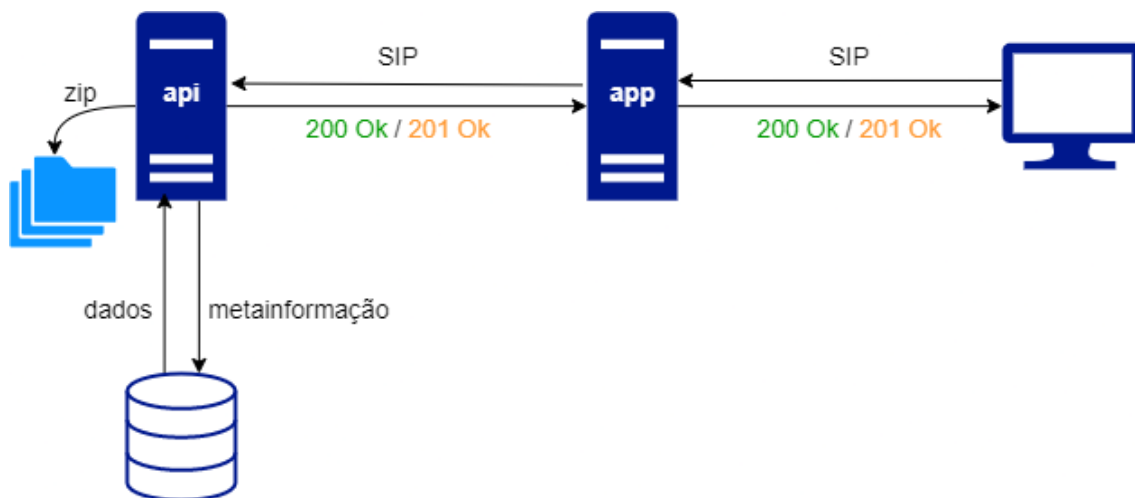


Figura 2.2: Upload de ficheiros.

2.3 *Download* e visualização

Para além de ser possível fazer *download* de um ficheiro ZIP tal como ele foi submetido, é possível fazer individualmente *download* de cada elemento que o compõe, quer de pastas, quer de ficheiros. No caso de ser feito *download* de uma pasta ou do próprio ZIP, será gerado um manifesto que descreva o seu conteúdo.

Para a visualização no *browser* dos ficheiros, é necessário que a aplicação peça ao servidor da API pelo ficheiro correspondente, ou seja, a aplicação faz um *download* da API e depois serve esse ficheiro.

Capítulo 3

Implementação da solução

3.1 Utilizadores e Funcionalidades

Um utilizador da aplicação poderá aceder à mesma como um utilizador registado (equivalente a um produtor), como um utilizador não registado (equivalente a um consumidor) ou como um administrador. Cada um destes tipos de utilizador terão diferentes permissões no que se refere ao acesso às diferentes funcionalidades da aplicação.

- **Utilizador não registado**

Pode aceder à página de notícias e ao repositório de recursos e de submissões, ordenando e pesquisando os resultados se necessitar, e realizar downloads.

- **Utilizador registado**

Pode fazer o mesmo que um utilizador não registado, e também realizar novas submissões; consultar a sua página de submissões; deixar *reviews*; editar e/ou apagar as suas *reviews*; e editar e/ou apagar os recursos submetidos pelo próprio.

- **Administrador**

Pode fazer o mesmo que um utilizador normal, e também aceder à página de estatísticas de utilização; gerir utilizadores; adicionar, editar, e/ou apagar notícias; apagar qualquer *review*; e editar e/ou apagar qualquer recurso.

3.1.1 Autenticação

A partir do servidor de autenticação foi criada uma coleção "users" na base de dados, que irá conter o *username* (único e identificador), nome, email, palavra-passe, nível (administrador ou normal) e estado (apagado ou não) de cada utilizador registado da mesma. Sendo assim, qualquer operação limitada a utilizadores registados, administradores, ou utilizadores específicos poderá ser propriamente autenticada, evitando assim que possa ser efetuada por utilizadores não autorizados a tal.

No que se refere ao método de autenticação, optamos pelo uso do passport com JSON Web Tokens (JWTs).

Cada token será, em primeiro lugar, autenticado pela *Local Strategy* do passport no momento de login, garantindo que o *username* contido neste existe na base de dados e corresponde a um utilizador não

apagado, e que a sua palavra-passe coincide com a palavra-passe introduzida. A partir desse ponto, o token - que irá incluir o *username* e o nível do utilizador a que se refere - será incluído no *query string* de todas as rotas relativas a funcionalidades que envolvam alterações na base de dados. Desse modo será possível verificar através de funções de *middleware* (recorrendo a *jwt.verify* para consultar os conteúdos da payload do jwt) se o *token* incluído num dado *request* corresponde a um utilizador ou a um tipo de utilizador registado específico, retornando erro e impossibilitando o uso da funcionalidade dessa rota caso contrário.

3.2 Upload

Na secção 2.2 já foi descrito o processo de *upload* de um ficheiro, mas nesta secção serão explicados alguns pormenores de implementação.

3.2.1 Schemas

Como já foi dito, um dos processos de validação do SIP é a validação do manifesto de acordo com um JSON *schema*. Foi utilizado o pacote *ajv* para criação do *schema* e para a respetiva validação. O manifesto deve possuir um formato semelhante ao exemplo seguinte:

```
1 [
2   {
3     "checksum": "ae9b7741ccab35b44ea2bb32f92f6427bbf1d9f9",
4     "path": "fotos/eg1.jpeg"
5   },
6   {
7     "checksum": "4dfe052902c57caa3f6086b02d7b907aee3df23c",
8     "path": "fotos/eg2.jpeg"
9   }
10 ]
```

O *checksum* é em SHA-1 e é também utilizado um pacote, chamado *checksum*, para o cálculo do *checksum* de um ficheiro.

Relativamente ao *schema* dos XML foi utilizado o pacote *libxml* para criação do *schema* e validação dos ficheiros XML. Um exemplo de XML válido é:

```
1 <RRD>
2   <meta>
3     <titulo>Teste 2020</titulo>
4     <descricao>Isto e uma descricao</descricao>
5     <data_criacao>20-05-2000</data_criacao>
6     <produtor>Maria</produtor>
7     <tipo>Teste</tipo>
8   </meta>
9   <corpo>
10    <item>exercicio 1</item>
11    <item>exercicio 2</item>
12  </corpo>
```

3.2.2 Metainformação

Como foi dito, o servidor da API responde ao servidor da aplicação com um pedido por metainformação dos ficheiros que não são XML. É gerada uma página com um conjunto de formulários, um por cada ficheiro que necessitada de metainformação.

The screenshot shows the DATALOG application interface. At the top is a teal navigation bar with the DATALOG logo and links for 'Início', 'Notícias', 'Repositório', and 'Sobre'. Below this is a large heading 'Metainformação'. A message states: 'Por favor preencha os seguintes formulários acerca dos ficheiros que fez upload. Caso não o faça estes não serão guardados.' The main content area displays a form for the file 'pasta2/pasta3/anotacoes.pdf'. The form fields are: 'Título' (filled with 'anotacoes'), 'Tipo de Recurso' (a dropdown menu currently showing 'Outro'), 'Descrição' (an empty text area), 'Data de Criação' (filled with '06/29/2022' and featuring a calendar icon), and 'Produtor' (filled with 'maria').

Figura 3.1: Pedido de metainformação.

O formulário indica a diretoria do ficheiro em causa e um conjunto de campos a serem preenchidos. A maioria dos campos já se encontra preenchida previamente para facilitar a tarefa ao utilizador. O **título** será o nome do ficheiro, sem o tipo do mesmo, a **data de criação** será a data atual e o **produtor** será o nome do utilizador que fez *upload*. Existe também um campo com o **tipo de recurso**, que possui um conjunto de opções, sendo a "Outro" a pré-definida.

No final da página existem 2 botões: **Submeter** e **Cancelar**. Se for clicado o botão de cancelamento, o ZIP será eliminado do servidor da API. Se for clicado o botão de submissão, a metainformação será, por fim, armazenada na base de dados e o manifesto será eliminado.

De modo a que seja possível navegar pelas várias pastas de um SIP submetido, foram criadas 3 coleções distintas na base de dados:

- **sips** - informação sobre o ZIP submetido, como **nome**, **descrição**, **visibilidade** (se é privado ou público), **data de submissão** e **submissor**.
- **folders** - informação sobre uma pasta de um ZIP submetido. Possui uma referência para o SIP a que pertence e uma referência para a pasta pai. Contém também um **nome** e um **path** dentro do ZIP, útil para encontrar a pasta rapidamente dentro do ZIP armazenado, para o caso de querermos, por exemplo, fazer *download* apenas dessa pasta.
- **files** - informação sobre um ficheiro. Possui uma referência para o SIP a que pertence e uma para a pasta pai. Possui também, para além das informações que são requisitadas ao utilizador ou são extraídas dos XML, um **mimetype**, **checksum** e **path**. Este último é útil, tal como no caso anterior, para encontrar facilmente o ficheiro dentro do ZIP, para o caso de *download*, edição ou eliminação do ficheiro.








3.3 *Download* e visualização de ficheiros

Na secção 2.3 foi explicado como a visualização e a transferência de ficheiros funciona na generalidade. Esta secção irá abordar alguns detalhes de implementação.

A imagem seguinte é um exemplo do conteúdo de um SIP submetido. É possível clicar nas pastas e ver o seu conteúdo e ao clicar num ficheiro iremos ver uma página dedicada a si, com informações adicionais e comentários. É também possível, como foi dito, fazer *download* quer das pastas, quer dos ficheiros, bem como visualizar no *browser* o ficheiro.

Submissões Recursos Didáticos

Download

Nome	Título	Data criação	Produtor	Tipo	
 pasta1					
 pasta2					
 ficheiro.xml	Teste	20-05-2000	Maria	Teste	 

Voltar

Figura 3.2: Exemplo de conteúdo de um SIP submetido.

Download

Ao clicar no botão de *download* o servidor da aplicação irá requisitar ao servidor da API a pasta ou ficheiro.

Na receção de um pedido de *download* de um ficheiro, o servidor da API extrai o ficheiro do ZIP guardado no disco. Para isso, é utilizado o pacote *adm-zip*. Posteriormente, envia o ficheiro e elimina o ficheiro do local para onde foi extraído. No lado do servidor da aplicação utiliza a função *download* do *express* para que ocorra o *download* do lado do utilizador.

No caso de ser feito um pedido de *download* de uma pasta, a pasta será extraída para uma diretoria gerada na altura com um nome aleatório. É criado um manifesto com todos os ficheiros que estão dentro dessa pasta, o que envolve alguma complexidade, já que é necessário encontrar todas os ficheiros dentro de todas as subpastas. A diretoria é comprimida para um ficheiro ZIP, o nome é alterado para o nome original da pasta e, após ser enviada para o servidor da aplicação, tanto o ZIP criado como a diretoria auxiliar são eliminados.

É também possível fazer *download* do ZIP original. O processo é parecido ao *download* de uma pasta. No entanto, o processo de construção do manifesto é muito mais simples, pois todos os ficheiros associados a um SIP têm uma referência para si.

Visualização

Ao clicar no botão de visualização de um ficheiro, inicia-se o processo de transferência do servidor da API para o da aplicação. O servidor da aplicação guardará o ficheiro recebido numa diretoria de recursos estáticos públicos. Assim, após finalizado o *download*, o ficheiro, caso não seja XML, é mostrado num *modal* através da *tag* HTML *object*. Se for um ficheiro XML, é exibido numa caixa de texto não editável. Isto deve-se ao facto de que com a *tag* *object*, as *tags* do ficheiro XML eram omitidas. Ao colocar numa caixa de texto é possível ver o ficheiro na íntegra, sem qualquer tipo de formatação.

3.4 Página de um recurso

Na página de um recurso está descrito o seu título, nome do ficheiro associado e descrição. Caso existam *reviews*, será apresentado também um *rating* global do recurso através da média dos valores das mesmas. A partir da página do recurso é também possível realizar o *download* ou visualização no *browser* do mesmo, consultar os comentários feitos por outros utilizadores, e, caso sejamos um utilizador autenticado, criar, editar ou apagar a nossa própria *review*.

3.4.1 Reviews

Para esta aplicação consideramos que as funcionalidades de comentário e *rating* poderiam ser implementadas de forma combinada, numa estrutura semelhante à que existe em plataformas como a *Google Play* e a *Amazon*.

Assim, na nossa aplicação, cada utilizador pode realizar uma única *review* por recurso (sendo que esta pode ser editada ou apagada mais tarde) que irá conter o seu *rating* de 0 a 10 do mesmo e, opcionalmente, o seu comentário. Na listagem de *reviews* na página do recurso serão visíveis apenas

aquelas que incluem comentários, de modo a reduzir a quantidade de conteúdo redundante, já que todos os *ratings* serão sempre contabilizados no cálculo do *rating* global do recurso.

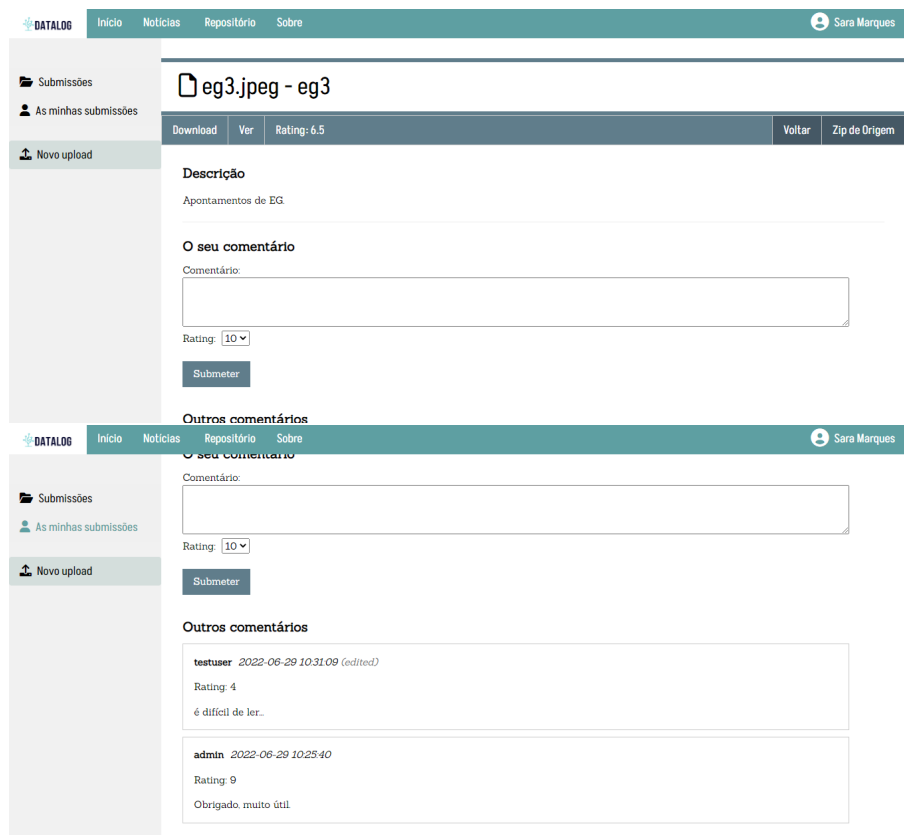


Figura 3.3: Página de um recurso e respetivas reviews

3.5 Notícias

As notícias são definidas por um título, um corpo/conteúdo, uma data de criação, uma data de última modificação e uma visibilidade. Para manter a consistência das notícias, estas são ordenadas por data de criação. A visibilidade permite esconder, quando não se pretende apagar, uma notícia. Por *default*, esta está a *true*.

As notícias estão disponíveis na página de notícias. Aqui podemos encontrar duas situações, uma página para ver notícias no caso de um utilizador, ou uma página para gerir notícias no caso de um administrador. Ambos podem utilizar a pesquisa por uma *string*, mas apenas o administrador têm acesso às funcionalidades de criação, alteração e eliminação de notícias. Enquanto que o administrador tem o poder de aceder a todas as notícias, o utilizador pode apenas ver as que têm visibilidade *true*.



Figura 3.4: Página de notícias para utilizadores

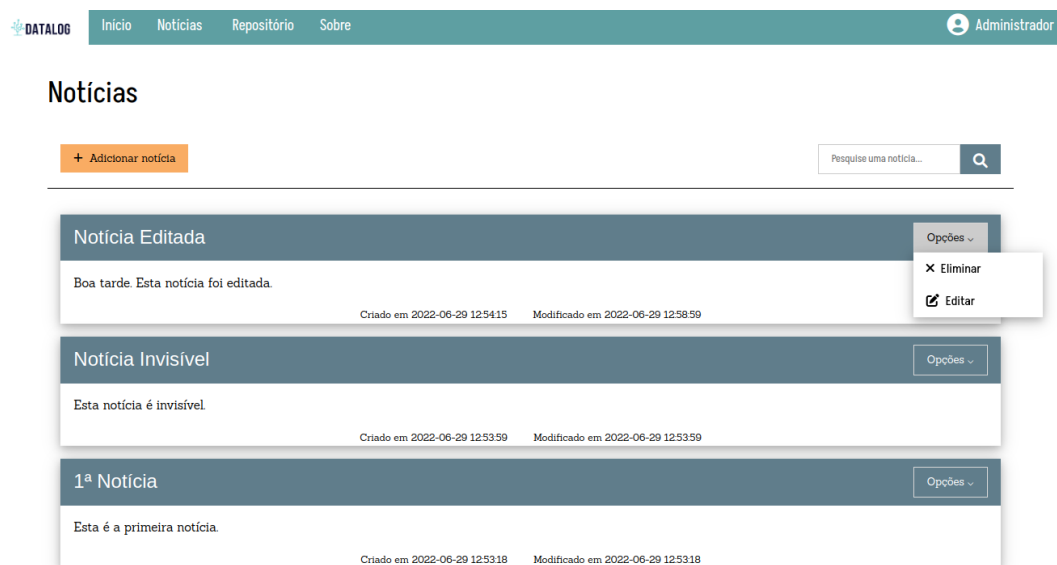


Figura 3.5: Página de notícias para administradores

Para criar ou alterar uma notícia, é usado um formulário com os campos título, conteúdo e visibilidade. A data de criação e a data de modificação são criadas pelo servidor de forma a automatizar o sistema.

A screenshot of a web application showing a modal form titled "Nova Notícia". The form has a title input field labeled "Título:" and a larger content input field labeled "Conteúdo:". Below the content field, there is a checkbox labeled "Visível" which is checked. At the bottom of the form is a teal button labeled "Submeter". The modal is set against a background of a sidebar and a main content area. At the bottom of the modal, it says "Criado em 2022-06-29 12:53:59" and "Modificado em 2022-06-29 12:53:59".

Figura 3.6: *Form* para notícias

Existe também um *slider* na página inicial com uma *preview* destas.

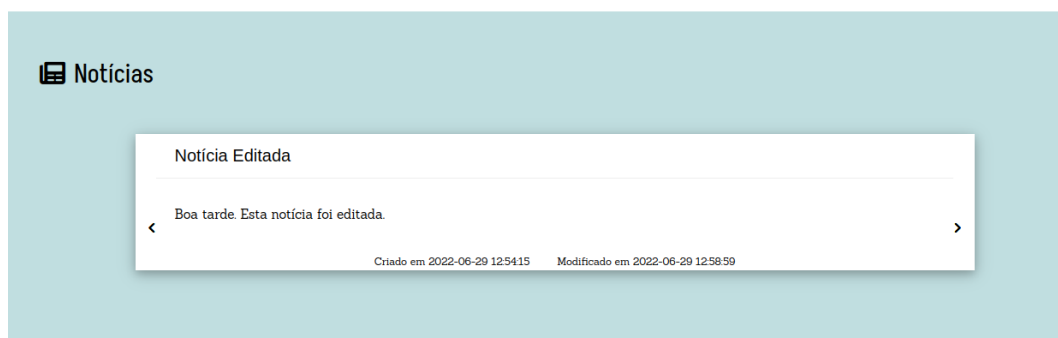


Figura 3.7: *Slider* de notícias

3.6 Gestão de recursos

Os utilizadores podem editar e eliminar os recursos dos quais são proprietários, ao passo que os administradores podem editar e eliminar qualquer recurso. De seguida são apresentadas algumas imagens que mostram estas funcionalidades.



Figura 3.8: Opções disponíveis para um utilizador registado (sem ser administrador).

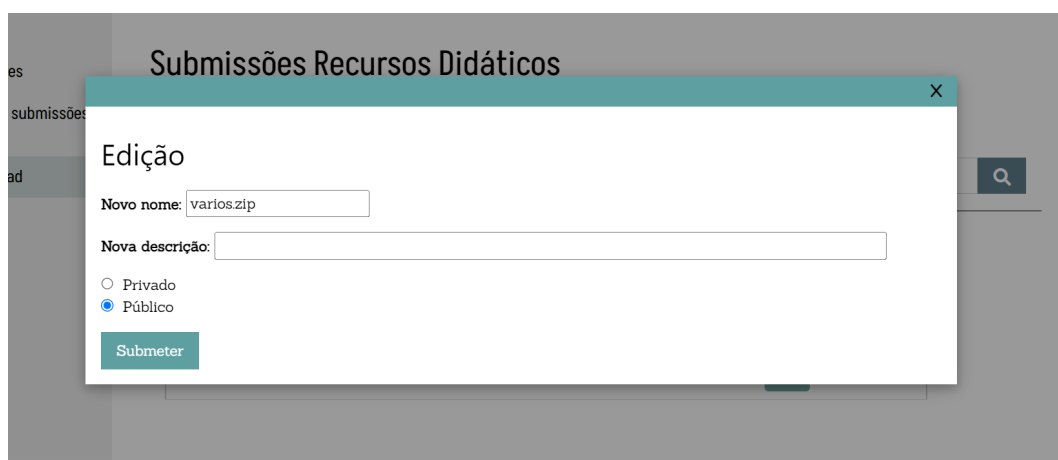


Figura 3.9: Modal com opções de edição de um ZIP.

Na imagem acima verifica-se que é possível mudar o nome do zip, sendo possível editar o seu tipo, ou seja, mudar de .zip para outro tipo qualquer. Na realidade, não será possível submeter a alteração caso isso seja feito, irá aparecer uma caixa de erro, tal como se encontra na figura abaixo.

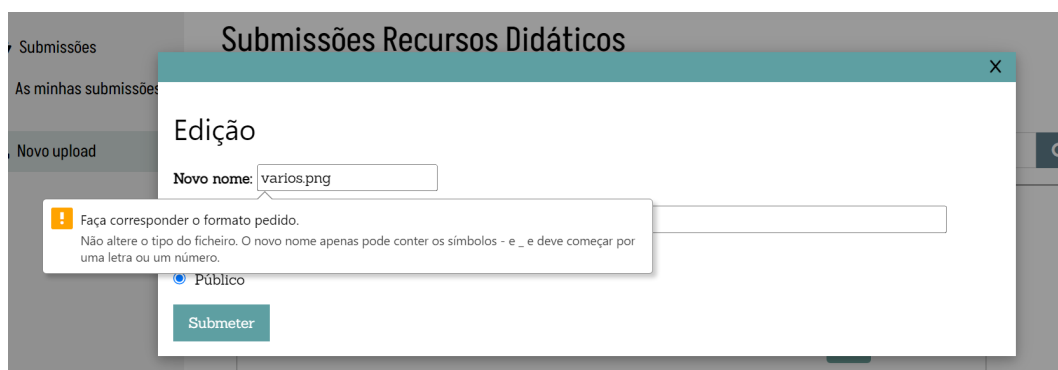


Figura 3.10: Erro ao tentar alterar o tipo do ficheiro.

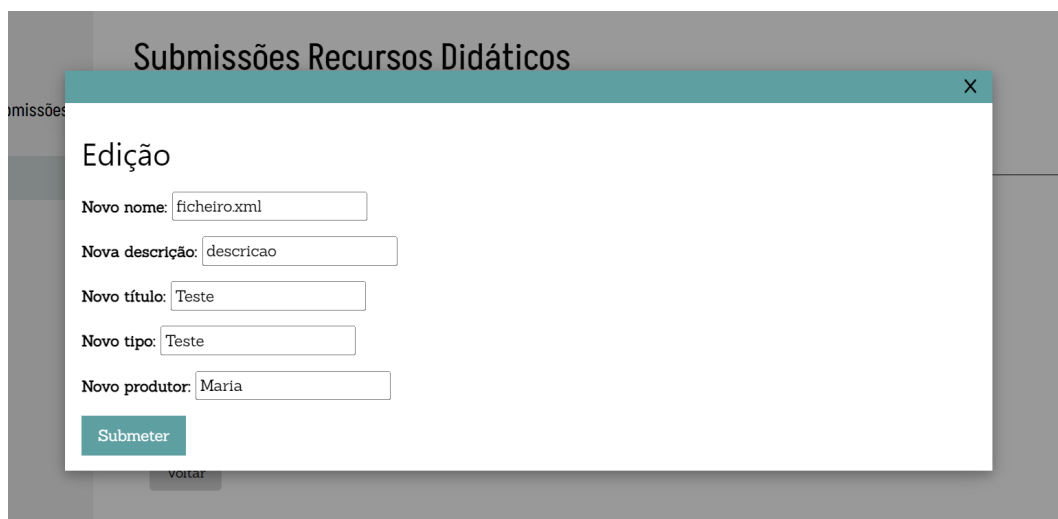


Figura 3.11: Modal de edição de um recurso.

Caso se tenta alterar o tipo do ficheiro, irá aparecer um erro semelhante ao da figura 3.10.



Figura 3.12: Eliminação de uma pasta.

A eliminação de pastas, ficheiros e dos próprios ZIPs é feita utilizando o *middleware* do *mongoose* (<https://mongoosejs.com/docs/middleware.html>). Antes de eliminar um ficheiro é necessário eliminar todas as *reviews* associadas ao mesmo, ao passo que para eliminar uma pasta é necessário eliminar todas as suas subpastas e ficheiros. Para realizar estas operações são definidas um conjunto de funções "pre" do *mongoose* para *queries* de eliminação de documentos na base de dados.

3.7 Gestão de Utilizadores

Uma das funcionalidades exclusivas a administradores refere-se à gestão de utilizadores. Será possível para qualquer administrador aceder a uma página onde poderá consultar uma lista de todos os utilizadores registados, registar novos utilizadores (como novos administradores ou como utilizadores normais), e editar ou apagar os utilizadores existentes.

No que se refere à edição de utilizadores, não consideramos pertinente o administrador conseguir editar dados identificativos do utilizador como o seu *username*, ou dados necessários ao seu acesso à plataforma como a palavra-passe ou o email (imaginando que este último poderia ser utilizado para a recuperação de palavra-passes numa aplicação real). O administrador poderá, no entanto, alterar o nome ou o nível dos utilizadores.

A eliminação de utilizadores, por sua vez, teve também de levar em conta outras considerações. Se ao eliminar um utilizador este for completamente removido da base de dados, será então possível que um novo utilizador se registre com o *username* do utilizador eliminado, ficando este com o "crédito" de todos os comentários e uploads feitos pelo prévio. De modo a evitar este conflito, a operação de eliminação apenas afeta o campo "deleted" do *schema* do utilizador, atribuindo ao mesmo o estado de eliminado e impossibilitando o seu futuro *login*. Poderão então ser mantidos todos os uploads e *reviews* feitos pelo utilizador eliminado com o seu *username* original.

É também de notar que é impossível para o administrador *logged-in* eliminar-se ou retirar o nível de administrador a si próprio, de modo a evitar conflito nas operações na página e garantir que existe sempre pelo menos um administrador.

The image displays two screenshots of the 'Gestão de Utilizadores' (User Management) page in the DATALOG application. The top screenshot shows the main interface with a table of existing users and a form to add a new user. The bottom screenshot shows the 'Adicionar Novo Utilizador' (Add New User) form with fields for Name, Username, Email, Password, and Role.

Top Screenshot: Gestão de Utilizadores

Search:

Utilizador	Nome	Email	Tipo	
admin	Administrador	admin@gmail.com	admin	<input type="button" value="✎"/>
sara	Sara Marques	sara@gmail.com	user	<input type="button" value="✎"/> <input type="button" value="🗑"/>
testuser	<input type="text" value="Teste 2"/>	test@gmail.com	<input type="button" value="admin"/>	<input type="button" value="Confirmar"/> <input type="button" value="Cancelar"/>

Adicionar Novo Utilizador

Nome
ex: Maria Ramos

Bottom Screenshot: Adicionar Novo Utilizador

Nome

Username

Email

Palavra-passe

Tipo:

Figura 3.13: Página de gestão de utilizadores

3.8 Estatísticas de utilização

O servidor da aplicação dispõe de um ficheiro *txt* onde são registados *logs* de todos os *requests* realizados pelo mesmo, através da utilização do middleware *morgan*. Partindo deste ficheiro, serão identificados todos os *requests* referentes a downloads e a visualizações, e a que recursos (ficheiros ou zips, ou apenas ficheiros no caso das visualizações) estes se referem. Essas informações serão depois listadas na página de estatísticas de utilização, acessível apenas a administradores.

Como os *requests* registados nos *logs* incluem apenas ids, e não nomes de recursos, é possível que sejam listados ids de recursos que tenham entretanto sido apagados da base de dados, caso esse em que esses recursos serão listados na página de estatísticas como "Ficheiro/ZIP Eliminado" em vez do seu nome.

Por outro lado, se o recurso ainda existir, a página irá incluir também um link para o mesmo.



Figura 3.14: Página de estatísticas de utilização

Capítulo 4

Conclusão e trabalho futuro

Vários dos desafios propostos foram cumpridos.

Consideramos que a forma como foi decidido apresentar os recursos ao utilizador, preservando a estrutura inicial de um SIP e permitindo a navegação do mesmo, visitando as suas pastas e ficheiros, trouxe alguma complexidade, já que foi necessário criar várias coleções e muitas vezes foi necessário fazer a junção de informação das várias. No entanto, esta escolha torna a aplicação interessante e fácil de utilizar.

Seria interessante, no futuro, a adição de notificações quando, por exemplo, o recurso de um utilizador é comentado. Seria interessante, também, melhorar a parte da visualização dos ficheiros, de modo a que periodicamente os ficheiros fossem eliminados da pasta com os recursos estáticos a serem servidos pelo servidor da aplicação.

Outra tarefa interessante para o futuro seria a criação de um *dockerfile* que cria um administrador de modo a que o *container* com a base de dados tenha desde logo um utilizador, que é um administrador, ao iniciar o *docker-compose* criado para o projeto.