

Trabalho Prático de Processamento e Representação de Informação

Diogo Paulo da Costa Pereira - A84092
Universidade do Minho

Gonçalo Rodrigues Pinto - A83732
Universidade do Minho

Luís Francisco Mendes Lopes - A85367
Universidade do Minho

Luís Mário Macedo Ribeiro - A85954
Universidade do Minho

(7 de Fevereiro de 2021)

Resumo

O presente relatório descreve o processo de análise, planeamento, de implementação de uma aplicação WEB que disponibiliza recursos educativos. O presente trabalho teve como objetivo a consolidação dos conhecimentos na área da representação de informação, em relação ao seu armazenamento e processamento. De forma ao seu desenvolvimento fez-se a persistência da informação seguindo o modelo MVC com recurso a *microservices*, serviços esses que comunicando entre si de forma a garantir a autenticação da aplicação. Assim foi possível a apresentação de notícias, a realização do processo de ingestão, administração e disseminação e ainda algumas funcionalidades, tais como a possibilidade de classificar, adicionar à lista de favoritos e efetuar comentários sobre recursos. É possível ainda executar pesquisas sobre os vários parâmetros dos recursos educativos e utilizadores, e consequentemente visualizar estatísticas sobre os mesmos. Posto isto, foi possível fornecer uma forma rápida e fácil de obter um determinado recurso educativo.

1 Introdução

No 1º semestre do 1º ano do Mestrado em Engenharia Informática da Universidade do Minho, existe uma unidade curricular enquadrada no perfil de Processamento de Linguagens e Conhecimento denominada por Processamento e Representação de Informação, que tem como objetivo a consolidação dos conhecimentos na área da representação de informação, em relação ao seu armazenamento e processamento.

O presente trabalho enquadra-se nesta unidade curricular e pretende desenvolver competências nos alunos na criação das estruturas necessárias à representação da informação e na criação das ferramentas necessárias para o seu consumo e distribuição.

Neste trabalho pretendeu-se que cada grupo realizasse um trabalho de análise, planeamento, de implementação de uma aplicação WEB que disponibiliza recursos educativos. Para esse fim, foi estabelecido como objetivo a disponibilização de recursos educativos de vários tipos (livros, artigos, publicações, etc.), para tal deve permitir adicionar novos tipos de recursos e novos recursos. Outro objetivo estabelecido foi ter recursos classificados por ano, tipo e tema. Outro objetivo é permitir que um utilizador faça um *Post* sobre um recurso como também, permitir que outros utilizadores comentem *Posts*. Por fim, deve ser criado um sistema de *ranking* para os recursos.

2 Contextualização

Para realizar a análise, planeamento e implementação foi necessário efetuar um levantamento de características e estudo das mesmas, com o objetivo de perceber a melhor estrutura aplicacional para a aplicação a desenvolver. Após concluído esse processo decidiu-se utilizar a *framework Express.js*[1] do *Node.js*[2] pois o seu objetivo é otimizar a construção de aplicações web e *API's*.

2.1 Persistência da Informação

De forma a garantir a persistência dos dados foi criado um documento com as respetivas coleções numa base de dados orientada a documentos, *MongoDB*[3], para esse efeito também foi necessário, realizar a conexão com a base de dados criada previamente para tal utilizou-se a *framework mongoose*[4].

2.2 MVC (Model, View, Controller)

Pensando no futuro da aplicação a desenvolver, onde a quantidade de recursos educativos ou utilizadores pode crescer exponencialmente. Esta aplicação foi desenvolvida tendo em conta os conceitos de modularidade e encapsulamento.

A aplicação segue assim, o modelo *MVC*, *Model-View-Controller*:

- O *Model* consiste nos dados da aplicação, lógica e funções;
- A *View* encarrega-se da representação dos dados;
- O *Controller* descodifica o que recebeu da *View* e converte em comandos para o *Model*;

2.3 *Microservices*

A aplicação foi construída utilizando o conceito denominado *microservices*[5] que segue uma abordagem arquitetónica e organizacional, esta é composta por pequenos serviços independentes que se comunicam usando *API's* (*Application Programming Interface*[6]) bem definidas. Esta abordagem facilitou a escalabilidade e agilizou o desenvolvimento da aplicação, habilitando a inovação e acelerando o tempo de execução do projeto.

Com uma arquitetura de *microservices*, a aplicação foi criada como componentes independentes que executam cada processo da aplicação como um serviço. Esses serviços comunicam por meio de uma interface bem definida usando *API's* leves. Os serviços criados realizam uma única função e são executados de forma independente, cada serviço pode ser atualizado, implementado e escalado para atender à procura de funções específicas da aplicação.

Tal como foi referido os serviços criados comunicam através da interface, além disso foram também criados outros dois serviços, que são:

- Serviço de Autenticação - *API* construída de forma a autenticar os utilizadores, bem como, os registar na aplicação;
- Serviço de Dados - *API* construída que efetua o registo de novas informações na base de dados bem como fornece essas mesmas;

Existindo apenas uma interface para controlar o processo de interação, os outros serviços criados não possuem esta componente de comunicação com o utilizador, limitando-se a interagir com a interface construída, assim estes serviços recebem os pedidos nas rotas previamente definidas e respondem a esses no formato *JSON*[7].

3 Conceitos básicos

Como foi apresentado anteriormente desenvolveu-se uma aplicação web, assente numa divisão de serviços de forma a dar uma experiência intuitiva ao utilizador. Para isto, foi necessário identificar alguns conceitos básicos importantes.

3.1 Utilizador

O sistema encontra-se protegido com autenticação, isto é, existe a necessidade de efetuar um registo prévio onde se define a password e email para posteriormente se aceder.

Na aplicação desenvolvida existem 3 níveis de acesso:

- Administrador que tem acesso a todas as operações;
- Produtor (autor de recurso) pode consultar tudo e executar todas as operações sobre os recursos de que é produtor/autor;
- Consumidor pode consultar e descarregar os recursos públicos.

Um utilizador possui as seguintes informações: nome, email, filiação (estudante ou docente, curso, departamento), nível, data de registo na plataforma, data de último acesso, password e ainda o seu nome de utilizador na plataforma *Github*.

3.2 Recurso

Um recurso pode ser visto como um conjunto de ficheiros.

Um recurso possui as seguintes informações: um tipo (relatório, tese, artigo, aplicação, slides, teste/exame, problema resolvido e ainda tem a hipótese de introduzir um diferente tipo do conjunto base estabelecido), título, subtítulo (opcional), data de criação, data de registo (entrada no sistema), visibilidade (público: todos podem ver e descarregar ou privado: apenas disponível para administradores e produtor), uma descrição, bem como, o seu produtor/autor.

3.3 Notícias

Após um determinado recurso público ser publicado é gerado uma notícia ou o administrador pode introduzir uma ele mesmo. Uma notícia é composta pelo email do autor da notícia, o seu nome, a data e uma descrição.

4 A Proposta

Com base nos conceitos básicos apresentados o grupo decidiu apresentar as seguintes propostas aos problemas propostos:

4.1 Comunicação entre serviços

O fluxo de pedidos e respostas realizado entre as *API's* e a Interface Web é o seguinte:

1. Quando é realizado um pedido à interface, é iniciado um fluxo de informação entre vários componentes;
2. A interface inicia a página correspondente ao pedido mediante o que se encontra definido nas *routes* da interface;
3. No entanto, a interface envia os pedidos necessários à *API*, através das rotas definidas;
4. A *API* recebe estes pedidos e invoca o método na rota correspondente;
5. Os dados resultantes desta execução são enviados para a interface;
6. Finalmente, a interface trata os dados recebidos (em formato *JSON*), para posterior apresentação.

4.2 Autenticação

Na aplicação desenvolvida um dos objetivos requeridos foi a existência de níveis de acesso, assim como, um registo prévio. Para atingir esse objetivo o grupo decidiu destacar estas funcionalidades criando um novo serviço que seja responsável apenas exclusivamente do processo de autenticação.

Recorremos à utilização de *JSON Web Tokens* (JWT) [8] uma vez que é um padrão da Internet para a criação de dados com assinatura opcional e/ou criptografia cujo *payload* contém o *JSON* com as informações. Os *tokens* podem ser assinados usando um segredo privado ou uma chave pública/privada.

A ideia por trás desta utilização passa pelo servidor de autenticação gerar um *token* de forma autenticar os utilizadores e fornecendo ao utilizador, assim este pode usar esse *token* que foi atribuído para provar que está ligado ao sistema e provar a sua identidade. Uma particularidade interessante ao utilizar-se *tokens* é o facto de que estes podem ser assinados por uma chave privada de uma parte da comunicação (neste caso o servidor de autenticação assina esse *token*), assim esta pode verificar posteriormente se o *token* que recebe é legítimo. Se a outra parte, por alguns meios adequados e confiáveis, estiver na posse da chave pública correspondente, ela também poderá verificar a legitimidade do *token*.

Os *tokens* foram projetados para serem compactos, seguros para URL e utilizáveis, especialmente num contexto de login único no navegador da web. As declarações JWT foram usadas para transmitir a identidade dos utilizadores autenticados entre os diversos serviços implementados.

4.3 Processo de ingestão, administração e disseminação

Em relação ao processo de ingestão de recursos, estes são introduzidos na aplicação como ficheiros ZIP, formato do *Submission Information Package* (SIP) definido, que seguem uma determinada estrutura simples, baseada no BagIt[9] que é um conjunto de convenções de sistema de documentos hierárquicos projetado para suportar o seu armazenamento.

Em relação ao processo de administração é efetuado um conjunto de operações, CRUD e outras, sobre os recursos armazenados. Este processo é responsável por descompactar o ficheiro, sendo este o formato do *Archival Information Package* (AIP) considerado, é também responsável por verificar através do manifesto (que acompanha cada recurso) os ficheiros recebidos, isto é, por uma questão de simplificação decidiu-se que o formato do manifesto é em *JSON* com uma determinada estrutura estabelecida, pois facilita a verificação deste processo, bem como, permite uma grande flexibilidade porque possibilita a representação de hierarquias de pastas ou ficheiros que um determinado recurso pode possuir, de seguida apresentamos a estrutura básica estabelecida do manifesto que o nosso sistema aceita.

```
{
  "ficheiros": [
    {
      "titulo": "...",
      "tipo": "..."
    }
  ],
  "pastas": [
    {
      "titulo": "...",
      "conteudo": {
        "ficheiros": [
          {
            "titulo": "...",
            "tipo": "..."
          }
        ],
        "pastas": []
      }
    }
  ]
}
```

Nesta validação é verificado se existe um ficheiro denominado *manifesto.json* como também uma pasta denominada *data* que possui os ficheiros. A validação efetuada consiste em verificar se o nome juntamente com o tipo presente no manifesto existe no ficheiro ZIP introduzido no upload. Se tudo estiver bem o recurso é armazenado na plataforma ficando disponível como um conjunto de ficheiros descomprimido para os utilizadores, caso contrário o sistema avisa que o *upload* não ocorreu.

Em relação ao processo de disseminação que acontece quando o utilizador pretende descarregar um recurso é feita a conversão do AIP num *Dissemination Information Package* (DIP), por uma questão de organização considerou-se que o formato do DIP é igual ao SIP, ou seja, o download corresponde ao processo inverso, isto é, o conjunto de ficheiros é transformado num ficheiro único ZIP que possui o manifesto introduzido.

Por uma questão de simplificação optou-se por criar uma pasta no servidor de dados que guarda todos os ficheiros, onde esta é composta por uma pasta para cada produtor, que por sua vez é constituída pelos uploads deste e downloads, ou seja, na pasta de uploads podemos encontrar o conjunto de ficheiros descomprimidos e na pasta de download podemos encontrar o conjunto de ficheiros comprimidos pronto a descarregar.

4.4 Apresentação das notícias

Por uma questão de clareza o nosso grupo definiu que as notícias iriam ser geradas após o recurso ter sido validado e armazenado. Além disto foi considerado a criação de uma rota para introdução de notícias possibilitando o administrador de avisar os utilizadores. Uma notícia é composta então pelo identificador do recurso que foi submetido, a data em que é registada, o nome do autor da notícia, o seu tipo (podendo ser "Warning" ou "File") e ainda uma breve descrição (caso seja referente a um *upload* indica que houve uma nova submissão por parte do autor de um determinado tipo de recurso e o nome do recurso que foi indicado aquando da introdução no sistema).

4.5 Funcionalidades

4.5.1 Classificação

Para podermos classificar um determinado recurso o nosso grupo decidiu que, cada recurso armazenado no documento respetivo criado na base de dados possui uma chave, denominada "estrelas", que vai representar a classificação de um determinado recurso. Esta encontra-se associada a um valor composto pelo número médio de estrelas dados e uma lista dos autores que já avaliaram, assim é possível construir a média, assim, é permitido que cada utilizador avalie apenas uma vez ou reformule a avaliação dada.

4.5.2 Favoritos

No sentido de ser possível a adição de um recurso à lista de favoritos do utilizador, o grupo propôs que cada recurso armazenado no documento respetivo criado na base de dados possui uma chave, denominada "favoritos", cujo valor associado é uma lista de emails que identificam cada utilizador, deste modo é possível identificar quais os recursos favoritos de um determinado utilizador.

4.5.3 Comentários

De forma a podermos efetuar comentários sobre um determinado recurso o nosso grupo decidiu que cada recurso armazenado no documento respetivo criado na base de dados possui uma chave, denominada "comentarios", que vai representar a lista de comentários, onde cada elemento que faz parte desta lista é um valor que representa o comentário, para esse efeito optou-se por cada comentário possuir um identificador único que é gerado pela base de dados, uma data, o autor do comentário e a descrição que este escreveu neste comentário.

4.5.4 Pesquisa

Uma funcionalidade proposta para se implementar foi uma pesquisa sobre o nome do utilizador, sobre o título ou tipo de recurso, ou ainda uma pesquisa sobre a data de criação. Esta funcionalidade é particularmente relevante para o grupo, para tal pensámos em destacar esta, no sentido de ser sempre possível efetuar uma pesquisa, assim criámos rotas exclusivas para cada tipo de pesquisa no Serviço de Dados onde é passado o que cada utilizador pretende pesquisar.

4.5.5 Estatísticas

Por fim uma outra funcionalidade proposta prende-se com uma apresentação de estatísticas no sentido de perceber quais os 3 recursos mais bem classificados, quais os 3 recursos preferidos e ainda quais são os 3 autores que mais contribuíram para plataforma (com maior número de *uploads*), tendo em conta as propostas para solucionar outras funcionalidades, a apresentação destas estatísticas torna-se trivial.

5 A Implementação

Com base nas diversas propostas que o nosso grupo apresentou para os mais diversos problemas, de seguida iremos apresentar como foi implementado cada uma delas.

5.1 Comunicação entre serviços

Para que os serviços funcionem corretamente tal como a sua comunicação entre eles, decidiu-se dividir as rotas necessárias pelos conceitos básicos, ou seja, existem rotas apenas para aceder aos recursos, utilizadores, notícias e adicionalmente para pesquisa isto no Serviço de Dados; por outro lado, no Serviço de Autenticação apenas foram implementadas rotas que permitem efetuar o login e o registo de utilizadores, além de autenticar a interface construída.

Por outro lado, a interface possui rotas específicas para manipular a informação dos utilizadores e dos recursos, além disto a rota por predefinição encaminha para a página principal com as notícias e estatísticas, bem como, para a pesquisa dado serem funcionalidades principais e básicas.

Em cada uma destas rotas é efetuado um pedido ao serviço que permite obter a informação respetiva. Para isso, utilizámos a ferramenta *axios* para efetuar os pedidos aos serviços implementados, se o pedido ocorrer com sucesso, isto é, o serviço em causa responde a informação através de *JSON* então a interface apresenta uma página com a informação devida, para tal no sentido de tornar cada página intuitiva e agradável, decidimos utilizar a plataforma *Nicepage*[10] que permite a liberdade criativa no *design* da aplicação, através de um editor avançado de arrastar e soltar que oferece o máximo de expressão criativa; contudo caso o serviço retorne alguma mensagem de erro é apresentado uma página de erros explicativa que de forma muito simples indica qual o erro ocorrido.

Uma outra implementação utilizada a fim de tornar a experiência de utilização única e apelativa foi o recurso a *Modal Boxes*. Basicamente, criou-se um ficheiro de *scripts* que é carregado cada vez que uma página é apresentada, este ficheiro possui funções que apresentam pequenas caixas de texto para inserir a informação ou simplesmente apresentar a mesma, para tal recorreu-se à ferramenta *ajax*[11] para comunicar com os serviços implementados. De forma a pensar no futuro da aplicação, bem como, na organização da informação recorreu-se a *datatables*[12] que é um *plug-in* para tabelas da *jQuery*[13], esta utilização permitiu que as tabelas sejam organizadas mediante a preferência do utilizador, tal como, facilita uma pesquisa sobre algum campo da tabela e ainda permite mostrar as linhas que o utilizador pretender consequentemente é gerado uma página de linhas da tabela.

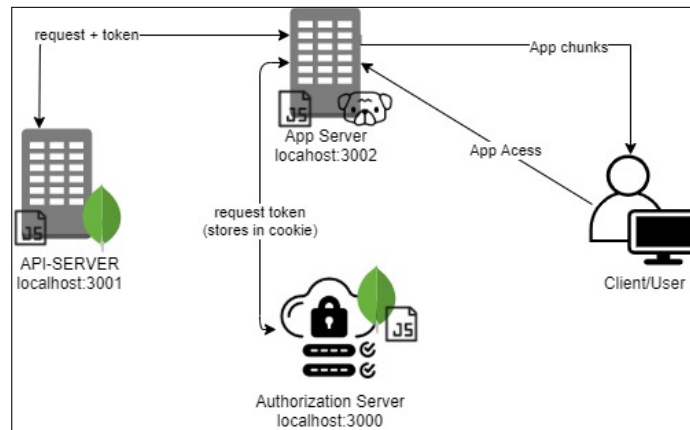


Figura 1: Serviços implementados e a sua comunicação.

5.2 Autenticação

No sentido de garantir a autenticação dos utilizadores bem como dos serviços implementados, a primeira comunicação necessária entre os serviços é entre a interface e o servidor de autenticação, no sentido de esta primeira autenticar a sua autenticidade junto deste segundo serviço. Para isso é feito um *POST* com uma chave gerada ao servidor de autenticação, este recebe assim o pedido verifica a chave recebida e caso seja uma chave válida atribui um *token* com data de expiração de um dia assinado com a chave privada que foi gerada previamente, com recurso do software *OpenSSL*[14], posteriormente envia através de *JSON* o *token* gerado com *framework jsonwebtoken*[15].

Após atribuído um *token* à interface, esta reencaminha o utilizador para a página de login. Caso seja um novo utilizador então este preenche o formulário que a interface fornece com os seus dados e posteriormente é feito um pedido ao servidor de autenticação, no sentido de registar este novo utilizador na base de dados utilizada. Caso já se encontre registado apenas tem que preencher o formulário de *login* com as credenciais, após isso é feito um pedido ao servidor de autenticação de forma a verificar a existência do utilizador na base de dados bem como se a password introduzida é correta, caso este processo ocorra com sucesso é gerado então com recurso ao *jsonwebtoken* um *token* com o email, nível de acesso e a data ao qual este expira (estabeleceu-se 1 dia) para codificar este *token* foi utilizado um segredo previamente definido. O *token* é guardado nos *cookies* do *browser*.

Após atribuído o *token* o utilizador deve utilizar este, para efetuar pedidos à interface e esta consequentemente aos outros serviços, ou seja, em cada serviço implementado no seu *middleware* é verificado se o pedido que chegou possui um *token* e posteriormente é feita uma verificação com recurso ao segredo utilizado aquando da geração, de forma a obter o email bem como nível de acesso, assim cada serviço tem conhecimento do utilizador que fez o pedido.

5.3 Processo de ingestão, administração e disseminação

Para possibilitar o *upload* e o *download* de recursos educativos são aplicados vários processos que são responsáveis por converter os pacotes utilizados.

Assim, o processo de ingestão corresponde ao *upload* de recursos na plataforma, tal como foi referido é necessário a existência de um *manifesto.json* que indica quais os ficheiros e respetivos tipos existem na pasta denominada *data*.

No sentido de demonstrar o processo de administração iremos recorrer à especificação textual utilizada para explicar um *use case* :

Use Case: Validação do *upload* de um recurso.

Descrição: Processo de validação de um recurso que foi introduzido pelo utilizador.

Cenários: O utilizador insere um recurso na plataforma.

Pré-condição: O utilizador introduziu um recurso.

Pós-condição: O utilizador consegue efetuar o *upload* de um novo recurso.

Fluxo Normal:

1. O sistema verifica que o tipo do ficheiro introduzido é do tipo *application/x-zip-compressed*;
2. O sistema descomprime o ficheiro recebido com auxílio da *framework adm-zip*[16] colocando o conjunto de ficheiros na pasta "quarentena" de upload com a extensão ".zip" e elimina o ficheiro comprimido;
3. É verificado a existência do *manifesto.json*;
4. De seguida é verificado a existência da pasta denominada *data*;
5. Para a lista de ficheiros presente é verificado a existência do ficheiro especificado juntamente com o seu tipo na atual pasta;
6. Caso não seja especificado nenhuma pasta na lista de pastas no manifesto a validação através do mesmo termina;
7. O conjunto de ficheiros é movido da pasta "quarentena" de *uploads* para a pasta de *uploads* do utilizador em causa;
8. É registado na base de dados um novo recurso com as propriedades que o utilizador inseriu no formulário de *upload* bem como algumas informações retiradas do ficheiro submetido (nome, tipo, tamanho);
9. Caso tenha sido indicado que é um recurso público é criado uma notícia com o nome do utilizador, email, data, o identificador do ficheiro e uma descrição completa;
10. A *API* retorna para um pedido à interface para apresentar a página pessoal do utilizador;

Fluxo Alternativo 1: [Lista de pastas não é vazio] (Passo 6)

- 6.1. O sistema verifica se a pasta com o título indicado existe na atual pasta;
- 6.2. O sistema verifica o conteúdo da atual pasta;
- 6.3. Retorna a 5;

Fluxo Alternativo 2: [Recurso é privado] (Passo 9)

- 9.1. O sistema não cria nenhuma notícia;
- 9.2. Retorna a 10;

Fluxo de Excepção 1: [Tipo inválido] (Passo 1)

- 1.1. O sistema elimina da pasta "quarentena" de *uploads* o ficheiro introduzido, percorrendo todas subpastas incluídas;
- 1.2. O sistema informa a interface que o *upload* não ocorreu;

Fluxo de Excepção 2: [Manifesto inexistente] (Passo 3)

- 3.1. O sistema elimina da pasta "quarentena" de *uploads* o conjunto de ficheiros, percorrendo todas subpastas incluídas;
- 3.2. O sistema informa a interface que o *upload* não ocorreu;

Fluxo de Excepção 3: [Pasta *data* inexistente] (Passo 4)

- 4.1. O sistema elimina da pasta "quarentena" de *uploads* o conjunto de ficheiros, percorrendo todas subpastas incluídas;
- 4.2. O sistema informa a interface que o *upload* não ocorreu;

Fluxo de Excepção 4: [Ficheiro especificado inexistente] (Passo 5)

- 5.1. O sistema elimina da pasta "quarentena" de *uploads* o conjunto de ficheiros, percorrendo todas subpastas incluídas;
- 5.2. O sistema informa a interface que o *upload* não ocorreu;

Fluxo de Excepção 5: [Pasta especificada inexistente] (Passo 6.1)

- 6.1.1 O sistema elimina da pasta "quarentena" de *uploads* o conjunto de ficheiros, percorrendo todas subpastas incluídas;
- 6.1.2 O sistema informa a interface que o *upload* não ocorreu;

Em relação ao processo de disseminação como foi apresentado previamente é a operação de transformar o formato do recurso armazenado no formato ao qual foi inserido neste caso ZIP, para tal é acedido ao recurso que se encontra na pasta de *uploads* do utilizador em causa, posteriormente com recurso à *framework adm-zip* criou-se um ficheiro ZIP com os ficheiros e pastas presentes no recurso, em causa de seguida este ficheiro criado é colocado na pasta de *downloads* do utilizador, por fim o recurso encontra-se no formato devido para posteriores acessos.

5.4 Apresentação das notícias

No sentido da apresentação das notícias o grupo decidiu apresentar as 5 notícias mais recentes na página inicial, para tal a interface efetua o pedido ao Serviço de Dados na rota devida, esta acede ao documento que possui as notícias ordena-as pela data colocando as notícias mais recentes no topo da resposta e posteriormente efetua uma limitação desta. Finalmente encaminha o resultado obtido para a interface onde é apresentado.

5.5 Funcionalidades

5.5.1 Classificação

Para ser possível a classificação implementámos uma função no ficheiro *scripts* que é carregado onde nesta função é passado o número de estrelas a atribuir ao recurso em causa juntamente com o identificador deste e a classificação atual deste associada ao ficheiro (Será 0 caso não tiver nenhuma classificação atribuída associada ao utilizador). Esta função utiliza então *ajax* para efetuar a substituição da classificação associada ao recurso pois como foi dito a classificação associada é a média dada pelos utilizadores da aplicação, assim o Serviço de Dados acede ao documento que representa o recurso obtém a quantidade de autores e a classificação atual, posteriormente efetua o cálculo da nova classificação e substitui a informação.

5.5.2 Favoritos

Para adicionar ou remover um determinado recurso aos favoritos, inicialmente criou-se duas funções no ficheiro *scripts* que é carregado. No entanto, de modo a simplificar esta funcionalidade, usando *jquery* para a apresentação na parte do cliente, sem a necessidade de carregar a página toda de novo, criou-se apenas uma função, substituindo as duas acima. Para tal é indicado o identificador do ficheiro ao qual o utilizador pretende adicionar como favorito, em ambas as funções é utilizado *ajax* para atualizar a lista presente no documento que representa um recurso cuja lista possui os identificadores dos utilizadores que tem como favorito aquele recurso, assim dada a estrutura definida o processo de adição e remoção torna-se bastante simples pois apenas temos que adicionar ou remover o identificador do utilizador da lista do recurso em causa.

5.5.3 Comentários

Muito idêntico à funcionalidade previamente apresentada, para adicionar um comentário criámos uma *Modal Box* que permite escrever um determinado comentário. Após submetido o comentário é então efetuado um pedido ao servidor de dados para adicionar o que foi escrito à lista de comentários de um determinado recurso, para tal é necessário o identificador do recurso em causa, bem como o comentário. Assim para adicionar um comentário basta adicionar à lista.

Para a remoção foi necessário aceder aos comentários do recurso em causa, verificar quem é o autor e caso seja um administrador, o autor do comentário ou inclusive a pessoa autora do recurso é então removido o comentário em causa.

5.5.4 Pesquisa

Tal como foi referido previamente criou-se rotas exclusivas para pesquisas mediante o tipo pretendido, assim a barra da pesquisa encontra-se sempre disponível para o utilizador nesta tem uma lista de opções de tipos à qual pode escolher (nome do utilizador, nome do recurso, tipo de recurso ou data de criação do recurso), após escrever o que pretende é efetuado um pedido ao Servidor de Dados no tipo correspondente juntamente com a expressão que o utilizador inseriu. Posteriormente esta *API* recorrendo à função que é disponibilizada pela base de dados utilizada que fornece recursos de expressões regulares para *strings* de correspondência de padrões em consultas, *\$regex*, aplica esta consulta ao tipo que se foca a pesquisa obtendo assim os recursos ou utilizadores que correspondem ao que foi escrito, caso não seja uma data aplicou-se por opção ignorar a insensibilidade de maiúsculas e minúsculas, sendo possível combinar maiúsculas e minúsculas.

5.5.5 Estatísticas

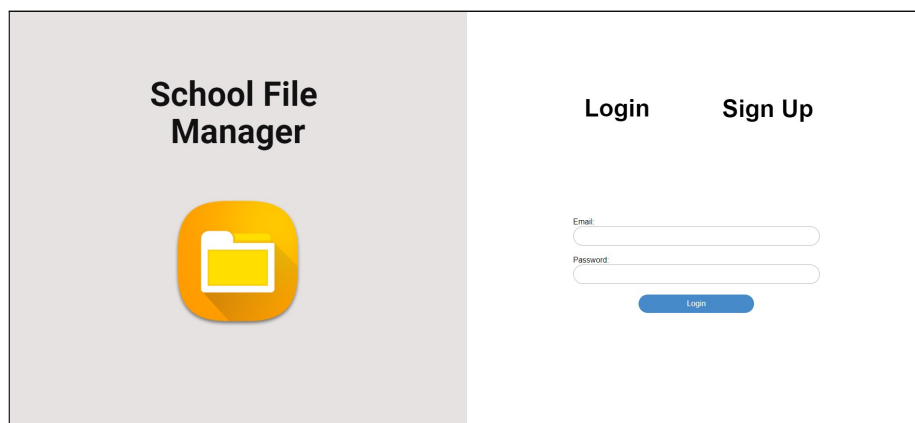
Após o processo de implementação das mais diversas funcionalidades tal como foi dito na proposta de resolução dado a estrutura e rotas criadas, a apresentação de estatísticas torna-se bastante simples na medida em que o utilizador quando solicita a página inicial junto do pedido ao servidor de dados são feitos pedidos com *axios* de forma a obter os 3 recursos mais bem classificados, os 3 recursos preferidos e os 3 autores que mais contribuíram para plataforma (com maior número de *uploads*), o servidor de dados nas rotas devidas aplica consultas considerando apenas os ficheiros públicos onde ordena pelo número de estrelas, para obter os recursos preferidos dos utilizadores é calculado o tamanho da lista de favoritos, para obter os recursos com maior número de *uploads* é criado um documento por cada autor onde é criado uma lista de recursos ao qual deu upload de seguida é calculado o tamanho da lista criada, em cada uma destas consultas os recursos são ordenados de forma descendente limitando apenas a apresentação a 3 recursos. Posteriormente *API* retorna os resultados obtidos à interface em *JSON*.

6 Resultados obtidos

A divisão em *microservices* permitiu a distinção clara entre os serviços *backend* e *frontend* da aplicação.

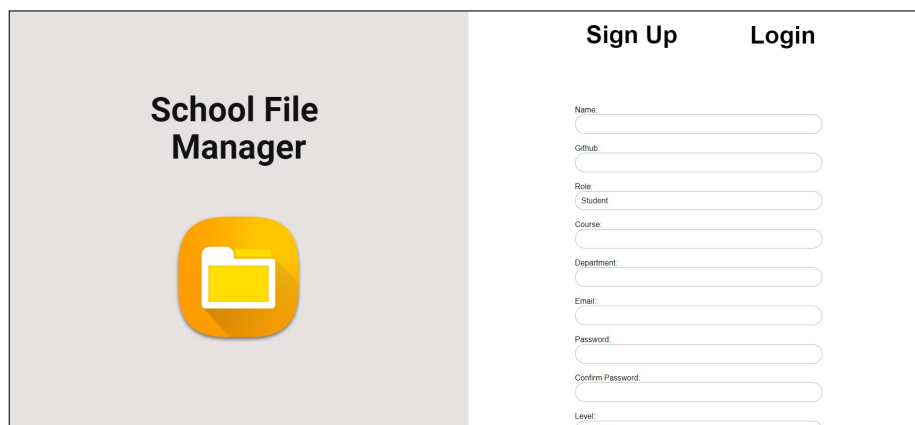
Um dos objectivos delimitados consistiu no desenvolvimento de uma interface que permitisse gerir recursos educativos.

Assim sendo, foi implementado um menu dinâmico que permite a seleção da informação que se pretende visualizar. A seleção de um elemento deste menu, reencaminha para um conjunto de dados e indicadores relativos a esse parâmetro. De forma a facilitar a monitorização recorreu-se a tabelas, ícones ilustrativos das funcionalidades, e outras representações visuais.



The screenshot shows a web interface for 'School File Manager'. On the left, there is a grey sidebar with the title 'School File Manager' and a yellow folder icon. On the right, the main content area has a white background. At the top right, there are two links: 'Login' and 'Sign Up'. Below these links, there are two input fields labeled 'Email' and 'Password'. A blue 'Login' button is positioned below the password field.

Figura 2: Menu para efetuar o *login*



The screenshot shows the same 'School File Manager' interface, but with the 'Sign Up' link selected. The main content area now displays a sign-up form with the following fields: 'Name', 'Github', 'Role' (with a dropdown menu showing 'Student'), 'Course', 'Department', 'Email', 'Password', 'Confirm Password', and 'Level' (with a dropdown menu showing 'Consultant').

Figura 3: Menu para efetuar o registo.

School File Manager

Search

File

Q

Top 3 Most Classified

Date	File Title	Average Classification	Author	
2021-01-18	Resolução do Teste	5	Luis Ribeiro	
2019-06-13	Fichas de Algoritmos e Complexidade	5	Diogo Pereira	
2020-12-01	Projeto Final	4	JCR	

Top 3 Favourites

Date	File Title	Number of Favourites	Author	
2021-01-18	Resolução do Teste	3	Luis Ribeiro	
2020-12-01	Projeto Final	2	JCR	
2019-06-13	Fichas de Algoritmos e Complexidade	1	Diogo Pereira	

Top 3 Authors

Author	Number of Uploads
JCR	5
Gonçalo Pinto	2
Luís Lopes	1

News

New submission: Producer JCR has just released an Slides entitled "Aula 1".

Check this!

2021-02-07T13:58

New submission: Producer Gonçalo Pinto has just released an Application entitled "Projeto de GCS".

Check this!

2021-02-07T13:56

New submission: Producer Luís Lopes has just released an Test/Exam entitled "Mini-Testes de PT".

Check this!

2021-02-07T13:54

New submission: Producer Luis Ribeiro has just released an Problem Solved entitled "Resolução do Teste".

Check this!

2021-02-07T13:51

New submission: Producer Diogo Pereira has just released an Files entitled "Fichas de Algoritmos e Complexidade".

Check this!

Figura 4: Página inicial.

School File Manager

Search

File

Q

Search:

Date	File Title	Resource Type	Average Classification	Your Classification	Author			
2021-01-04	Projeto de GCS	Application	No classification	★★★★★	Gonçalo Pinto			
2020-12-01	Projeto Final	Enunciated	4	★★★★★	JCR			
2019-06-13	Fichas de Algoritmos e Complexidade	Files	5	★★★★★	Diogo Pereira			
2021-01-25	Resultados da Avaliação de PT	Grades	No classification	★★★★★	JCR			
2021-01-18	Resolução do Teste	Problem Solved	5	★★★★★	Luis Ribeiro			
2020-06-13	Material de PL	Slides	No classification	★★★★★	JJ			

Previous

1

2

Next

Figura 5: Página onde é possível visualizar os recursos.

Date	File Title	Resource Type	Average Classification	Your Classification	Author
2019-06-13	Fichas de Algoritmos e Complexidade	Files	5	★★★★★	Diogo Pereira
2020-12-01	Projeto Final	Enunciated	4	★★★★★	JCR
2021-01-18	Resolução do Teste	Problem Solved	5	★★★★★	Luís Ribeiro

Figura 6: Página onde é possível visualizar os recursos preferidos.

Name	Git
Administrator	
Administrador2	
Ana Rita Rosendo	Rita-Rosendo
António Gomes	A67845
António Lindo	António070
Bernardo Carvalho	Pacifico53

Figura 7: Página onde é possível visualizar os utilizadores registados.

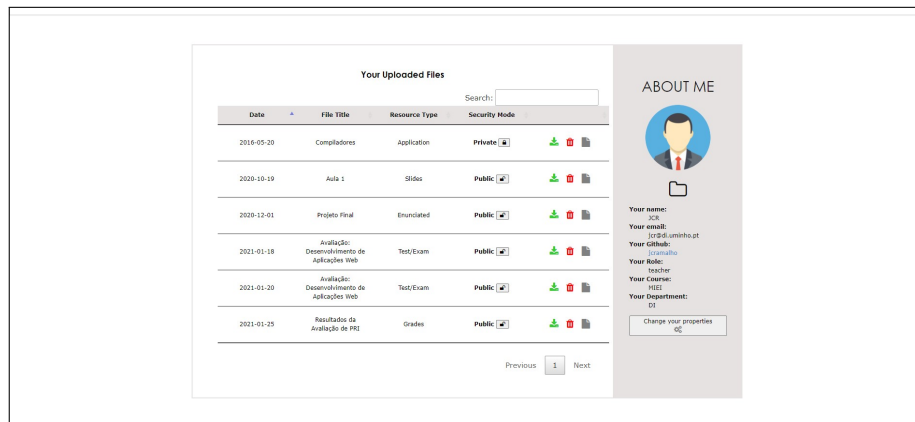


Figura 8: Página pessoal de um utilizador.

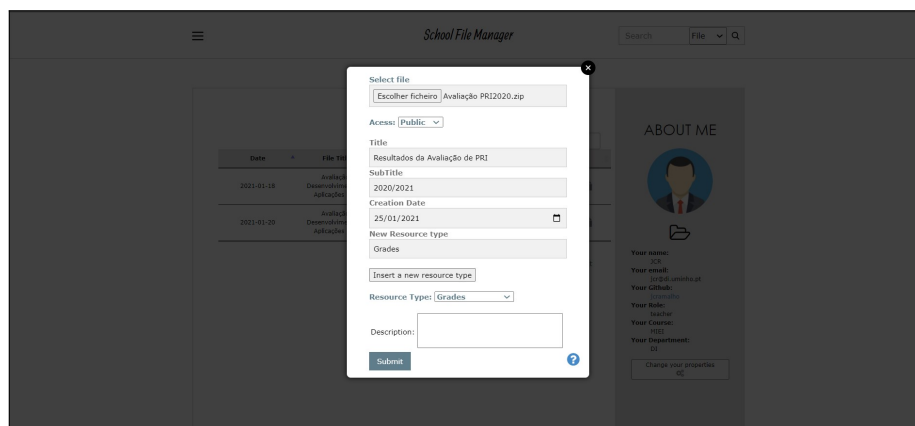


Figura 9: Página de um utilizador.

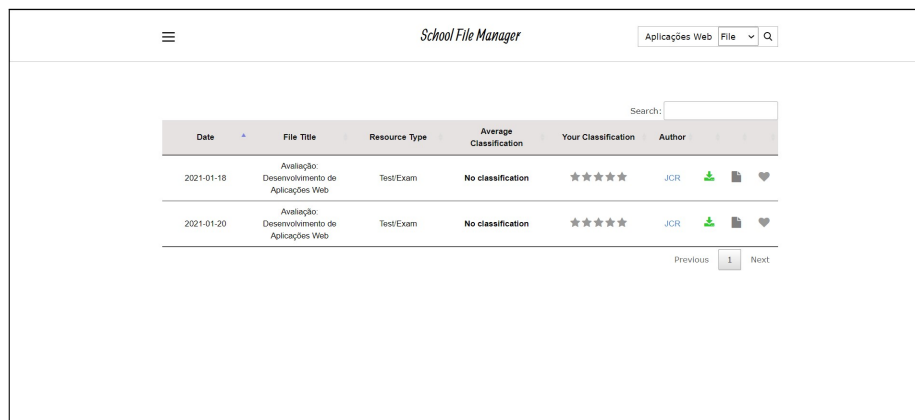


Figura 10: Página de uma pesquisa efetuada.

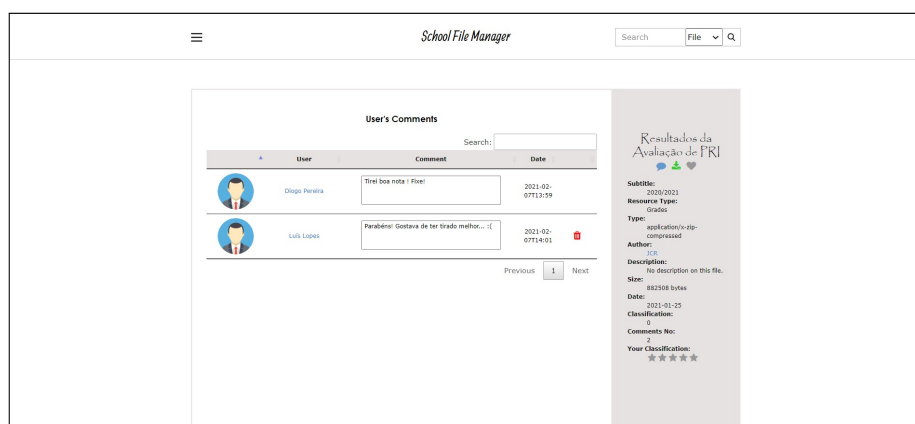


Figura 11: Página sobre um determinado recurso.

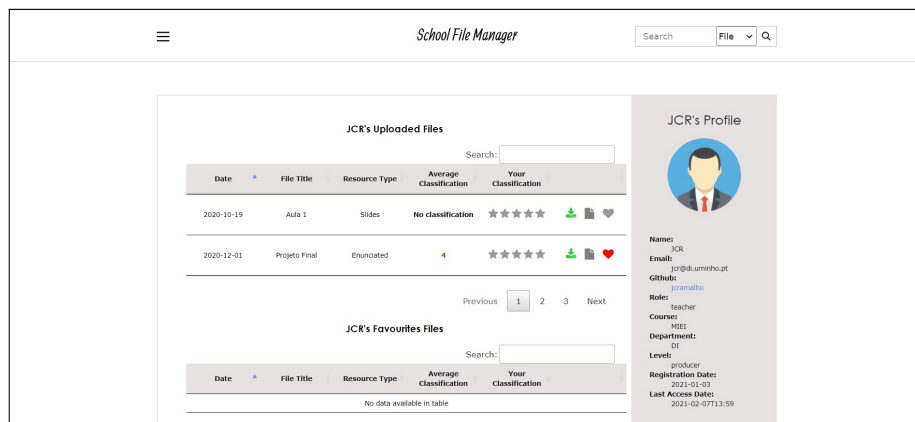


Figura 12: Página sobre um determinado utilizador.

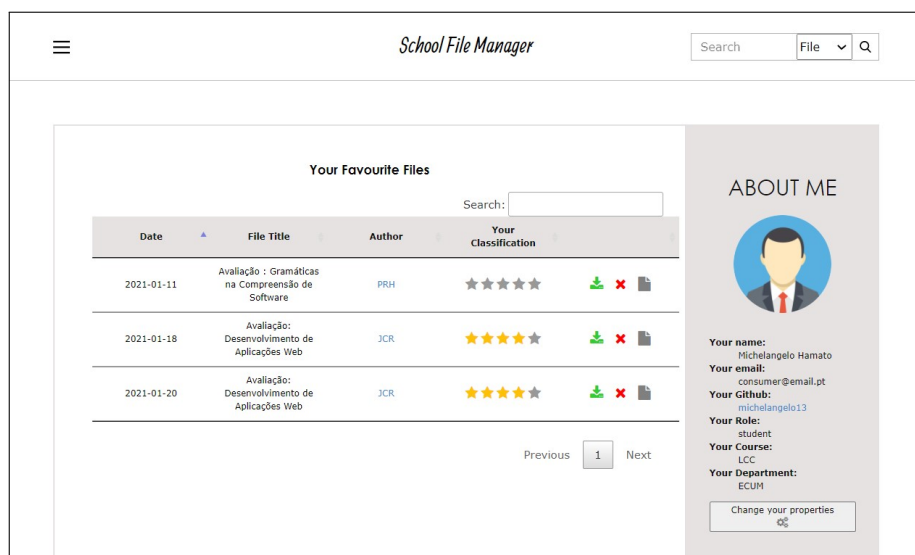


Figura 13: Página pessoal de um consumidor.

School File Manager

File

ALL FILES

Search:

Date	File Title	Resource Type	Author	Author Name	Privacy	
2016-05-20	Compiladores	Application	jcr@di.uminho.pt	JCR	Private	
2017-06-13	Mini-Testes de PI	Test/Exam	a85387@uminho.pt	Luís Lopes	Public	
2019-06-13	Fichas de Algoritmos e Complexidade	Files	a84092@uminho.pt	Diogo Pereira	Public	
2020-06-13	Material de PL	Slides	jj@di.uminho.pt	JJ	Public	
2020-10-19	Aula 1	Slides	jcr@di.uminho.pt	JCR	Public	
2020-10-20	TPC1 de PRI	Student Work	a83732@uminho.pt	Gonçalo Pinto	Public	

Previous

1

2

3

Next

ALL USERS

Search:

Email	Name	Level	Registration Date	Git	Role - Course	Department	
a60201@uminho.pt	Tiago Moreira	consumer	2021-02-07	TiagoMoreira10	student - MIEI	DI	
a67645@uminho.pt	António Gomes	consumer	2021-02-07	A67645	student - MIEI	DI	

Figura 14: Página de gestão de utilizadores e recursos a qual só o administrador têm acesso.

School File Manager

File

Error

Look like you're lost,
check your credentials !
... débrouillez-vous ...

Go to Home

Figura 15: Página de erro.

7 Conclusão

O presente relatório descreveu, de forma sucinta, o trabalho de análise, planeamento e implementação de uma aplicação web que disponibiliza recursos educativos.

Durante a elaboração deste trabalho surgiram algumas dificuldades que, com o esforço e entusiasmo do grupo pelo resultado, acabaram por ser ultrapassadas. Entre as quais destacam-se o desafio que foi a realização do processo de autenticação, mais propriamente na utilização e gestão dos *tokens* que são atribuídos de forma a validar os autores dos pedidos aos serviços tal como o seu nível de acesso.

Numa perspetiva futura, considera-se que seria interessante a autenticação através da *google* ou *facebook* facilitando assim o registo na aplicação e uma possível integração com o *google drive* com objetivo de carregar os documentos presentes nessa plataforma para a aplicação desenvolvida.

Após a realização deste trabalho, o grupo ficou consciente da importância da criação de estruturas à representação da informação além de que uma boa escolha das ferramentas contribui bastante para o consumo e distribuição da informação.

Consideramos que os objetivos propostos com a realização deste trabalho foram cumpridos, bem como a consolidação dos conhecimentos no processamento e representação da mais diversa informação.

Por fim, o grupo espera que os conhecimentos obtidos e consolidados sejam de enorme utilidade tendo uma perspetiva futura.

Referências

- [1] Express. 2019. URL: <https://expressjs.com/> (acedido em 06 de Fevereiro de 2021).
- [2] Node.js. 2020. URL: <https://nodejs.org/en/> (acedido em 06 de Fevereiro de 2021).
- [3] MongoDB. 2020. URL: <https://www.mongodb.com/> (acedido em 06 de Fevereiro de 2021).
- [4] Mongoose. 2018. URL: <https://www.npmjs.com/package/mongoose> (acedido em 06 de Fevereiro de 2021).
- [5] What are Microservices? AWS. 2021. URL: <https://aws.amazon.com/pt/microservices/> (acedido em 06 de Fevereiro de 2021).
- [6] What is an API? In English, please. Freecodecamp. 2019. URL: <https://www.freecodecamp.org/news/what-is-an-api-in-english-please-b880a3214a82/> (acedido em 06 de Fevereiro de 2021).
- [7] Trabalhando com JSON. MDN Web Docs. 2020. URL: <https://developer.mozilla.org/pt-BR/docs/Aprender/JavaScript/Objetos/JSON> (acedido em 06 de Fevereiro de 2021).
- [8] JSON Web Token. 2010. URL: <https://jwt.io/> (acedido em 06 de Fevereiro de 2021).
- [9] BagIt. 2018. URL: <https://tools.ietf.org/id/draft-kunze-bagit-16.html> (acedido em 06 de Fevereiro de 2021).
- [10] Nicepage. 2018. URL: <https://nicepage.com/> (acedido em 06 de Fevereiro de 2021).
- [11] Ajax. 2017. URL: <https://api.jquery.com/jquery.ajax/> (acedido em 06 de Fevereiro de 2021).
- [12] Datatables. 2019. URL: <https://github.com/DataTables/DataTables> (acedido em 06 de Fevereiro de 2021).
- [13] jQuery. 2018. URL: <https://api.jquery.com/> (acedido em 06 de Fevereiro de 2021).
- [14] OpenSSL. 2020. URL: <https://www.openssl.org/docs/> (acedido em 06 de Fevereiro de 2021).
- [15] jsonwebtoken. 2019. URL: <https://www.npmjs.com/package/jsonwebtoken> (acedido em 06 de Fevereiro de 2021).
- [16] adm-zip. 2021. URL: <https://www.npmjs.com/package/adm-zip> (acedido em 06 de Fevereiro de 2021).