

Universidade do Minho
Escola de Engenharia

EduShare - Plataforma de Recursos Educativos

Trabalho Prático de Desenvolvimento de aplicações Web
MEI - 4º Ano - 1º Semestre

PG42820 Constança Machado Aires Lobo Elias
PG42844 Maria Laura De Araújo Barbosa

7 de fevereiro de 2021

Conteúdo

1	Introdução	3
1.1	Contextualização	3
1.2	Estrutura do Relatório	3
2	Análise e especificação do problema	4
2.1	Descrição informal do problema	4
2.2	Especificação dos requisitos	4
3	Concepção da resolução	5
3.1	Arquitectura da Aplicação	5
3.2	Bases de dados	5
3.2.1	Users	6
3.2.2	Posts	6
3.2.3	Resources	7
3.3	Implementação	8
3.3.1	Autenticação e Autorização	9
3.3.2	FileStore	9
3.3.3	Upload	9
3.3.4	Utilizadores	10
3.3.5	Funcionalidades do Administrador	10
3.3.6	Página de um Recurso	10
3.3.7	Página Principal	11
4	Testes e Resultados	12
4.1	Dataset	12
4.2	Plataforma eduShare	12
5	Utilização da ferramenta	16
6	Conclusão	17

Lista de Figuras

3.1	Flowchart da arquitectura da ferramenta	5
4.1	Página de login	13
4.2	Página inicial do site	13
4.3	Página que permite ao user realizar upload de Recursos.	14
4.4	Resultado da pesquisa, aplicando o filtro <i>daw</i> no titulo.	14

Capítulo 1

Introdução

1.1 Contextualização

Este trabalho foi desenvolvido no âmbito da unidade curricular de Desenvolvimento de Aplicações Web. O objectivo principal do projeto consiste no desenvolvimento de uma plataforma de Gestão e Disponibilização de recursos educativos, colocando em prática os conteúdos abordados na unidade curricular ao longo do semestre. A aplicação foi desenvolvida em *NodeJS*, utilizando a framework *Express*, e a persistência dos dados através de bases de dados MongoDB.

1.2 Estrutura do Relatório

Este relatório divide-se em cinco partes principais:

- A primeira parte consiste na análise do problema, dos requisitos e da arquitetura da ferramenta desenvolvida.
- Na segunda parte, é explicada a implementação do projeto e o trabalho desenvolvido.
- A terceira corresponde à explicação da interface web e dos resultados obtidos.
- A quarta explica como proceder à utilização da aplicação.
- A quinta contém uma breve síntese do trabalho realizado, apresentando as principais conclusões do mesmo bem como a indicação do trabalho futuro que poderia ser desenvolvido.

Capítulo 2

Análise e especificação do problema

2.1 Descrição informal do problema

Este projeto consiste na criação de uma plataforma de Gestão e Disponibilização de Recursos educativos. Pretende-se obter uma aplicação capaz de receber recursos pedagógicos, importá-los para a plataforma num formato universal e armazená-los, de forma a ser possível fazer a exportação desses recursos sempre que necessário. Em simultâneo, a ferramenta deverá ter uma componente social, permitindo aos utilizadores comentar, avaliar e partilhar os conteúdos previamente adicionados.

2.2 Especificação dos requisitos

Os requisitos principais para esta ferramenta passam por:

- Permitir adicionar novos recursos.
- Classificar cada recurso por ano, tipo, tema.
- Permitir que o utilizador faça um Post sobre um recurso.
- Permitir comentários nos Posts.
- Criar um sistema de ranking para os recursos.
- Criar um sistema de autenticação de utilizadores, permitindo adicionar novos utilizadores.
- Apresentar, na página principal da interface Web, notificações das ações realizadas pelos vários users.
- Permitir exportar e importar os recursos, no mesmo formato.

Capítulo 3

Concepção da resolução

3.1 Arquitectura da Aplicação

A arquitectura da nossa aplicação é composta por três servidores, como se apresenta na figura 3.1. O app-server é responsável pela interação com o utilizador. O auth-server responde a pedidos de dados de utilizadores e é responsável por gerar o token que assina os pedidos ao api-server. O api-server responde a pedidos de dados de recursos e posts.

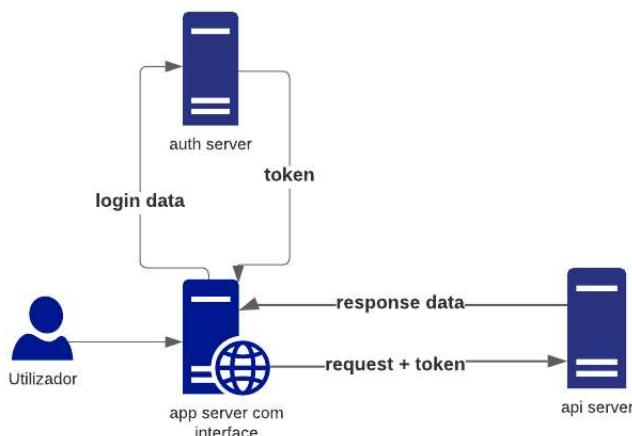


Figura 3.1: Flowchart da arquitectura da ferramenta

3.2 Bases de dados

As base de dados são responsáveis por guardar os dados dos utilizadores do sistema, dos posts realizados e dos recursos arquivados. Tal como indicado anteriormente, a persistência destes dados foi feita em MongoDB.

Adicionalmente foi criada uma funcionalidade de *backUp* do conteúdo das base de dados via API. O script *backup.js* que se encontra na directória principal do projeto, quando executado, coloca o conteúdo actual das três bases de dados nos ficheiros que se encontram na pasta **datasets** através de invocações do comando *mongoexports*. Para

criar e popular as bases de dados, foi feito um script, *populate.js*, que insere na base de dados o conteúdo dos ficheiros que se encontram na directória `./datasets`.

Apresentamos, em seguida, uma breve explicação do modelo dos documentos de cada base de dados.

3.2.1 Users

Para cada utilizador considerámos importante guardar o seu nome, e-mail (que serve de id), password, filiation (student, teacher ...), registerDate e lastAccessDate. Possui ainda o campo level que indica o tipo de usuário (admin, producer ou consumer). Estes dados são guardados na base de dados `users`.

Users model

```
var user = new mongoose.Schema({
  name: String,
  mail: String,
  password: String,
  filiation: String,
  level: String,
  registerDate: Date,
  lastAccessDate: Date
});
```

3.2.2 Posts

Em relação aos Posts, o grupo considerou importante, além de criar um id, guardar a seguinte informação associada a um Post:

- O id do ficheiro ao qual o post está associado;
- O nome do ficheiro ao qual o post está associado;
- O conteúdo do post.
- Data e Hora em que o post foi realizado.
- Id do utilizador que realizou o Post.
- O tipo de post.
- Para um post do tipo *file* ou *share*, é guardada uma lista de comentários associados. O esquema de um comentário possui apenas um id, uma data, o conteúdo e a identificação do user (mail) que o realizou.

Os posts podem ter quatro tipos: *comment*, *ranking*, *share* ou *file*. O post que está associado à publicação de um ficheiro é do tipo *file*. Se uma publicação está relacionada com a partilha de um ficheiro e tem associada uma descrição, é do tipo *share*. Caso o post esteja associado à escrita de um comentário por parte de um user é do tipo *comment* e se o post está associado à avaliação de um ficheiro é do tipo *ranking*. Estes dois últimos tipos de posts são utilizados para apresentar as notificações ao utilizador.

De acordo com o tipo de post, os campos apresentados no modelo abaixo podem não ser todos utilizados. Os ids dos posts e dos comentários são criados de forma aleatória recorrendo ao gerador aleatório *uuidv4()*.

Post model

```
var commentSchema = new mongoose.Schema({
  text : String,
  user_name : String,
  date : String,
  id:String
})

var postSchema = new mongoose.Schema({
  id: String,
  id_file:String,
  name_file:String,
  description: String,
  date: String,
  time:String,
  user_name:String,
  postType:String, // file, comment, share, ranking
  comments: [commentSchema]
})
```

3.2.3 Resources

Nesta base de dados guardou-se a informação relevante associada a cada recurso que é inserido na plataforma. Cada um dos recursos possui um registo na base de dados que contém os seguintes campos:

- O Id do recurso;
- O título do recurso;
- O subtítulo do recurso;
- O nome do recurso, que corresponde ao nome do ficheiro inserido.
- O mimeType do recurso;
- O tamanho do recurso em mega bytes;
- O id do user que o inseriu;
- A lista de Autores;
- Data de Upload;
- Data da última atualização;
- O tipo de ficheiro (isto é, a classificação do recurso, que pode ser do tipo Test, Report, Slides, Exercises ou Others)
- O assunto que aborda o Ficheiro;
- O ranking atual do ficheiro;
- Uma lista de ranking. Que associa a cada user a avaliação por ele introduzida.

- A visibilidade do Recurso (public ou privado)
- Uma lista de hashtags associadas.

Resource model

```
var resourceSchema = new mongoose.Schema({
  id : String,
  title:String,
  subtitle:String,
  name:String,
  mimeType:String,
  size:String,
  id_produces: String,
  authors:[String],
  dateLastUpdate: String,
  timeLastUpdate: String,
  type : String,
  subject : String,
  description: String,
  id_post:String,
  rankingList: [ { id_user : String , ranking : Number}],
  ranking: Number,
  visibility:String,
  hashtags: [String]
});
```

3.3 Implementação

A solução desenvolvida para o problema previamente identificado foi distribuída por três servidores que comunicam entre si, como foi explicado anteriormente. São eles:

- **app-server** - Um servidor Web que funciona como ponto principal da nossa solução, permitindo ao utilizador interagir com a ferramenta (registar-se, iniciar sessão, consultar o feed de notícias, carregar ficheiros para a ferramentas, consultar recursos e interagir com outros utilizadores). Foi desenvolvido em *NodeJS* e em *Express* e encontra-se à escuta de pedidos no endereço <http://localhost:7002/>.
- **api-server** - Servidor responsável por gerir as bases de dados dos Posts e Recursos. Por defeito, quando inicializado, o módulo *moongoose* conecta à base de dados *default*, estabelecendo depois conexão com a base de dados dos Posts ou Recursos de acordo com o modelo que se encontra a verificar. Este servidor encontra-se ‘à escuta de pedidos na porta 7001.
- **auth-server** - Servidor responsável por gerir a base de dados com os dados dos utilizadores. Ao iniciar, estabelece conexão com a base de dados *users*. É o responsável pela autenticação e criação dos tokens de acesso à API de dados. Encontra-se à escuta de pedidos na porta 7003.

Em seguida, faremos uma breve explicação relativa às várias funcionalidades criadas, à medida que são explicadas as principais decisões tomadas.

3.3.1 Autenticação e Autorização

A autenticação na aplicação é feita através de JSON Web Tokens (JWT). Foi adotado este método de autenticação pelo facto de a utilização de *JWTs* ser de fácil implementação, não necessitar de base de dados e permitir, de forma eficiente, proteger a nossa aplicação. No nosso sistema, os *JWTs* são alocados em cookies, com tempo de expiração. Quando um token expira, é enviado ao utilizador a mensagem de que a sessão expirou pelo que deve ser feito um novo login. Quando é feito logout, o JWT é apagado da cookie pelo que, até novo login, as rotas da aplicação não têm autorização para serem acedidas.

Cada vez que o client faz um pedido ao app-server, o servidor vai buscar o JWT ao cookie, no middleware vertical do app-server, e depois processa o pedido, enviado-o ao api-server assinado com o token. O api-server, por sua vez, possui também middleware vertical, que verifica, antes de aceder a qualquer rota, se o token é válido.

3.3.2 FileStore

A pasta fileStore encontra-se na directória *api-server* e contém os recursos que alimentam a base de dados. Esta pasta encontra-se dividida em sub-pastas, cada uma correspondente a um tipo de recurso. Neste momento existem 5 tipos de recursos (Test, Report, Exercises, Slides e Others) e por isso a pasta fileStore possui cinco sub-diretorias, contendo em cada uma pastas associadas a recursos classificados com esse tipo. O nome da pasta de cada recurso corresponde ao id do recurso.

3.3.3 Upload

Um utilizador que pretende carregar recursos na plataforma, deve preencher os campos do formulário de Upload. Ao submeter, é executado um conjunto de ações: que começam com a leitura dos dados preenchidos e terminam com a inserção do recurso na pasta correspondente na fileStore e na base de dados. Além disto, é criado um post associado a esta inserção que é guardado na respetiva base de dados. É possível realizar upload de um ficheiro único ou de um recurso Zipado.

Formato da pasta criada

Depois do user realizar upload do recurso, a ferramenta guarda-o numa pasta, cujo o nome corresponde ao id gerado para o ficheiro. Por exemplo, para o recurso daw2020.pdf, cujo o id é r_123 e que está associado ao tipo Test, temos a seguinte estrutura:

Estrutura fileStore

```
-fileStore
  -Test
    -r_123
      - data
        - daw2020.pdf
      -manifesto-sha256
      -bag.txt
      -bag-info.txt
```

3.3.4 Utilizadores

A plataforma pode ser utilizada por três tipos de utilizadores:

- **Consumer** - Um utilizador quando se regista é enquadrado nesta categoria, tem acesso a todos os recursos públicos e pode interagir na plataforma realizando *posts*, comentários e avaliações. Adicionalmente, consegue apagar os seus posts e comentários e avaliar recursos públicos.
- **Producer** - Um consumer passa a ser producer quando realiza o primeiro upload. Mantém as funcionalidades anteriores, passando agora a dispor de uma página *MyResources* onde pode consultar a lista de recursos que já inseriu na plataforma, editá-los ou até removê-los.
- **Admin** - Este utilizador tem permissões especiais. Além de realizar todas as ações que são permitidas aos outros utilizadores, pode ainda eliminar comentários, eliminar posts e recursos de um qualquer utilizador. Tem também autorização para exportar e importar recursos da plataforma, no formato *BagIt*. Os utilizadores de nível administrador podem apenas ser criados via API-server.

3.3.5 Funcionalidades do Administrador

Ao desenvolver esta aplicação, tivemos em conta a necessidade futura de se extrair o seu conteúdo de modo a ser possível incorporar noutra APP. Foram então criadas duas funcionalidades que permitem extrair um recurso e importar um recurso de outra plataforma para a eduShare. Estas duas funcionalidades são restritas ao administrador. Um utilizador normal pode apenas fazer *upload* e *download* de ficheiros simples.

Exportar

Quando esta ação é invocada. A pasta que contém o recurso é transferida para uma directória auxiliar e o ficheiro *bag.txt* é substituído pelo ficheiro *bag.json*. Este contém todas as informações sobre o recurso que se encontra na bases de dados. Em seguida, a pasta é zipada e descarregada para o computador do administrador.

Importar

O objetivo desta funcionalidade é garantir a transferência de recursos entre plataformas. Assim, o sistema admite importar recursos que se encontrem na estrutura previamente descrita, zipados e que possuam um ficheiro *bag.json*. O sistema carrega esta pasta para uma directória temporária, descompacta-a e procede à leitura do conteúdo do ficheiro *bag.json*, importando os dados associados para a base de dados. Por último, a pasta é transferida para a posição correta na *fileStore*.

3.3.6 Página de um Recurso

Esta página encontra-se apenas acessível para recursos públicos. Permite visualizar, comentar e descarregar os ficheiros.

3.3.7 Página Principal

Na página principal, são apresentados:

- Top 10 da lista de recursos com melhor avaliação;
- Feed de notícias, que apresenta os últimos recursos inseridos e os posts realizados sobre esses recursos.
- Notificações dos comentários e avaliações que são realizados na plataforma.

Capítulo 4

Testes e Resultados

4.1 Dataset

Para testar a plataforma, utilizámos três datasets que se encontram dentro da pasta *Datasets*, na directória principal. Para importar o seu conteúdo para as bases de dados deverá executar-se ***node populate.js***.

Foram criados aproximadamente 1000 utilizadores, incluindo um utilizador com permissões de administrador.

Em relação aos recursos, utilizámos, para o teste inicial, ficheiros facultados pelos docentes da UMinho ao longo do nosso ciclo de estudos. Para facilitar a incorporação dos mesmos no API-Server, criou-se um script *resourcesGenerator.js* responsável por retirar todos os ficheiros que se encontram na pasta *UploadApi*, colocá-los no formato correto, adicionar a entrada corresponde na base de dados dos *Resources* e *Posts* e, por último, transferir para a posição correta na fileStore. Importa referir que por este ser um processo automático, algumas das características não são preenchida. Nomeadamente, no recurso, os campos *Subject*, *Description* e *subTitle* ficam em branco. O título, por sua vez, é o nome do ficheiro sem extensão. O *id_producer* é preenchido com o e-mail de utilizadores registados na base de dados cujo o estatuto seja de *producer*. O dataset final para teste da plataforma é composto por aproximadamente 550 recursos. A lista de posts associados a estes recursos é gerada à medida que cada recurso é criado.

4.2 Plataforma eduShare

Nesta secção pretendemos explicar, de um modo geral, o funcionamento da aplicação à medida que apresentamos algumas das funcionalidades presentes no site.

Um utilizador que aceda à plataforma pela primeira vez, deve começar por se registar, no formulário criado para o efeito, fornecendo o seu e-mail, nome, password e filiação (student, professor ou outro). Os dados fornecidos serão guardados na base de dados *users*, ficando disponíveis para serem acedidos sempre que tal seja necessário. Em seguida, o utilizador necessita de iniciar sessão na conta criada, sendo redirecionado para a sua página principal.

A figura 4.1 mostra a página de login da aplicação.

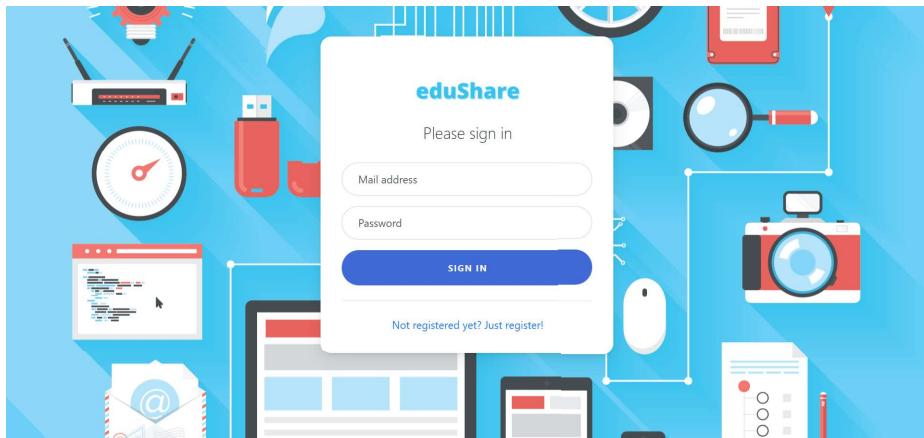


Figura 4.1: Página de login

A figura 4.2 permite visualizar a página inicial da plataforma. Esta é constituída, no canto superior, por botões que conduzem às várias funções da ferramenta. Ao centro, é apresentado um resumo dos últimos *Posts*. Além disso, no lado esquerdo, podemos ver o top 10 dos recursos com melhor avaliação e, no lado direito, são expostas as notificações da APP.

Most starred resources ★		Recent Activity	Notifications
trabalho 2 de ts	★★★★★ 5 / 5	middletonmays@playce.com 2021-09-7 229532 Inserted a new file TC-TrabalhosPraticos.pdf	minnienorman@playce.com ranked the file Ficha4 - Texturas.pdf
trabalho final de aib - 2020/2021	★★★★★ 5 / 5		minnienorman@playce.com commented on file TS.Trabalho_2.pdf
ficha texturas - cg - uminho	★★★★★ 5 / 5		middletonmays@playce.com ranked the file SO.pdf
trabalhos práticos de tc	★★★★★ 5 / 5	middletonmays@playce.com 2021-09-7 229532 [UMINHO - CG] Falei com o professor Pedro, que me disse que o teste de amanhã é sobre texturas. Ele recomendou que se resolva a esta ficha..	middletonmays@playce.com ranked the file progC-salasCinema.zip
trabalho sistemas operativo 2018	★★★★★ 5 / 5	Ficha4 - Texturas.pdf	middletonmays@playce.com ranked the file TC-TrabalhosPraticos.pdf
TS_TP3	★★★★★ 4 / 5		middletonmays@playce.com ranked the file SO.pdf

Figura 4.2: Página inicial do site

A imagem 4.3 corresponde à funcionalidade de *upload* de um recurso simples por um utilizador comum, e mostra o formulário que é apresentado ao utilizador com os vários campos que devem ser preenchidos para carregar o ficheiro.

Figura 4.3: Página que permite ao user realizar upload de Recursos.

Resource	Hashtags	Rating	Last update
daw - enunciado do trabalho others - jcr	daw, trabalho	0 ★	2021-02-7 by maria.a.b2000@hotmail.com

Figura 4.4: Resultado da pesquisa, aplicando o filtro *daw* no titulo.

É ainda permitido ao utilizador editar os meta-dados associados aos seus ficheiros bem como eliminar os seus ficheiros. Isto deve ser feito na sua área pessoal, acedendo ao separador *MyResources*.

Cada utilizador dispõe ainda de uma página de perfil, na qual pode verificar os seus dados pessoais e consultar a lista de recursos que possui.

Capítulo 5

Utilização da ferramenta

Para utilizar e testar a ferramenta devem executar-se os seguintes passos:

1. Instalar os módulos necessários. Nas directorias *api-serve*, *app-server*, *auth-server* executar:

npm i

2. Povoar as bases de dados: Na directoria principal *DAW2020_eduShare* executar o comando:

node populate.js

3. Iniciar os servidores: Na pasta *app-server* executar o comando:

npm run all

4. Aceder à página da aplicação Web: endereço **http://localhost:7002/**.

Para novos utilizadores, deve ser criado um novo registo. Os utilizadores registados devem apenas iniciar sessão para aceder a plataforma. É fundamental referir que todas as rotas se encontram protegidas pelo que é necessário um utilizador efetuar o login para poder aceder aos ficheiros, mesmo que seja apenas para visualização.

Capítulo 6

Conclusão

Este relatório descreveu o problema, fez uma análise da conceção da resolução e forneceu uma explicação de todo o trabalho desenvolvido. Ao elaborar o que foi pedido, o grupo achou necessário criar três bases de dados para *users*, *Posts* e *Resources* que garantiram a persistência dos dados e facilitaram a apresentação dos dados na interface da aplicação. Foram criadas várias rotas para satisfazer os pedidos necessários. Tentou-se ainda desenvolver uma interface web que fosse apelativa ao utilizador. Deste modo, a resolução do trabalho mostrou-se desafiante no sentido de chegar à solução obtida e consideramos que os requisitos foram alcançados com sucesso.

A nível de trabalho futuro, várias funcionalidades poderiam ser modificadas e aperfeiçoadas. Nomeadamente, a nível de segurança podem ser cifradas as passwords antes de serem guardadas. Poderiam ser implementadas novas funcionalidades como permitir a autenticação com a Google ou o Facebook, bem como permitir a adição de novos tipos de recursos. A nível de pesquisa poderiam ser adicionados mais filtros (para os diversos metadados dos recursos) bem como a procura de utilizadores.