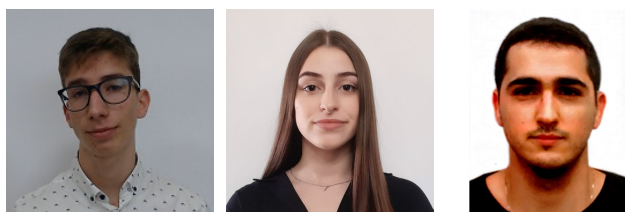




Universidade do Minho
Mestrado Integrado em Engenharia Informática
4ºano - 1º Semestre

Desenvolvimento de Aplicações *Web*

Plataforma de Gestão e Disponibilização de Recursos Educativos



a84197 – João Pedro Araújo Parente
pg44412 - Ana Margarida da Rocha Ferreira
pg42814 – Raimundo Nonato Barros Neto

7 de fevereiro de 2021

Conteúdo

1	Introdução	3
2	Conceitos Implementados	4
3	Persistência de Dados	5
3.1	Política de Armazenamento	5
4	Modelos	6
4.1	Utilizadores	6
4.1.1	Níveis de Acesso	7
4.1.2	Autenticação	8
4.2	Tipos	9
4.3	Recursos	10
5	Pacotes SIP, AIP e DIP	12
5.1	Manifesto	12
6	Importação/Exportação	14
6.1	Utilizadores e Tipos	14
6.2	Recursos	14
7	Rotas	16
7.1	Utilizadores - /utilizadores	16
7.2	Index - /	17
7.3	Recursos - /recursos	17
7.4	Tipos - /tipos	18
7.5	Publicações - /publicacoes	18
7.6	Importação - /import	19
7.7	Exportação - /export	19
8	Conclusão	20
9	Referências	21

Lista de Figuras

1	Exemplo de um documento relativo a um utilizador.	7
2	Exemplo de um documento relativo a um tipo de ficheiro. . . .	9
3	Exemplo de um documento relativo a um recurso na base de dados.	11
4	Exemplo do conteúdo de um SIP ou DIP.	12
5	Exemplo do conteúdo de um manifesto.	13
6	Exemplo de um utilizador em formato CSV.	14
7	Exemplo de um tipo em formato CSV.	14
8	Exemplo do ficheiro ZIP que constitui um <i>export</i> de recursos. .	15
9	Exemplo do ficheiro CSV de recursos.	15

1 Introdução

O presente trabalho prático foi realizado no âmbito da unidade curricular de Desenvolvimento de Aplicações *Web* e consistiu no desenvolvimento de uma plataforma de gestão e disponibilização de recursos educativos, isto é, um servidor aplicacional que recebe pedidos em determinadas rotas e fornece uma página HTML adequada, permitindo ao utilizador registar-se, fazer *login*, fazer *upload* e *download* de documentos, comentar publicações, tudo a partir do seu *browser*.

A plataforma desenvolvida teve por base o modelo ***Open Archival Information System (OAIS)***. Assim, a aplicação possui **três níveis de acesso**, permitindo que os utilizadores sejam consumidores (apenas podem consultar e descarregar recursos públicos), produtores (além das permissões do consumidor podem carregar arquivos e editá-los) ou administradores (têm permissão para todas as operações). Deste modo, foi necessária a autenticação dos utilizadores e a gestão de níveis de acesso.

Além disso, a plataforma constitui **três processos** principais, nomeadamente a ingestão, que corresponde ao depósito de informação no sistema, a administração, que corresponde a um conjunto de operações CRUD sobre os recursos armazenados e, por último, a disseminação, que corresponde às operações do consumidor, isto é, a extração de informação do sistema. Assim, existem, também, **três pacotes**: o pacote de submissão de informação (SIP), o pacote armazenado (AIP) e o pacote de disseminação de informação (DIP).

Para o desenvolvimento do presente projeto foram utilizadas várias tecnologias, nomeadamente a *framework* Express para Node.js, que permite a otimização da construção de aplicações *web*, e a base de dados MongoDB, que se trata de uma base de dados NoSQL, orientada a documentos.

2 Conceitos Implementados

Tendo em conta os objetivos propostos para o presente projeto, referidos na Secção 1, na plataforma desenvolvida foram implementados os seguintes conceitos:

- **Ficheiro:** arquivo de computador, que tem um tipo associado.
- **Tipo de ficheiro:** extensão de um ficheiro e *metadata* adicional que pode conter.
- **Utilizador:** representação de um cliente no sistema.
- **Recurso:** conjunto de ficheiros.
- **Publicação:** referente a um recurso do sistema, que contém um comentário feito pelo criador da publicação e aceita outros comentários dos utilizadores.
- **Pacote SIP:** formato que um recurso deve seguir para ser inserido no sistema.
- **Pacote AIP:** formato de um recurso armazenado no sistema.
- **Pacote DIP:** formato que o sistema usa para partilhar a informação com os clientes.

3 Persistência de Dados

A persistência dos dados da plataforma, relativamente aos utilizadores, recursos e tipos de recursos, foi feita recorrendo ao sistema de base de dados MongoDB, que segue o modelo não relacional e é orientado a documentos. Deste modo, na MongoDB foram criadas três coleções: **Utilizadores**, **Recursos** e **Tipos**, que corresponderam aos modelos implementados no servidor aplicacional, recorrendo à biblioteca `mongoose`.

Relativamente aos ficheiros correspondentes aos recursos introduzidos na plataforma, estes foram armazenados numa pasta denominada `fileStore`, sendo que se no registo do recurso na base de dados se fez referência ao caminho no qual os seus ficheiros se encontram.

3.1 Política de Armazenamento

Uma vez que existe um limite de número de arquivos dentro de uma pasta, achou-se pertinente criar uma política de armazenamento, a partir de informações do recurso.

A política de armazenamento implementada foi simples e consistiu na divisão dos recursos por pastas dentro da `fileStore`, consoante o primeiro *hashtag*. Deste modo, os ficheiros de recursos com o mesmo *hashtag* inicial ficam guardados dentro da mesma pasta, cujo nome é o próprio *hashtag*. Esta política, permite, assim, evitar em maior parte dos casos alcançar o limite de ficheiros por pasta, acreditando que os recursos têm *hashtags* mais ao menos uniformemente distribuído.

É de salientar que recursos que possuam uma *hashtag* com caracteres especiais, não suportados pelo sistema de ficheiros, são armazenados numa pasta denominada `others`.

4 Modelos

Tal como foi referido, as entidades representadas no servidor aplicacional e armazenadas na base de dados foram os **Utilizadores**, os **Recursos** e os **Tipos**.

4.1 Utilizadores

Na plataforma desenvolvida cada utilizador tem os seguintes campos, que serão registados na base de dados:

- **_id:** *username* do utilizador, do tipo *String*, que corresponde ao seu identificador único.
- **password:** *password* do utilizador do tipo *String*.
- **dataRegisto:** data e hora em que o utilizador foi registado, do tipo *Date*.
- **dataUltimoAcesso:** data e hora da última vez que o utilizador efetuou *login* no sistema, do tipo *Date*.
- **nivel:** nível de acesso do utilizador no sistema, que pode ser um inteiro de 0 a 2.
- **email:** email do utilizador, do tipo *String*.
- **nome:** nome do utilizador, do tipo *String*.
- **filiacao.estudante:** é atribuído o booleano *true* ou *false* caso o utilizador seja estudante ou não, respetivamente.
- **filiacao.docente:** é atribuído o booleano *true* ou *false* caso o utilizador seja docente ou não, respetivamente.
- **filiacao.departamento:** departamento associado ao utilizador, do tipo *String*.
- **filiacao.curso:** curso associado ao utilizador, do tipo *String*.

Na Figura 1 consta um exemplo de um documento da base de dados relativo ao utilizador com o *username* "joao123".

```

{
  "_id": "joao123",
  "dataRegisto": {
    "$date": "2021-01-26T09:22:00Z"
  },
  "dataUltimoAcesso": {
    "$date": "2021-01-27T13:40:00Z"
  },
  "nivel": 0,
  "nome": "João Pedro Afonso",
  "email": "jpbp.pfdrp@gmail.com",
  "filiacao": {
    "estudante": true,
    "docente": false,
    "curso": "Miei",
    "departamento": "Informática"
  },
  "password": "123"
}

```

Figura 1: Exemplo de um documento relativo a um utilizador.

4.1.1 Níveis de Acesso

A cada utilizador é associado um nível, sendo que cada nível atribui diferentes permissões, isto é, restringe as operações a que o utilizador tem acesso.

- **Nível 0 - Consumidor**

A um consumidor apenas é permitido visualizar e fazer *download* de recursos, editar a informação referente a ele próprio, criar publicações sobre um recurso, editar ou remover as publicações feitas por si e comentar publicações.

- **Nível 1 - Produtor**

Um produtor têm as permissões de um consumidor e consegue, ainda, dar *upload* a recursos e editar recursos criados pelo mesmo.

- **Nível 2 - Administrador**

Um administrador têm as permissões dos níveis anteriores e tem, ainda, a possibilidade de criar, editar e remover todos os tipos/recursos, além

de poder editar e remover utilizadores e fazer *import* ou *export* de utilizadores, recursos ou tipos.

4.1.2 Autenticação

Tendo em conta que se implementou diferentes níveis de acesso no servidor aplicacional desenvolvido, a autenticação e gestão de níveis de acesso é fundamental para distinguir os utilizadores, uma vez que dependendo do nível do utilizador as ações que permitidas variam.

Deste modo, utilizou-se sessões no servidor aplicacional e *cookies* nos clientes, através dos módulos do Node.js: `cookie-parser`, `express-session`, `session-file-store`, `uuid`, `passport` e `passport-local`. Preferiu-se esta abordagem à utilização de JSON *Web Tokens*, dado que existe apenas um servidor que trata de autenticação e responde pedidos e a autenticação é mais simples e permite implementar todas as funcionalidades pretendidas.

4.2 Tipos

A plataforma desenvolvida permite a criação de vários tipos de recursos e ficheiros. Assim, foi necessário criar um modelo para o tipo, que apresenta os seguintes campos que são guardados na base de dados:

- **_id**: corresponde ao nome e identificador único deste tipo, do tipo *String*.
- **parametros**: *array* de *metadata* referente a este tipo, em que cada elemento do *array* tem duas propriedades: o **nome_param** que especifica o nome do parâmetro e o **tipo_param** que indica se este parâmetro é do tipo *String* ou *Number*.

Na Figura 2 é apresentado um exemplo de um documento da base de dados relativo ao tipo de recurso "txt".

```
{
  "_id": "txt",
  "parametros": [
    {
      "nome_param": "numero_linhas",
      "tipo_param": "Number"
    },
    {
      "nome_param": "sobre",
      "tipo_param": "String"
    }
  ]
}
```

Figura 2: Exemplo de um documento relativo a um tipo de ficheiro.

4.3 Recursos

Cada recurso é constituído por 2 partes: a *metadata* referente ao recurso que vai ser guardada na base de dados e por um conjunto de ficheiros que é guardado na pasta `fileStore` do servidor.

A *metadata* do recurso, guardada na base de dados, apresenta os seguintes campos:

- `_id`: identificador único do recurso, do tipo *String* e composto pela data e hora de criação do recurso e um número aleatório entre 0 e 1.
- `titulo`: título do recurso, do tipo *String*.
- `subtitulo`: subtítulo do recurso, do tipo *String*.
- `produtor`: nome do utilizador que inseriu o recurso, do tipo *String*.
- `dataCriacao`: data e hora em que o recurso foi criado, do tipo *Date*.
- `dataRegisto`: data e hora em que o recurso foi registado no sistema, do tipo *Date*.
- `visibilidade`: é atribuída a *String* público se qualquer utilizador puder visualizar o recurso ou privado se apenas quem inseriu o recurso o puder visualizar.
- `likes`: lista de *usernames* dos utilizadores que gostaram do recurso, em que cada elemento é do tipo *String*.
- `hashtags`: lista de *hashtags* associados ao recurso, em que cada elemento é do tipo *String*.
- `posts`: lista de publicações referentes ao recurso. Cada elemento do *array* pode apresentar o campo `meta` e o campo `coments` que corresponde a uma lista de comentários da publicação, em que cada elemento é do tipo `meta`. O campo `meta`, por sua vez, guarda o *username* (`_id`) do utilizador que fez a publicação ou o comentário, o seu nome (`nome`), a data e hora (`data`) em que foi feita a publicação ou comentário e o seu conteúdo (`conteudo`).
- `manifesto`: estrutura do recurso, do tipo *String*.
- `path`: especifica a pasta na qual se encontram guardados os ficheiros do recurso, do tipo *String*.

A Figura 3 representa um exemplo de um documento da base de dados relativo a um recurso.

```
{
  "_id": "2021-01-27T10:38:23Z-Projectos",
  "subtitulo": "3ae_Ano-1ae_Semestre-Sistemas_Distribuidos-Projectos",
  "dataCriacao": {
    "$date": "2021-01-27T10:38:23Z"
  },
  "dataRegisto": {
    "$date": "2021-01-27T10:38:23Z"
  },
  "visibilidade": "publico",
  "likes": [],
  "hashtags": ["Sistemas_Distribuidos"],
  "titulo": "Projectos",
  "produtor": "admin",
  "posts": [],
  "manifesto": "{\\"ficheiros\\":[],\\"pasta_rec\\":[{\\"nome\\":\\"2015-2016\\",\\"pasta\\":\\"2015-2016\\"}]}",
  "path": "fileStore/Sistemas_Distribuidos/2021-01-27T10:38:23Z-Projectos"
}
```

Figura 3: Exemplo de um documento relativo a um recurso na base de dados.

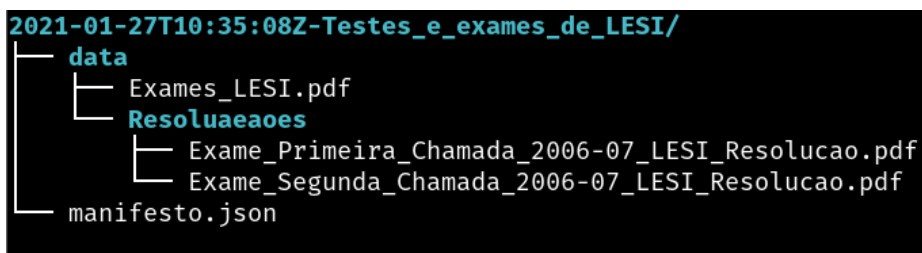
5 Pacotes SIP, AIP e DIP

Tal como foi referido na Secção 1, a plataforma desenvolvida teve por base o modelo OAIS. Segundo este modelo, na aplicação devem existir, para além dos três níveis de acesso e três processos principais de ingestão, administração e disseminação, três pacotes: SIP, AIP e DIP.

Como simplificação, considerou-se o pacote de submissão de informação (SIP) igual ao pacote de disseminação de informação (DIP). O formato escolhido para estes pacotes foi um arquivo ZIP, constituído por um ficheiro, denominado obrigatoriamente `manifesto.json` e no qual consta a estrutura dos ficheiros do recurso, e uma pasta designada `data`, que contém o recurso em si.

O pacote AIP vai ser exatamente igual ao DIP e SIP, só que não vai estar zipado, para facilitar a disponibilização de apenas parte do recurso.

A Figura 4 apresenta um exemplo do conteúdo de um SIP ou DIP, onde se pode verificar que existem dois ficheiros principais, correspondentes ao manifesto e à pasta com os ficheiros do recursos. Como se pode observar a pasta `data` contém um ficheiro PDF e uma pasta com mais dois ficheiros PDF.



```
2021-01-27T10:35:08Z-Testes_e_exames_de_LESI/
├── data
│   ├── Exames_LESI.pdf
│   └── Resolucoes
│       ├── Exame_Primeira_Chamada_2006-07_LESI_Resolucao.pdf
│       └── Exame_Segunda_Chamada_2006-07_LESI_Resolucao.pdf
└── manifesto.json
```

Figura 4: Exemplo do conteúdo de um SIP ou DIP.

5.1 Manifesto

O manifesto é um ficheiro denominado `manifesto.json` que contém uma estrutura em JSON, que deve apresentar as duas propriedades seguintes:

- **ficheiros**: lista de ficheiros em que cada elemento possui o nome do ficheiro (**nome**), o tipo (**tipo**) e a informação relativa ao mesmo (**meta**).
- **pasta_rec**: representa uma pasta que vai ser a recursividade, sendo que possui um nome (**nome**), uma propriedade **pasta** na qual estão guardados os ficheiros e pode, ainda, possuir a propriedade **pasta_rec**.

Na Figura 5 consta um exemplo do conteúdo que um manifesto pode conter. O manifesto apresentado corresponde ao presente do SIP ou DIP da Figura 4.

```
{
  "ficheiros": [{
    "nome": "Exames_LESI.pdf",
    "tipo": "pdf",
    "meta": []
  }],
  "pasta_rec": [{
    "nome": "Resolucoes",
    "pasta": {
      "ficheiros": [{
        "nome": "Exame_Primeira_Chogada_2006-07_LESI_Resolucao.pdf",
        "tipo": "pdf",
        "meta": []
      }, {
        "nome": "Exame_Segunda_Chogada_2006-07_LESI_Resolucao.pdf",
        "tipo": "pdf",
        "meta": []
      }],
      "pasta_rec": []
    }
  ]
}
```

Figura 5: Exemplo do conteúdo de um manifesto.

6 Importação/Exportação

A importação e exportação são operações que podem ser realizadas apenas pelos administradores do servidor aplicacional. Ambas recebem/devolvem o mesmo formato, sendo que se optou por utilizar ficheiros no formato CSV.

6.1 Utilizadores e Tipos

Como a informação dos utilizadores ou dos tipos esta toda na base de dados, apenas se converte ou desfaz a conversão do modelo de dados para um ficheiro CSV. As figuras seguintes correspondem a exemplos de um utilizador e um tipo no formato CSV, respetivamente.

```
[_id,nome,email,[estudante,docente,curso,departamento],dataRegisto,dataUltimoAcesso,password,nivel  
joao123,João Pedro Afonso,jpbp.pfdrp@gmail.com,[true,false;Miei;Informática],2021-01-26T09:22:00.000Z,2021-01-26T09:22:00.000Z,123,0
```

Figura 6: Exemplo de um utilizador em formato CSV.

```
_id,[param]  
js,[numero_linhas#Number;bibliotecas_adicionais#String]
```

Figura 7: Exemplo de um tipo em formato CSV.

6.2 Recursos

Uma vez que os recursos possuem parte da informação na base de dados, correspondente à *metadata*, e outra parte dentro de uma pasta, que são os ficheiros relativos ao recurso, os processos de importação e exportação ocorrem de modo diferente comparativamente ao caso dos utilizadores e dos tipos.

Neste caso, para a *metadata* armazenada na base de dados é criado um ficheiro CSV com a informação, tal como acontece para os dados dos utilizadores e dos tipos. No entanto, os ficheiros armazenados na `fileStore` são transformados num ficheiro ZIP com o nome igual ao campo `_id` do recurso. Após se efetuar este procedimento, estes ficheiros são zipados, formando um ZIP com o ficheiro CSV com toda a *metadata* dos recursos e um ficheiro ZIP por recurso, com os ficheiros de cada recurso.

Na Figura 8 é apresentado o conteúdo da pasta ZIP resultante da exportação de recursos e na Figura 9 encontra-se a informação do ficheiro CSV gerado.

```
export recursos0.5980619374700369 2021-01-27T15 14
└─ 2021-01-27T1514-0.673176074498731.zip
   export:recursos0.5980619374700369:2021-01-27T15:14.csv
```

Figura 8: Exemplo do ficheiro ZIP que constitui um *export* de recursos.

```
id,titulo,subtitulo,dataCriacao,dataRegisto,produtor,visibilidade,[likes],[hashtags],[post]
2021-01-27T1514-0.673176074498731,Lesi exame,,2021-01-27T15:14:00.000Z,2021-01-27T15:08:00.000Z,admin,publico,[],[Algoritmos_e_Complexidade],[]
```

Figura 9: Exemplo do ficheiro CSV de recursos.

7 Rotas

Nesta secção serão enumeradas as diferentes rotas existentes no servidor aplicacional desenvolvido, assim como a explicação da sua funcionalidade.

7.1 Utilizadores - /utilizadores

- **GET /menu** - página inicial da plataforma, na qual aparecem as opções de *login* e de registo.
- **GET /registo** - formulário de registo de um novo utilizador.
- **POST /registo** - cria um novo documento na coleção "utilizadores" da base de dados, a partir da informação passada no *body*.
- **GET /login** - formulário de autenticação de um utilizador, no qual devem ser introduzidas as credenciais.
- **POST /login** - verifica se as credenciais introduzidas correspondem a um utilizador na base de dados e, caso corresponda, realiza a autenticação.
- **GET /logout** - realiza o *logout*, terminando a sessão do utilizador em questão.
- **GET /** - lista de todos os utilizadores da plataforma.
- **GET /:username** - todas as informações sobre um utilizador em específico.
- **GET /delete/:username** - elimina o utilizador especificado, sendo que esta operação só pode ser realizada pelo próprio utilizador ou por um administrador.
- **GET /editar/:username** - formulário para editar informação sobre o utilizador especificado, sendo que esta operação só pode ser realizada pelo próprio utilizador ou por um administrador.
- **POST /editar/:username** - altera as informações editadas e passadas através do *body* no documento da base de dados correspondente ao utilizador.

7.2 Index - /

- **GET /** - página do perfil do utilizador que esteja conectado na plataforma, na qual são apresentadas notícias, alguma informação sobre o mesmo e botões para aceder a informação sobre si ou outros utilizadores, recursos e tipos.

7.3 Recursos - /recursos

- **GET /** - lista de recursos, em que pode se procurar recursos de uma *hashtag* ou o ano específicos ou ordenar consoante o título, os mais recentes, os mais antigos ou os melhor classificados.
- **GET /:id** - todas as informações relativas a um recurso em específico, assim como o conteúdo das pastas.
- **GET /:id/*** - informações de uma pasta dentro de um recurso.
- **GET /novo** - formulário de *upload* de um recurso, sendo que esta operação só pode ser realizada por um produtor ou administrador.
- **POST /novo** - cria um novo documento na coleção "recursos" da base de dados, a partir da informação passada no *body*.
- **GET /estrutura-manifesto** - página com a explicação da estrutura que o manifesto deve possuir.
- **GET /like/:id** - faz um *like* no recurso especificado.
- **GET /like/:id/*** - faz um *like* no recurso especificado, quando está dentro de uma pasta de um recurso.
- **GET /download/:id/*** - *download* de parte de um recurso.
- **GET /download/:id** - *download* do recurso na totalidade.
- **GET /delete/:id** - elimina o recurso especificado, sendo que esta operação só pode ser realizada pelo próprio produtor do recurso ou por um administrador.
- **GET /editar/:id** - formulário para editar informação sobre o recurso especificado, sendo que esta operação só pode ser realizada pelo próprio produtor do recurso ou por um administrador.
- **POST /editar/:id** - altera as informações editadas e passadas através do *body* no documento da base de dados correspondente ao recurso.

7.4 Tipos - /tipos

- **GET /** - lista de tipos de recursos.
- **GET /:_id** - todas as informações sobre um tipo.
- **GET /novo** - formulário de *upload* de um tipo, sendo que esta operação só pode ser realizada por um administrador.
- **POST /novo** - cria um novo documento na coleção "tipos" da base de dados, a partir da informação passada no *body*.
- **POST /delete/:_id** - elimina o tipo especificado, sendo que esta operação só pode ser realizada por um administrador.
- **GET /editar/:_id** - formulário para editar informação sobre o tipo especificado, sendo que esta operação só pode ser realizada por um administrador.
- **POST /editar/:_id** - altera as informações editadas e passadas pelo *body* no documento da base de dados correspondente ao tipo.

7.5 Publicações - /publicacoes

- **POST /:idrecurso/novo** - faz um novo *post* sobre o recurso especificado.
- **GET /:idrecurso/:idpost** - consulta o *post* especificado, assim como os comentários sobre o mesmo.
- **GET /:idrecurso/:idpost/delete** - elimina o *post* especificado, sendo que esta operação só pode ser realizada pelo próprio produtor do recurso ou por um administrador.
- **GET /:idrecurso/:idpost/editar** - formulário para editar dados sobre o recurso especificado, sendo que esta operação só pode ser realizada pelo próprio produtor do recurso ou por um administrador.
- **POST /:idrecurso/:idpost/editar** - altera as informações editadas e passadas através do *body* no documento da base de dados, correspondente ao *post* do recurso.
- **GET /:idrecurso/:idpost/coments** - destinado para o jQuery vir buscar os comentários.

- **POST** `/:idrecurso/:idpost/coments` - altera as informações editadas e passadas através do *body* no documento da base de dados no qual constam os comentários do *post* do recurso.
- **GET** `/:idrecurso` - lista das publicações existentes sobre um recurso.

7.6 Importação - `/import`

- **GET** `/` - página com as opções para fazer importação de recursos, tipos ou utilizadores.
- **GET** `/recursos` - formulário de *upload* do ficheiro de importação de recursos.
- **POST** `/recursos` - faz a importação dos recursos na plataforma.
- **GET** `/tipos` - formulário de *upload* do ficheiro de importação de tipos.
- **POST** `/tipos` - faz a importação dos tipos na plataforma.
- **GET** `/utilizadores` - formulário de *upload* do ficheiro de importação de utilizadores.
- **POST** `/utilizadores` - faz a importação dos utilizadores na plataforma.

7.7 Exportação - `/export`

- **GET** `/` - página com as opções para fazer exportação de recursos, tipos ou utilizadores.
- **GET** `/recursos` - faz a exportação dos recursos.
- **GET** `/tipos` - faz a exportação dos tipos.
- **GET** `/utilizadores` - faz a exportação dos utilizadores.

8 Conclusão

O presente trabalho consistiu no desenvolvimento de uma plataforma de gestão e disponibilização de recursos educativos e com a sua realização foi possível consolidar a aprendizagem na unidade curricular de Desenvolvimento de Aplicações *Web*.

Considera-se que todos os objetivos propostos para o projeto foram cumpridos, no entanto é de salientar alguns aspetos. Para o *unzip* de ficheiros foi utilizada a biblioteca `adm-zip`, que uma vez que traz os ficheiros para memória, apenas consegue lidar com ficheiros ZIP com um tamanho máximo de 2GB. Para o trabalho em questão e para testar com cargas na ordem das dezenas esta ferramenta é suficiente, no entanto com cargas superiores apresenta-se como uma limitação. Deste modo, como trabalhos futuros, uma melhoria que se poderia fazer ao sistema seria substituir esta biblioteca por uma que tenha maior capacidade ou lançar algum processo que execute alguma ferramenta própria de *zip* que lide com ficheiros de maior tamanho, como por exemplo, a ferramenta `zip` que está normalmente instalada nos sistemas Unix.

Em suma, considera-se que a realização deste trabalho prático permitiu desenvolver competências relativamente à ferramenta Express e com Node.js, que certamente serão cruciais nas carreiras profissionais dos elementos do grupo.

9 Referências

- [1] BagIt <https://tools.ietf.org/id/draft-kunze-bagit-16.html>