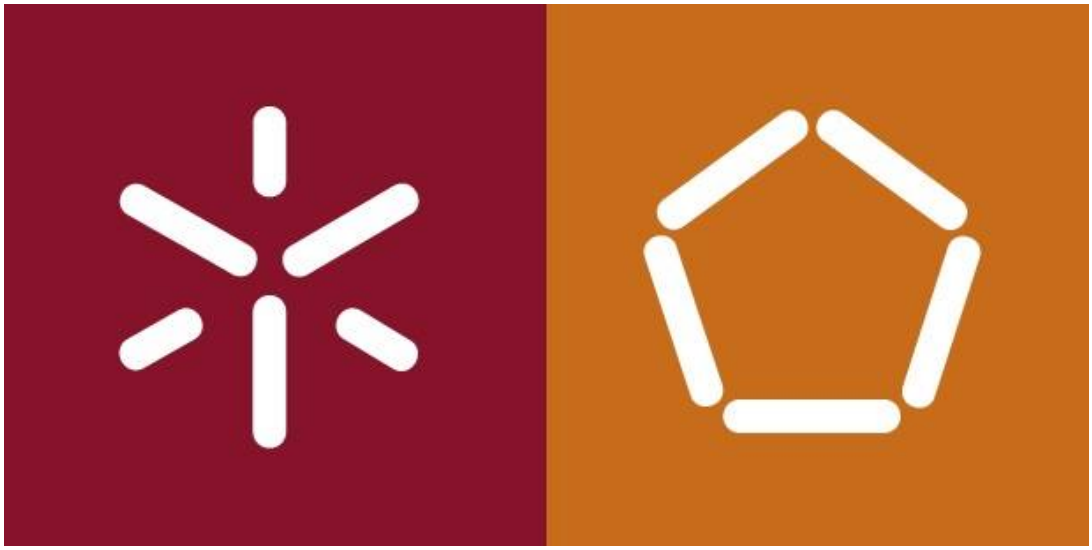
Universidade do Minho



Aplicação MyFacebook

Mestrado Integrado em Engenharia Informática

Desenvolvimento de Aplicações WEB

1º Semestre , 2018/2019

Grupo:
A70132 - Nuno André Lopes Leite
A77377 - Pedro Henrique Moreira Gomes Fernandes
A77689 - Simão Paulo Leal Barbosa

Gualtar,Braga
29 de Janeiro de 2019

Conteúdo

1 Introdução

O desenvolvimento deste trabalho surge no âmbito da unidade curricular de Desenvolvimento de Aplicações WEB, pertencente ao primeiro semestre do 4ºano/1ºano do curso de MIEI/MEI.

O objetivo deste trabalho prático é implementar uma aplicação que sirva um propósito semelhante ao de um diário digital, possuindo para o efeito uma interface pública que pode ser utilizado por qualquer utilizador autenticado. Além disso, foi também criada uma *API* de dados que permite a gestão dos dados inerentes à utilização da aplicação. A aplicação final será também composta por um pequeno módulo de administrador, onde o mesmo pode importar ou exportar os dados da aplicação seguindo um formato que será abordado posteriormente neste relatório, bem como observar algumas estatísticas relevantes e atuais da mesma.

A implementação desta aplicação foi feita recorrendo às seguintes tecnologias:

- *NodeJS/Javascript* - Linguagens de programação que permitem construir o servidor que escuta e soluciona os pedidos;
- *MongoDB* - Motor de base de dados não relacional que serve de suporte à aplicação, persistindo os dados necessários.

Finalmente, é também relevante referir que a aplicação suporta partilhas de publicações com outras redes sociais nomeadamente o *Facebook* e o *Twitter*.

2 Caracterização geral da aplicação

De forma a ter um melhor entendimento do que deveria ser a aplicação, o grupo primeiro tratou de identificar as entidades que deveriam estar definidas na utilização da mesma e, nesse sentido, identificámos as seguintes entidades:

- **Utilizador** - Um utilizador singular que utiliza a aplicação, devidamente autenticado, após se ter registado,sendo que este pode ter o papel de utilizador simples ou de administrador. Deve ser referido que, de início, existe apenas um administrador, o qual pode posteriormente inserir novos;
- **Grupos** - Um grupo é uma entidade composta por vários utilizadores, que permite a partilha de publicações apenas entre esses membros do grupo. O grupo é sempre criado por um utilizador, que passa a ser o administrador do mesmo, podendo depois adicionar e remover membros;
- **Publicações** - Uma publicação é a forma de um utilizador expressar o que quer nesse momento, podendo escolher um de vários tipos de publicação para o fazer. Estas podem ser privadas, ou seja apenas acessíveis ao utilizador que a criou, ou pública, visíveis a qualquer utilizador autenticado que visite a página do utilizador.

Como foi previamente dito, existem vários tipos de publicações predefinidos entre as quais o utilizador pode escolher, que são as seguintes:

- *Genérica* - Composta apenas por um título, uma descrição e ficheiros anexados;
- *Ideia* - Composta por um título, uma explicação da ideia e classificadores de ideia;
- *Receita* - Composta por um nome de cozinhado, os ingredientes e as instruções para fazer o mesmo;
- *Evento* - Composta por um título, a data e o local que o evento se realiza,os convidados e uma descrição;
- *Evento Profissional* - Muito similar à publicação de um evento, mas que permite a adição de ficheiros à mesma;
- *Formação* - Composta por um título, a creditação que foi obtida, a instituição que forneceu a creditação e uma descrição;
- *Desportiva* - Composta por um título, a atividade desportiva que foi realizada, a duração da mesma, uma descrição, fotos referentes á atividade e um ficheiro *GPX (GPS Exchange Format)*;
- *Álbum* - Composta por título, uma descrição e várias fotos, em que para cada uma é dado o local e a data em que a mesma foi tirada.

Além disso, cada publicação guarda classificadores de conteúdo que são escolhidos pelo utilizador, bem como a indicação se a mesma é privada ou pública, entre outros aspetos que serão abordados na secção de Modelos de dados.

3 Modelos de dados

Como foi previamente mencionado, o grupo identificou 3 entidades de dados que devem ser devidamente modeladas na utilização desta aplicação:

- Os utilizadores, que interagem com a aplicação;
- As publicações, feitas pelos utilizadores num contexto apenas seu ou num contexto de um grupo;
- Os grupos, que permitem a partilha de publicações entre os seus membros (utilizadores)

Neste sentido, um utilizador foi modelado da seguinte forma:

```
{
    "email": {type:String, required:true, unique: true},
    "profissao" : String,
    "instituicao" : String,
    "nome": {type:String, required:true},
    "password": {type:String, required:true},
    "foto": String,
    "morada": String,
    "role": {type:String, enum: ['admin','user'], required:true},
    "classificadores": [String],
    "favoritos": [{
        email: {type:String, required:true},
        nome: {type:String,required:true}
    }]
    "dataAniversario": String
}
```

Listing 1: Modelo de utilizador

Através de uma breve observação, é possível verificar que o modelo de um utilizador da base de dados, mesmo na sua forma mais simplista, terá que garantir:

- O *email* do utilizador é obrigatório e único;
- O *nome* do utilizador é obrigatório;
- A *password* do utilizador é obrigatória;
- O *role* do utilizador é obrigatório e tem que ser ou **user** ou **admin**;
- No caso de o utilizador possuir *favoritos*, cada um deles deve ser identificado pelo seu email e nome obrigatoriamente.

De seguida, apresentámos e explicitámos a forma como modelámos uma publicação:

```
{
    "origin_email": {type:String, required:true},
    "tipo" : {type:String, required:true},
    "data" : {type:String, required:true},
    "dados": PubSchema,
    "classificacoes": [String],
    "comentarios": [{
        "origin_email": String,
        "comentario": String
    }],
    "groupId": String,
    "isPrivate": {type:Boolean, required:true, default: true}
}
```

Listing 2: Modelo de Publicação

Através de uma breve observação do modelo de publicação, podemos imediatamente denotar que uma publicação tem que obrigatoriamente conter um **origin_email**, referente ao utilizador que efetuou a publicação, um **tipo** de publicação, uma **data** indicativa de quando a publicação foi feita no formato *dd-mm-aaaa hh:mm* e um *booleano* indicando se a publicação é ou não privada, sendo que, por defeito ela é.

Além disso, os **dados** da publicação tem o tipo de **PubSchema**, que é um modelo que instancia apenas num tipo de publicação, contendo os dados necessários para a mesma. Os tipos de publicação e os dados necessários para que a mesma esteja correta já foram previamente discutidos na secção 2. As **classificacoes** referem-se aos classificadores granulares de conteúdo de uma publicação (*hashtags*).

Tendo já apresentando a modelação de dados para os utilizadores e para as publicações, é necessário terminar a secção, apresentando e explicitando a modelação de dados feita para o grupo:

```
{
    "nome": {type:String, required:true},
    "descricao" : {type:String, required:true},
    "fotoGrupo" : {type:String, required:true},
    "membros": [String],
    "admin": {type:String, required:true}
}
```

Listing 3: Modelo de Grupo

Após uma breve observação do modelo de dados para um grupo é possível observar que é obrigatório definir um grupo que tenha um **nome**, uma **descricao**, uma **fotoGrupo** e a

indicação do **admin** (criador) do grupo. Os membros do grupo serão sempre identificados pelo seu email que é utilizado na aplicação.

É necessário também referir que esta é a especificação feita pelo grupo e, na altura da criação/instanciação de qualquer um destes modelos, ao mesmo será acrescentado um **_id** unívoco que os permita identificar, apesar de que, no caso dos utilizadores, isso nunca será necessário.

4 API de dados

Nesta secção, iremos apresentar as rotas que foram definidas na API de dados para cada uma das entidades, explicitando o método de pedido que é (*POST*,*DELETE*,*GET* ou *PUT*).

A forma de apresentação dos pedidos aos quais a API responde será feita da seguinte forma:

```
MÉTODO.(rota) - descrição
```

É também necessário ter em atenção que as rotas apresentadas serão apresentadas a partir de:

- **'/api/users'** para os utilizadores;
- **'/api/pubs'** para as publicações;
- **'/api/groups'** para os grupos.

4.1 Users

A não ser que algo seja dito em contrário, todas as rotas abaixo referidas requerem autenticação. Além disso, todas as rotas que contenham o sub-caminho '/admin' requerem autenticação de administrador e não apenas de utilizador normal, exceto a rota '/admin/login'.

É também necessário referir que todas as rotas recebem um *access_token*, que é a forma de autenticação, seja através da query string ou através do body.

- **GET.('/')** - rota que deve receber na query string ou o parâmetro *email* ou o parâmetro *nome*. Também é possível não passar nenhum parâmetro (com excepção do *access_token* que tem que estar sempre presente), que irá consultar todos os utilizadores mas essa operação requer privilégios de utilizador administrador;
- **POST.('/')** - rota que recebe os parâmetros de inicialização necessários para criar um utilizador normal no *body* e o insere na base de dados;
- **PUT.('/')** - rota que recebe os parâmetros que devem ser atualizados no *body* e executa a atualização na base de dados. Esta atualização pode ser respeitante a dados de utilizador ou à sua password;
- **POST.('/login)** - rota que recebe os parâmetros de autenticação de utilizador normal no *body* e tenta fazer o *login* do utilizador. Esta rota não requer autenticação;
- **GET.('/favorites')** - rota que recebe um parâmetro da query string, *emailFav* e, que procura procura se o utilizador com esse email consta nos favoritos do utilizador que está autenticado;
- **POST.('/favorites')** - rota que recebe dois parâmetros na query string (além do access_token), *favoriteEmail* e *favoriteNome*, que correspondem ao email e nome do favorito a adicionar nos favoritos do utilizador autenticado;
- **DELETE.('/favorites')** - rota que recebe dois parâmetros na query string, *favoriteEmail* e *favoriteNome* e remove o utilizador com estes dados dos favoritos do utilizador autenticado;

- **POST.('/search')** - rota que recebe um campo de procura pelo *body* e executa a procura por nome e, caso não encontre nenhum utilizador, executa a procura por email. No final, devolve o resultado quer tenha encontrado ou não utilizadores com o campo de procura especificado;
- **POST.('/admin')** - rota que recebe os parâmetros de inicialização necessários para criar um utilizador administrador no *body* e o insere na base de dados. Esta rota é autenticada e o utilizador que faz o pedido tem que ser administrador;
- **PUT.('/admin')** - rota que recebe os parâmetros de alteração de password do administrador no *body* e executa a mesma;
- **GET.('/count')** - rota que, quando chamada, conta todos utilizadores existentes na base de dados e devolve essa informação. A execução desta rota requer privilégios de administrador;
- **POST.('/admin/login')** - rota que recebe os parâmetros de autenticação do administrador e verifica se é um administrador válido para que o possa autenticar. Esta rota não requer autenticação;
- **POST.('admin/import')** - rota que recebe um array de documentos para inserir na base de dados e executa a operação referida;
- **GET.('/admin/listExports')** - rota que calcula e devolve os ficheiros correspondentes aos *exports* feitos pelo administrador autenticado

4.2 Pubs

A não ser que algo seja dito em contrário, todas as rotas abaixo referidas requerem autenticação.

É também necessário referir que todas as rotas recebem um *access_token*, que é a forma de autenticação, seja através da query string ou através do body.

- **GET.('/')** - rota que devolve todas as publicações registadas na base de dados. Requer privilégios de administrador;
- **POST.('/')** - rota que insere uma publicação recebida no *body*, na base de dados;
- **GET.('/:id_pub')** - rota que recebe um identificador de publicação como parâmetro da rota e devolve essa rota, caso exista;
- **PUT.('/:id_pub')** - rota que recebe os campos de atualização no *body* e um identificador de publicação como parâmetro da rota e atualiza os campos da publicação com esse identificador;
- **DELETE.('/:id_pub')** - rota que recebe um identificador de publicação como parâmetro da rota e elimina essa publicação da base de dados;
- **POST.('/newComment')** - rota que recebe um comentário e o identificador da publicação no *body* e insere o comentário recebido nos comentários da publicação com o identificador recebido;

- **GET.('/:email/filter')** - rota recebe um email de utilizador como parâmetro da rota e seleciona as publicações de acordo com os filtros recebidos do utilizador com o email recebido como parâmetro. Também é tido em conta se o utilizador autenticado é o mesmo que o utilizador com o email recebido;
- **GET.('/fromUser')** - rota que recebe o email do utilizador do qual deve obter as publicações através de um parâmetro da query string (email). Devolve as publicações desse utilizador tendo em conta se o utilizador autenticado é o mesmo que o email que vem na query string ou não (devolve privadas e públicas ou só públicas, respetivamente);
- **GET.('fromGroup')** - rota que recebe como parâmetro da query string o identificador de um grupo (group_id) e devolve todas as publicações relacionadas com esse grupo;
- **POST.('/admin/import')** - rota de administrador que recebe um array de publicações para inserir na base de dados e executa essa operação. Necessita de privilégios de administrador;
- **GET.('/count')** - rota que recebe um tipo de publicação como parâmetro da query string (tipo) e devolve o número de publicações desse tipo. Se não for passado nenhum tipo, calcula o número total de publicações. Requer privilégios de administrador.

4.3 Groups

A não ser que algo seja dito em contrário, todas as rotas abaixo referidas requerem autenticação.

É também necessário referir que todas as rotas recebem um *access_token*, que é a forma de autenticação, seja através da query string ou através do body.

- **POST.('/')** - rota que recebe no *body* do pedido as informações de um grupo criado e executa essa operação na base de dados;
- **DELETE.('/')** - rota que recebe o identificador do grupo como parâmetro da query string (group_id) e elimina-o da base de dados;
- **PUT.('/')** - rota que recebe no *body* do pedido os valores de atualização do grupo e atualiza-o na base de dados;
- **POST.('/member')** - rota que recebe no *body* do pedido o email do membro a adicionar e o identificador de grupo e adiciona esse membro à lista de membros do grupo identificado pelo identificador recebido na base de dados;
- **DELETE.('/member')** - rota que recebe o identificador do grupo e o email do membro a remover como parâmetros da query string (group_id e email) e remove-o da lista de membros do grupo na base de dados;
- **GET.('/:group_id')** - rota que recebe como parâmetro da rota o identificador do grupo, consulta a base de dados e devolve-o caso exista;
- **GET.('/withUser')** - rota que recebe como parâmetro da query string o email do utilizador do qual pretendemos obter os grupos dos quais ele é membro e devolve os referidos grupos;

- **GET.('/')** - rota que devolve todos os grupos existentes na base de dados. Requer privilégios de administrador;
- **POST.('/admin/import')** - rota que recebe no *body* do pedido um array de grupos a inserir na base de dados e executa essa operação. Requer privilégios de administrador;
- **GET.('/count')** - rota que executa uma operação na base de dados apra contar o número de grupos existentes e devolve o referido número. Requer privilégios de administrador.

5 Frontend

Nesta secção, pretendemos demonstrar, sustentando a nossa explicação com imagens, o que a aplicação construída permite fazer de uma forma muito sucinta.

A figura 1 mostra um excerto da página inicial da aplicação, onde além da informação inicial é possível visualizar as *tabs* de registo e login, onde, como o nome indica, um utilizador se pode registar na aplicação e autenticar-se na mesma.

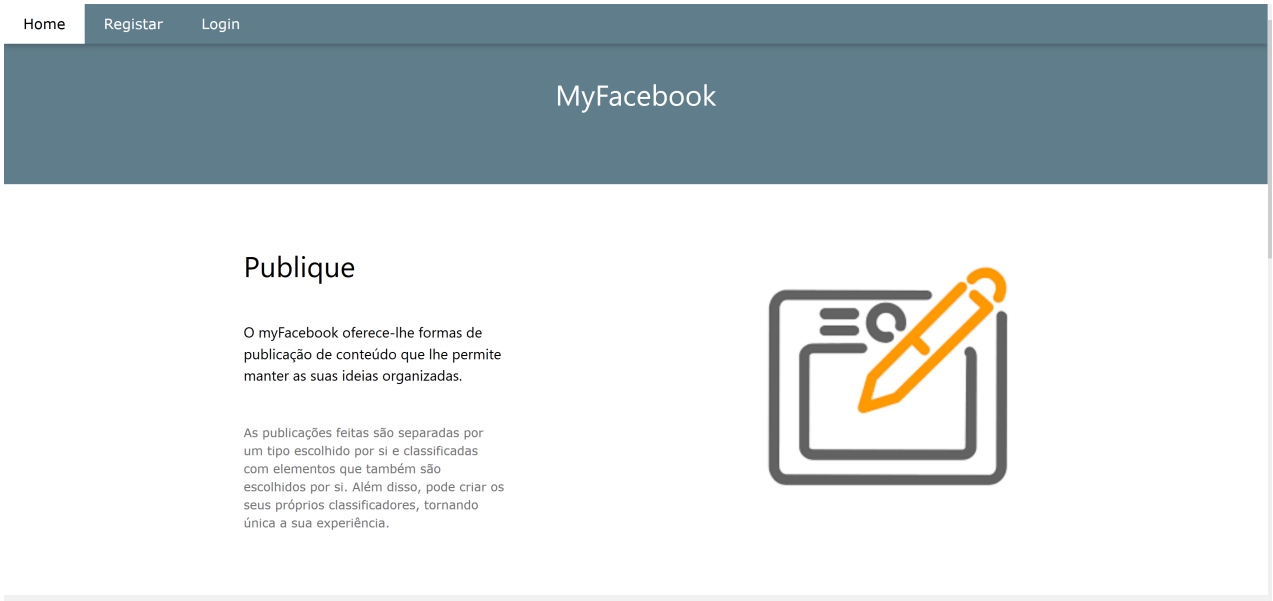


Figura 1: Excerto da página inicial da aplicação

A figura 2 mostra a *homepage* de um utilizador onde, como se pode ver, à esquerda aparece a informação básica do utilizador bem como os grupos nos quais ele está inserido. No centro, aparece a barra de pesquisa, que permite procurar utilizadores por nome ou email e vemos também a área de publicações, onde podemos inserir uma publicação (escolhendo o seu tipo) , abrir a área de filtragem para filtrar as publicações que queremos ver e a própria área de visualização de publicações.

A figura 3 mostra um exemplo de uma publicação, neste caso, desportiva, onde podemos visualizar os seus elementos e onde temos a opção de partilhar nas redes sociais a publicação e editá-la. Também temos a opção de ver os comentários associados àquela publicação, como mostra a figura 4. À direita, temos uma pequena área de favoritos, visto que, após visitar a página de um outro utilizador e o adicionar aos seus favoritos, o mesmo aparecerá nessa área para que não seja necessário fazer uma pesquisa constante.

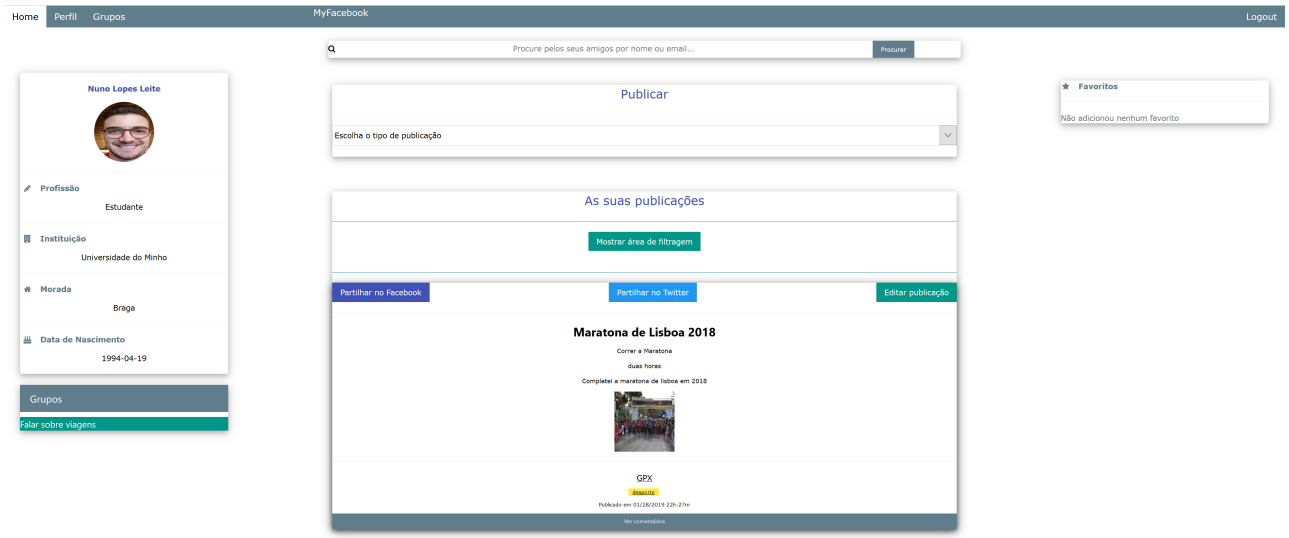


Figura 2: Homepage de um utilizador

Figura 3: Exemplo de uma publicação desportiva

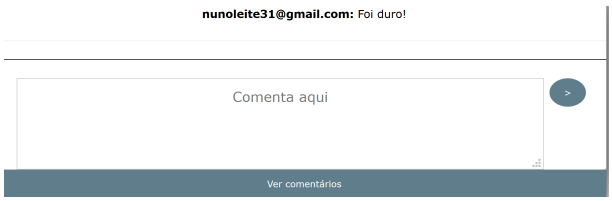


Figura 4: Exemplo de uma secção de comentários

A figura 5 mostra a página de um utilizador, quando essa utilizador não é o utilizador autenticado, mostrando-lhe a informação do utilizador, bem como as publicações (públicas) dele e um botão que permite adicionar ou remover o favorito.

Figura 5: Exemplo de visualização de página inicial de outro utilizador

A tab *Perfil* permite aceder a um formulário de atualização dos dados de utilizador, inclusive a password.

De seguida, é conveniente apresentar a página que pode ser vista na figura 6, que apresenta a possibilidade de um utilizador criar um grupo, além de mostrar todos os grupos nos quais o mesmo está inserido. A partir de um clique no grupo, o utilizador pode aceder à página do grupo. Um exemplo de uma página de grupo pode também ser vista na figura 7, na qual o administrador é que está a aceder.

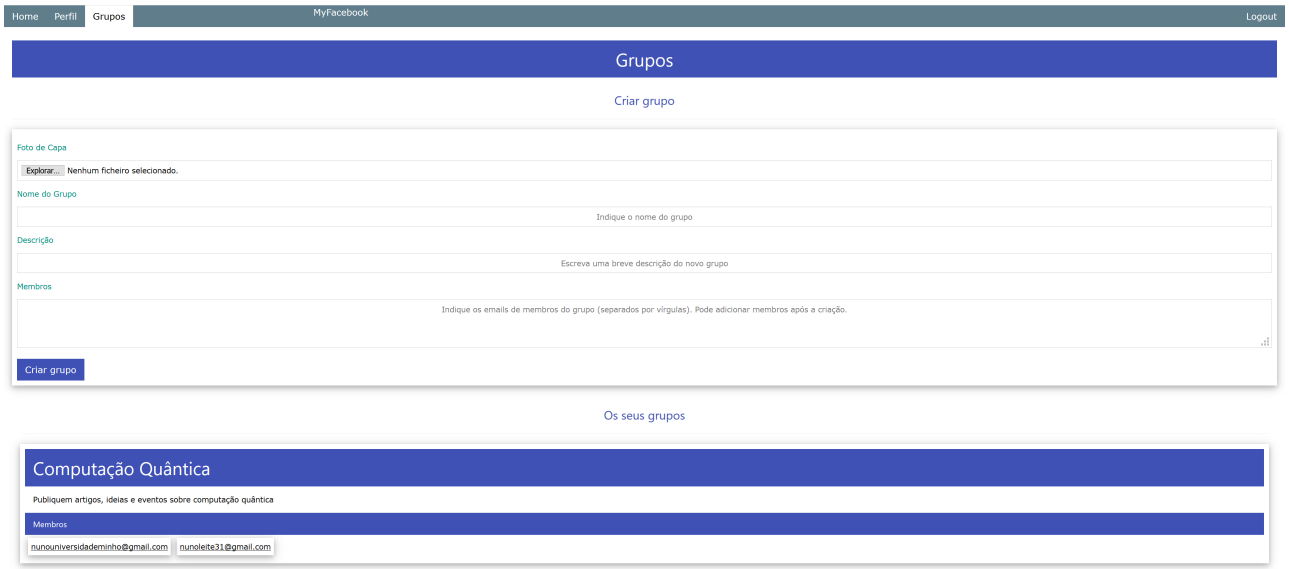


Figura 6: Exemplo de página de grupos de um utilizador

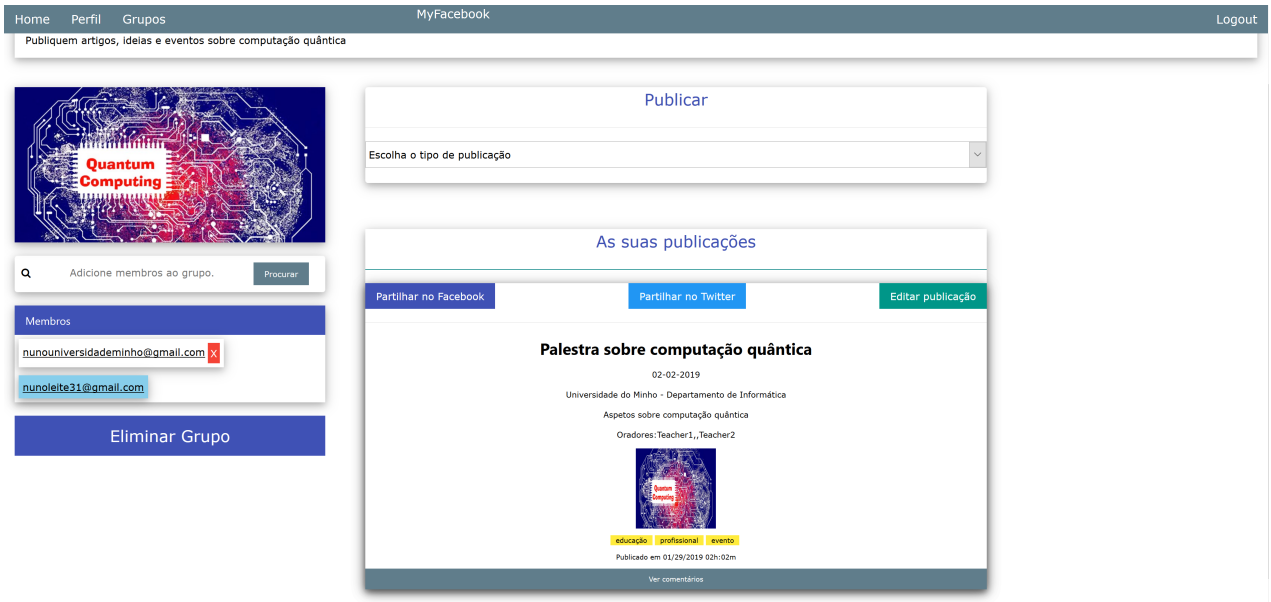


Figura 7: Exemplo de visualização de página inicial de um grupo

Terminámos assim uma breve apresentação das possibilidades fornecidas pela publicação, permitindo ter uma ideia de como a aplicação está construída, em termos da parte gráfica, aquela que é visível ao utilizador.

6 Aspetos específicos da aplicação desenvolvida

6.1 Autenticação

A autenticação implementada pelo grupo nesta aplicação foi totalmente baseada em *json web tokens*. A cada utilizador é fornecido um token de acesso onde está guardado o seu email, o seu *role* na aplicação e o seu nome.

O token é assinado com um dado segredo e umas dadas opções e verificado da mesma forma.

6.2 Formato de importação/exportação de dados

De forma a possibilitar uma exportação e importação de dados, que se torne necessária em qualquer altura por, por parte de um administrador, o grupo acrescentou, na página de administrador, 2 formulários. Em cada uma das situações, é dado a escolher ao administrador o tipo de dados que quer exportar ou importar. Esses tipos de dados podem ser:

- Apenas utilizadores;
- Apenas publicações;
- Apenas grupos;
- Todos

Os dados serão carregados (ao importar) ou guardados (ao exportar) de/num ficheiro *json*, que terá o seguinte formato:

- No caso de o tipo de dados ser apenas utilizadores, apenas publicações ou apenas grupos, o ficheiro será composto por um array de objetos, onde cada objeto representa um utilizador/publicação/grupo tal e qual como ele estava persistido na base de dados (listing 4);
- No caso de o tipo de dados corresponder a *Todos*, o ficheiro será composto por um objeto que contém 3 pares chave-valor. As chaves serão *"users"*, *"pubs"* e *"groups"* e os valores serão os respetivos arrays de objetos tal e qual como estão persistidos na base de dados (listing 5).

```
1 [
2     {User/Pub/Group1},
3     {User/Pub/Group2},
4     ...
5 ]
```

Listing 4: Modelo de ficheiro de dados de uma entidade

Além disso, para facilitar a tarefa de alocação dos dados, na página do administrador, após exportar, é-lhe mostrada uma lista com todos os *exports* que por ele já foram feitos, em forma de hiperligação, que lhe permite fazer o *download* do ficheiro, que pode ser, mais tarde, utilizado para importar novamente os dados.

```
{
    "users": [Users],
    "pubs": [Pubs],
    "groups": [Groups]
}
```

Listing 5: Modelo de ficheiro de dados de todas as entidades

6.3 Partilha de publicações em redes sociais

Foi implementada a partilha de publicações no *Facebook* e no *Twitter*.

De forma a ser possível partilhar publicações pelo `Twitter` são utilizados *links* semelhantes ao seguinte, `https://twitter.com/intent/tweet?text=`, onde associado ao parâmetro *text* é introduzido o texto das publicações. No caso das publicações conterem ficheiros/imagens, o processo é o mesmo. No entanto, as ligações para estes são *links* que serão tratados como texto, apesar de existir o parâmetro *url*, devido ao facto de os nossos *links* não serem reconhecidos por pertencerem a uma aplicação hospedada localmente.

No caso da partilha de publicações pelo *Facebook* foram encontrados alguns problemas relacionados com o facto da aplicação ser hospedada localmente. Não é possível utilizar a *API* do *Facebook* com uma aplicação local e por isso a única opção foi utilizar *links* como o seguinte: `https://www.facebook.com/sharer/sharer.php?quote= ... &u= ....` Esta é uma forma para partilhar um *link* no *Facebook* em que o parâmetro *quote* corresponde ao texto e o parâmetro *u* corresponde ao *url* do *link* que se pretende partilhar. Tendo em conta que os nossos *links* não são reconhecidos, e como o parâmetro *u* é obrigatório, foi utilizado o *url* do site do *Google* para contornar este problema.

6.4 Instalação no servidor e criação de Administrador

O primeiro passo a ser feito para instalar a aplicação num servidor (mesmo que seja na máquina local) é executar o comando:

```
npm install
```

Caso ocorra um erro com a biblioteca *bcrypt*, que se deve ao facto de a última versão ter problemas em determinados contextos, deve-se de seguida executar o comando:

```
npm i bcrypt@3.0.2
```

Depois destes 2 passos, todos os módulos estarão instalados.

Finalmente, é necessário criar o administrador da aplicação e, tendo em conta que ao instalar no servidor temos acesso ao motor de base de dados, deve-se aceder à página de registo usual e criar um utilizador que será o administrador após completar os seguintes passos:

- Após o registo de utilizador e considerando que esse utilizador criado tem o email com o valor **administrador@myfacebook.com**, é necessário executar os seguintes comandos:

```
mongo
use myFacebook
db.users.updateOne({email: "administrador@myfacebook.com"},
                   {$set :{role: "admin"}})
```

- Para terminar, é necessário apenas aceder à diretoria *public/uploaded/administrador@myfacebook.com* e criar as diretorias *exports* e *imports*.

Terminados estes passos, a aplicação está pronta a funcionar já com um administrador ativo. Relembrámos que o login de utilizador é feito a partir de *'/users/admin/login'*.

A aplicação deve ser iniciada com o seguinte comando:

```
nodemon --ignore sessions --ignore public/uploaded
```

Isto deve ser feito para que o servidor não reinicie quando existem alterações ao armazenar os ficheiros dos utilizadores.

7 Conclusão

Com este trabalho foi possível pôr em utilização várias das ferramentas e conceitos abordados ao longo do semestre, assim como ir além da complexidade das pequenas aplicações desenvolvidas nas aulas e nos trabalhos para casa, o que permitiu desenvolver habilidades relacionadas não só com o desenvolvimento de interfaces *web*, mas também interfaces RESTful de dados e, genericamente, programação *web* com JavaScript, no lado do cliente - com a injeção de HTML, tratamento de eventos de interação com elementos da página (como botões) e envio de dados em formulários ou JSON para o servidor, por exemplo - e no lado do servidor - com a criação de *handlers* para os diversos URLs de recursos e métodos HTTP suportados pelo servidor, para além da necessária interação com uma camada de dados, suportada por uma API e pelo Mongoose e MongoDB.

Por fim, consideramos ter atingido todos os objetivos propostos para este projeto e adicionalmente algumas funcionalidades que se assemelham às que seriam implementadas num projeto semelhante para uso público. Tal como em todos os projetos, as limitações de tempo impediram que fossem melhorados vários aspetos e integradas novas ideias e funcionalidades, assim como limitaram a maturidade da implementação de algumas dessas, mas, em suma e novamente, consideramos ter cumprido os requisitos propostos.