

Universidade do Minho

Mestrado Integrado em Engenharia Informática

Desenvolvimento de Aplicações WEB

myFacebook



Ana Rodrigues
(a78763)



Armando Santos
(a77628)



Filipe Nunes
(a78074)

29 de Janeiro de 2019

Conteúdo

1	Introdução	1
2	Análise e Planeamento	2
2.1	Descrição do problema	2
2.2	Planeamento dos recursos	2
2.3	Planeamento da API REST	3
3	Desenvolvimento da aplicação	4
3.1	Estrutura da diretoria	4
3.1.1	<i>Models</i>	4
3.1.2	<i>Controllers</i>	4
3.1.3	<i>Routes</i>	5
3.2	Base de dados	5
3.2.1	Importação e Exportação dos Dados	5
3.3	Autenticação	5
3.4	Interface com o utilizador	5
3.4.1	Página inicial	6
3.4.2	<i>Feed</i>	7
3.4.2.1	Alteração da informação do utilizador	8
3.4.2.2	Fazer e editar publicações	9
3.4.3	Menu	13
3.4.4	<i>Feed</i> de outro utilizador	15
4	Conclusão	16
A	Documentação da API RESTful	17
A.1	Autenticação	17
A.2	Utilizadores	17
A.3	Items	18

Resumo

Este documento diz respeito ao projeto desenvolvido na unidade curricular de Desenvolvimento de Aplicações WEB da Universidade do Minho.

O objetivo deste projeto consiste no desenvolvimento de uma aplicação *WEB* que implemente um repositório digital de registo de ideias, fotos e outros objetos que se pretenda partilhar ou simplesmente armazenar digitalmente, para poderem ser consultados mais tarde.

1 Introdução

Este trabalho prático tem como objetivo desenvolver uma aplicação *WEB* onde o utilizador poderá publicar uma série de notas, visitar o perfil de outros utilizadores, ver as publicações destes, comentar ou deixar o seu “like” nelas, entre outras funcionalidades. Cada perfil de utilizador funcionará como uma espécie de diário, onde todas as publicações feitas pelos utilizadores seguem uma ordem temporal, do mais recente para o mais antigo.

As ferramentas utilizadas para a realização deste projeto foram o *Express*, para a construção de toda a estrutura e funcionamento da aplicação, utilizando a linguagem *JavaScript*, o *MongoDB*, para a persistência de dados, e o *PUG*, para a criação da interface.

2 Análise e Planeamento

Para estruturar a aplicação foi necessário estudar os diferentes requisitos desta, identificar os diversos tipos de recursos existentes e delinear o conjunto de *endpoints* relativos à interface REST da API de dados.

2.1 Descrição do problema

Como já foi mencionado, pretende-se desenvolver uma plataforma, espécie de diário, onde vários utilizadores possam interagir, denominada *myFacebook*.

Ao aceder à aplicação, terá de ser apresentado ao utilizador uma página para este poder fazer o *login*/registo. Após este estar autenticado, ser-lhe-á apresentada a sua página de utilizador, onde estarão todas as informações do mesmo e todas as suas publicações. Para além disso, este poderá editar os seus dados pessoais e realizar novas publicações nessa mesma página. O utilizador terá a capacidade de criar vários tipos de publicações, desde eventos a simples frases.

Em toda a aplicação, exceto na página de *login*/registo, o utilizador poderá procurar por outros utilizadores, ver os eventos e os utilizadores existentes, ir para a sua página de perfil e efetuar *logout*, através do menu existente, no topo do *website*.

Um utilizador ao aceder à página de outro poderá ver os seus dados pessoais e as publicações públicas deste, podendo comentá-las e reagir a estas através de um botão de “gosto”.

2.2 Planeamento dos recursos

Após terem sido definidos quais os recursos da aplicação, foi necessário definir o modelo destes e os tipos de dados que serão armazenados na base de dados, estando estes ilustrados nas tabelas 1 e 2.

Tabela 1: Atributos da entidade Utilizador.

Entidade	Atributo	Tipo	Obrigatoriedade
Utilizador	Email	String	Sim
	Password		Sim
	Nome		Sim
	Data de nascimento		Sim
	Morada		Não
	Sexo		Sim
	Foto		Não

Tabela 2: Atributos da entidade Item.

Entidade	Atributo	Tipo	Obrigatoriedade
Item	Id utilizador	ObjectId	Sim
	Título	String	Sim
	Data	String	Não
	Tipo	Álbum, Ideia ou Evento	Sim
	Local	String	Não
	Privacidade	Boolean	Não
	Gostos	Number	Não
	Comentários	[Comentário]	Não
	Descritores	[String]	Não
	Descrição	String	Não

Observando a tabela de atributos do Item, pode-se constatar que existem alguns tipos de atributos não primitivos. Estes são definidos da seguinte maneira:

- Álbum:
 - Uma ou mais fotografias (obrigatório);
- Ideia:
 - String;
- Evento:
 - Data de início (obrigatório);
 - Data de fim (obrigatório);
 - Lista de participantes.
- Comentário:
 - Id utilizador;
 - Descrição;
 - Data;
 - Gostos.

2.3 Planeamento da API REST

Inicialmente, foi também importante definir a estrutura da interface de dados, de modo a permitir uma melhor organização e facilitar o desenvolvimento do projeto.

No apêndice A poderá ser encontrada a API de dados seguindo o padrão REST.

3 Desenvolvimento da aplicação

Tendo o plano e estrutura delineados, procedeu-se à implementação da aplicação.

Tal como foi referido em 1, foi utilizada a framework *Express*, que permite desenvolver aplicações *WEB* com bastante facilidade, seguindo o padrão de desenho MVC. No que diz respeito à camada de dados da nossa aplicação, optamos por utilizar *MongoDB*, uma base de dados *NoSQL* que, devido à sua flexibilidade, suporte de bibliotecas e por fazer uso de *JavaScript*, tornou-se bastante fácil e prático de ser integrada no nosso projeto.

3.1 Estrutura da diretoria

O padrão *Model-View-Controller* permite a reutilização de código e a separação de conceitos. Desta forma, este padrão arquitetural de *software* separa a representação da informação da interação do utilizador com o sistema. O padrão é constituído pelos *models*, que consistem nos dados propriamente ditos da aplicação, pelas *views*, que são a representação dos dados, e pelos *controllers* que permitem efetuar comandos a *models* ou *views*. Desta forma, a diretoria do projeto segue a seguinte estrutura:

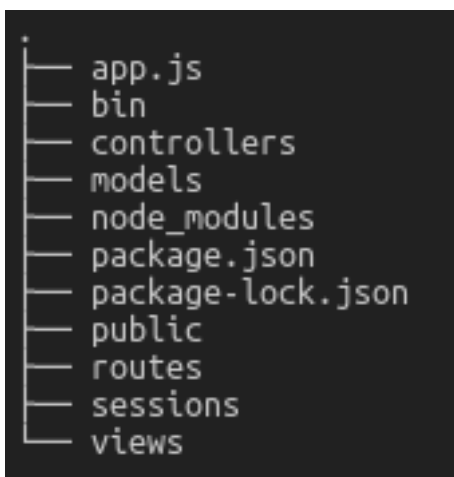


Figura 1: Estrutura da diretoria (1 Nível)

3.1.1 Models

Nesta pasta são definidos os modelos, usando *Mongoose*. Estes modelos são a tradução direta das tabelas apresentadas em 2.2 e serão utilizados pelos *controllers*.

3.1.2 Controllers

Nesta pasta encontram-se definidos os controladores que irão, efetivamente, responder aos pedidos da aplicação.

Após a construção dos *models*, é necessário construir as funções que nos permitem manipular os dados, a partir de *queries* em *MongoDB*, de modo a garantir que todas as funcionalidades propostas para a aplicação funcionem.

Para o efeito, consideramos, por exemplo, as funções de listagem de todos os utilizadores/items, de inserção e atualização de um utilizador/item.

3.1.3 Routes

Nesta pasta encontram-se definidos dois tipos distintos de rotas, as da API de dados, em formato *REST*, que disponibilizam as operações *CRUD* sobre os recursos e a informação em *JSON*, e as rotas da interface, que tiram partido da API para obterem os dados que precisam para os mostrar ao utilizador.

Na secção de apêndice A, no final do presente relatório, seguem todos os *endpoints* existentes para o *backend* e *frontend* da aplicação.

3.2 Base de dados

Como já possuíamos uma ideia bastante sólida dos recursos a persistir e os seus modelos, foi bastante simples codificá-los na nossa aplicação. Utilizamos a biblioteca *Mongoose* para tratar da conexão com a base de dados e para abstrair toda a implementação da sua estrutura através dos *Models*.

3.2.1 Importação e Exportação dos Dados

De modo a facilitar o povoamento da base de dados e garantir a recuperação dos dados, caso esta seja apagada por qualquer motivo, foram definidas duas funções na aplicação que fazem importação dos dados de ficheiros *JSON* para o *MongoDB* e exportação dos dados no sentido contrário. A primeira operação é realizada apenas uma vez, quando o servidor é iniciado, e a segunda é efetuada várias vezes enquanto o servidor se encontra ativo.

3.3 Autenticação

Para aceder à aplicação, o utilizador terá de estar autenticado, tendo, portanto, de realizar o *login*, com o seu *email* e *password*, ou de se registar neste caso ainda não esteja.

Ao realizar um registo, o utilizador terá de preencher os seus dados e estes serão guardados na base de dados. Contudo, a palavra-passe do utilizador tem de ser encriptada de maneira a tornar os dados mais seguros, tendo sido utilizada, para essa finalidade, a biblioteca *bcrypt*.

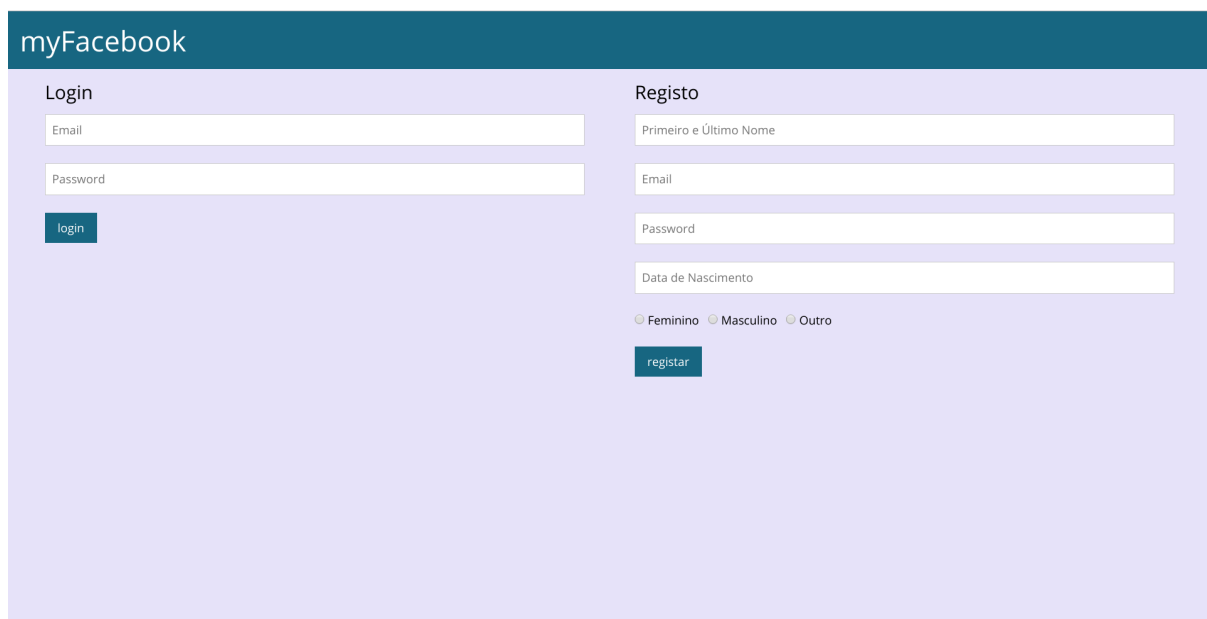
3.4 Interface com o utilizador

Após a realização de todos os passos descritos anteriormente, procedeu-se à criação da interface *WEB*, através dos ficheiros *pug* das *views*, cujos componentes estão descritos nas subsecções abaixo.

3.4.1 Página inicial

A página inicial contém os campos onde um utilizador se poderá registar e efetuar o *login*, caso já esteja registado. É a única página a que um utilizador não autenticado consegue aceder.

Uma vez na página inicial, o utilizador poderá preencher os campos necessários para efetuar o seu registo na aplicação. Todavia, também poderá preencher os dados para o seu *login*, caso já tenha efetuado previamente o registo.



The image shows a web interface for 'myFacebook' with a dark blue header. Below the header, the page is divided into two main sections: 'Login' on the left and 'Registo' on the right, both with light purple backgrounds. The 'Login' section contains two text input fields labeled 'Email' and 'Password', and a dark blue button labeled 'login'. The 'Registo' section contains four text input fields labeled 'Primeiro e Último Nome', 'Email', 'Password', and 'Data de Nascimento'. Below these fields are three radio buttons labeled 'Feminino', 'Masculino', and 'Outro', and a dark blue button labeled 'registar'.

Figura 2: Página inicial.

3.4.2 Feed

Após efetuado o registo/*login* na aplicação, o utilizador será redirecionado para o seu *feed*. É neste que um utilizador poderá alterar os seus dados, publicar e editar as suas ideias, os seus eventos e álbuns e comentar todos eles. Para além disso, este pode também visualizar comentários que outros utilizadores tenham feito nas suas publicações.

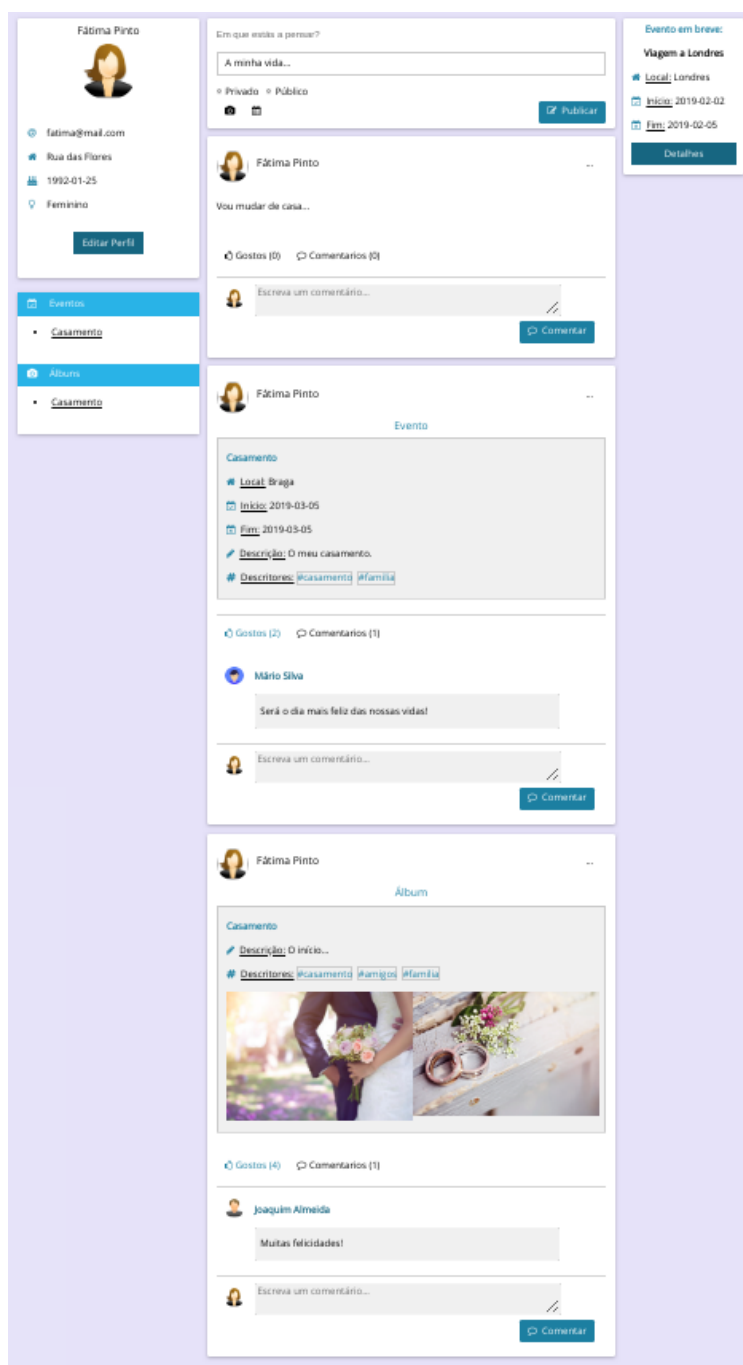


Figura 3: *Feed* de um utilizador.

3.4.2.1 Alteração da informação do utilizador

O utilizador, caso pretenda, poderá alterar os seus dados. Para isso, é-lhe apresentada uma nova página, com os seus dados atuais e editar aqueles que pretende.

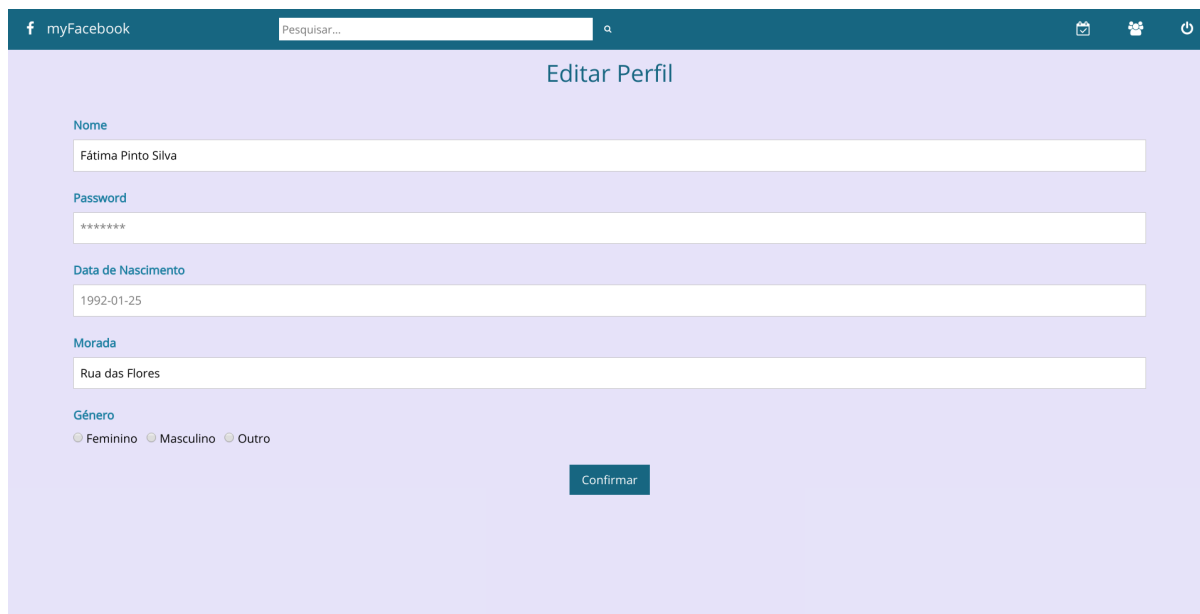


Figura 4: Página para edição da informação do utilizador.

Da mesma forma, o utilizador também poderá alterar a sua fotografia de perfil. Para isso, basta clicar na imagem atual, que se encontra do lado esquerdo, e carregar o novo ficheiro que pretende que seja a sua nova fotografia de perfil.




Figura 5: Mudar a fotografia do utilizador.

3.4.2.2 Fazer e editar publicações

Como mencionado anteriormente, o utilizador poderá publicar ideias, eventos e álbuns de fotografias no seu *feed*. Para isso, apenas terá de seleccionar qual das publicações pretende, sendo a ideia o pré-definido.

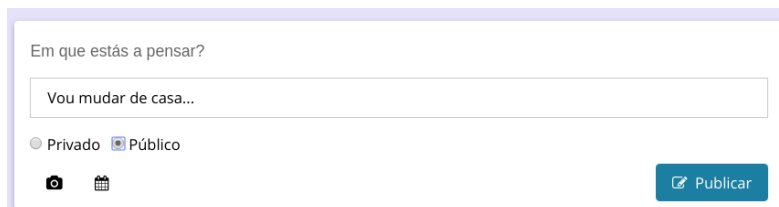


Figura 6: Menu para o utilizador fazer uma publicação.

Se o utilizador apenas clicar no botão de publicar será publicada uma ideia.

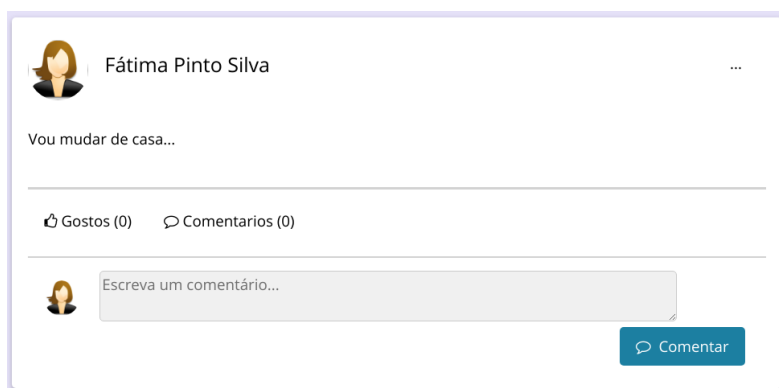


Figura 7: Ideia publicada pelo utilizador.

Caso este pretenda criar um álbum ou um evento, basta clicar no botão da máquina fotográfica ou no botão do calendário, respetivamente, o que redirecionará o utilizador para uma nova página, onde este poderá preencher os dados da publicação.

The screenshot shows the 'Criar Álbum' (Create Album) page. At the top, there is a header bar with the 'myFacebook' logo, a search bar, and navigation icons. The main form area has a light purple background. The title 'Criar Álbum' is centered at the top of the form. The form fields are as follows: 'Nome *' (Name) with the value 'Casamento'; 'Descrição' (Description) with the value 'O início...'; 'Local' (Location) with the value 'Braga'; 'Fotografias *' (Photos) with three slots, the first containing 'casamento.jpg', the second 'istock_82086545_large.jpg', and the third 'No file chosen'; 'Descritores' (Tags) with the value '#casamento #amigos #familia'; and 'Privacidade' (Privacy) with radio buttons for 'Privado' (selected) and 'Público'. A 'Confirmar' (Confirm) button is at the bottom right.

Figura 8: Página de criação de um álbum.

The screenshot shows the 'Criar Evento' (Create Event) page. The layout is similar to the album creation page. The title 'Criar Evento' is centered at the top of the form. The form fields are: 'Nome *' (Name) with the value 'Casamento'; 'Descrição' (Description) with the value 'O meu casamento.'; 'Local' (Location) with the value 'Braga'; 'Data de início *' (Start Date) with the value '2019-03-05'; 'Data de fim *' (End Date) with the value '2019-03-05'; 'Descritores' (Tags) with the value '#casamento #familia'; and 'Privacidade' (Privacy) with radio buttons for 'Privado' (selected) and 'Público'. A 'Confirmar' (Confirm) button is at the bottom right.

Figura 9: Página de criação de um evento.

Assim, o álbum e o evento resultantes estão ilustrados nas figuras 10 e 11, sendo possível verificar, também, nestas a interação de outros utilizadores com as publicações já criadas.

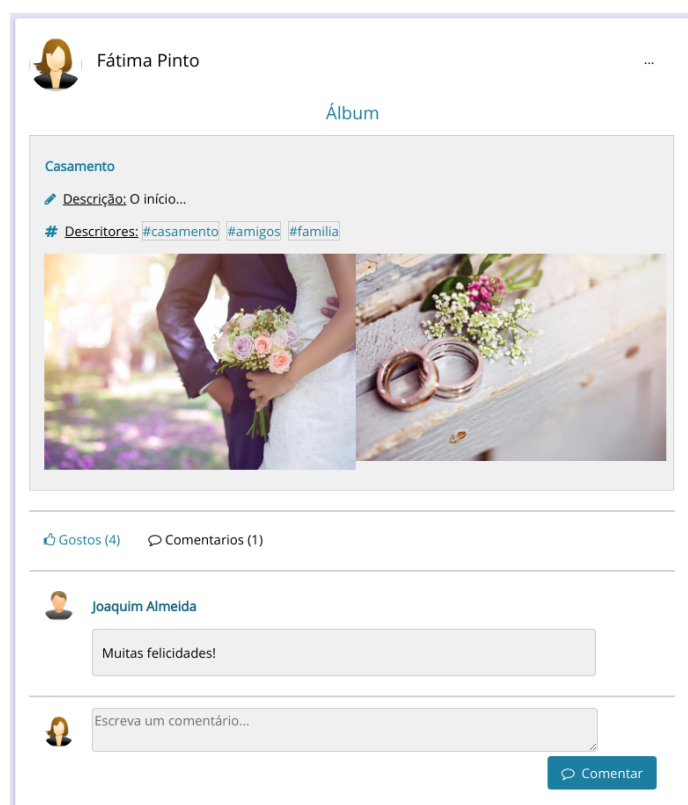


Figura 10: Álbum publicado pelo utilizador.

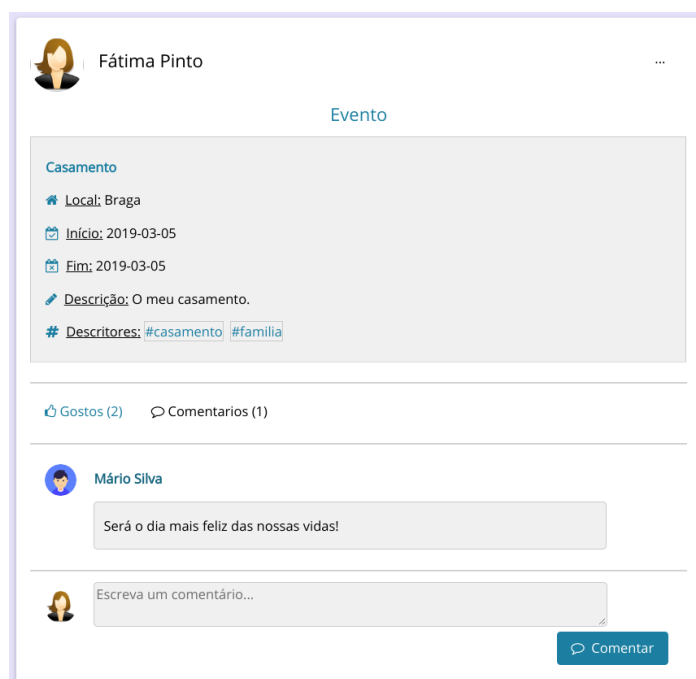


Figura 11: Evento publicado pelo utilizador.

Caso o utilizador queira, poderá editar ou eliminar qualquer publicação que fez, através do botão de três pontos existente no canto superior direito de cada uma. Caso pretenda editar a publicação, será redirecionado para a página de edição da mesma, sendo apresentados nesta os campos correspondentes ao tipo de publicação.



Figura 12: Página para editar um álbum.

Para além disto, o utilizador também tem disponível no seu *feed* uma janela que lhe permite saber, de entre os eventos disponíveis, aquele que será o próximo a ocorrer.



Figura 13: Evento mais perto de ocorrer.

3.4.3 Menu

Em qualquer página que o utilizador esteja, com exceção da página inicial, este tem ao seu dispor um conjunto de atalhos com diferentes funcionalidades. Ele poderá voltar ao seu *feed*, pesquisar por um dado utilizador, ver os eventos disponíveis e a lista de utilizadores existentes e efetuar *logout*.

Ao inserir o nome de um outro utilizador na barra de pesquisa, o utilizador será redirecionado para uma página onde aparecerão todos os utilizadores com esse nome.



Figura 14: Página resultante da pesquisa pelo nome de utilizador.

Ao clicar no símbolo do calendário com um visto (primeiro símbolo do canto superior direito), o utilizador será redirecionado para uma página que contém todos os eventos públicos criados.

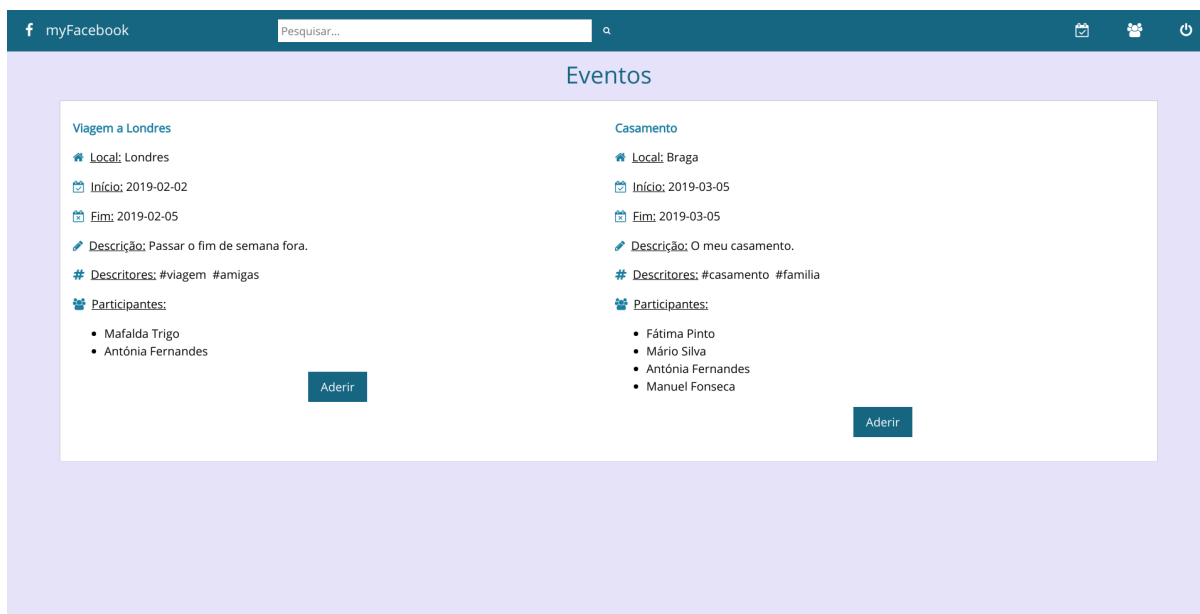


Figura 15: Página com todos os eventos públicos criados.

Por fim, ao clicar no símbolo dos “três utilizadores” (segundo símbolo do canto superior direito), o utilizador será redirecionado para uma página que contém todos os outros utilizadores registados na aplicação.

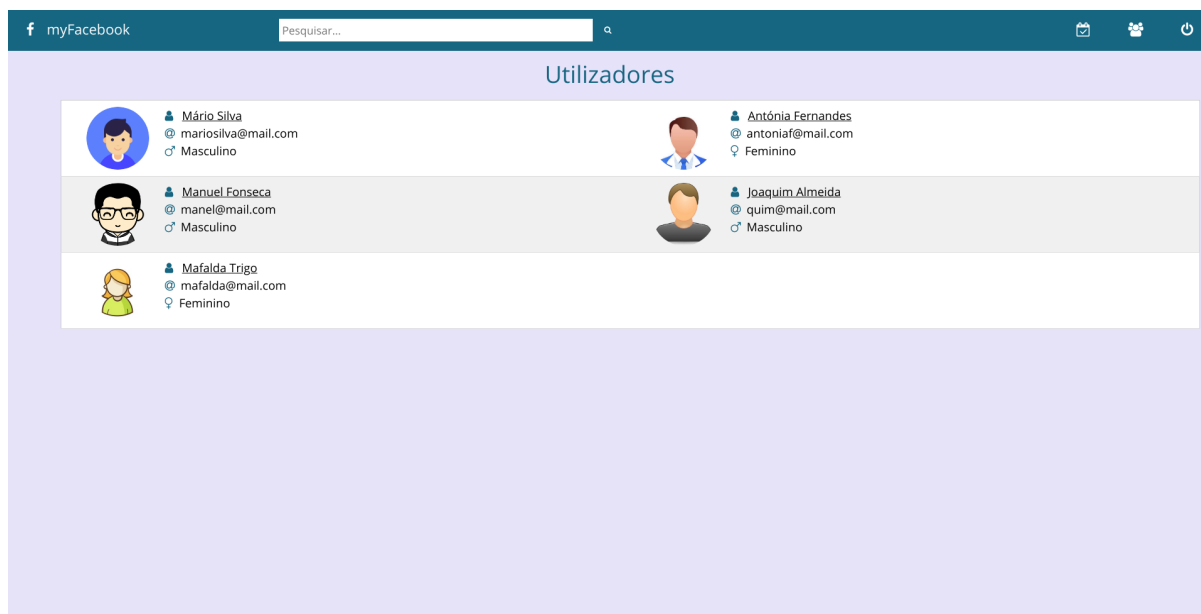


Figura 16: Página com todos os outros utilizadores registados.

3.4.4 *Feed* de outro utilizador

Através da página que mostra todos os restantes utilizadores ou pesquisando pelo nome de outro utilizador, o utilizador com *login* feito poderá aceder ao *feed* de quem pretender. Lá, este poderá visualizar todas as publicações (ideias, eventos e álbuns) públicas do utilizador a que pertence o *feed*. Também poderá deixar o seu “gosto” e comentar.

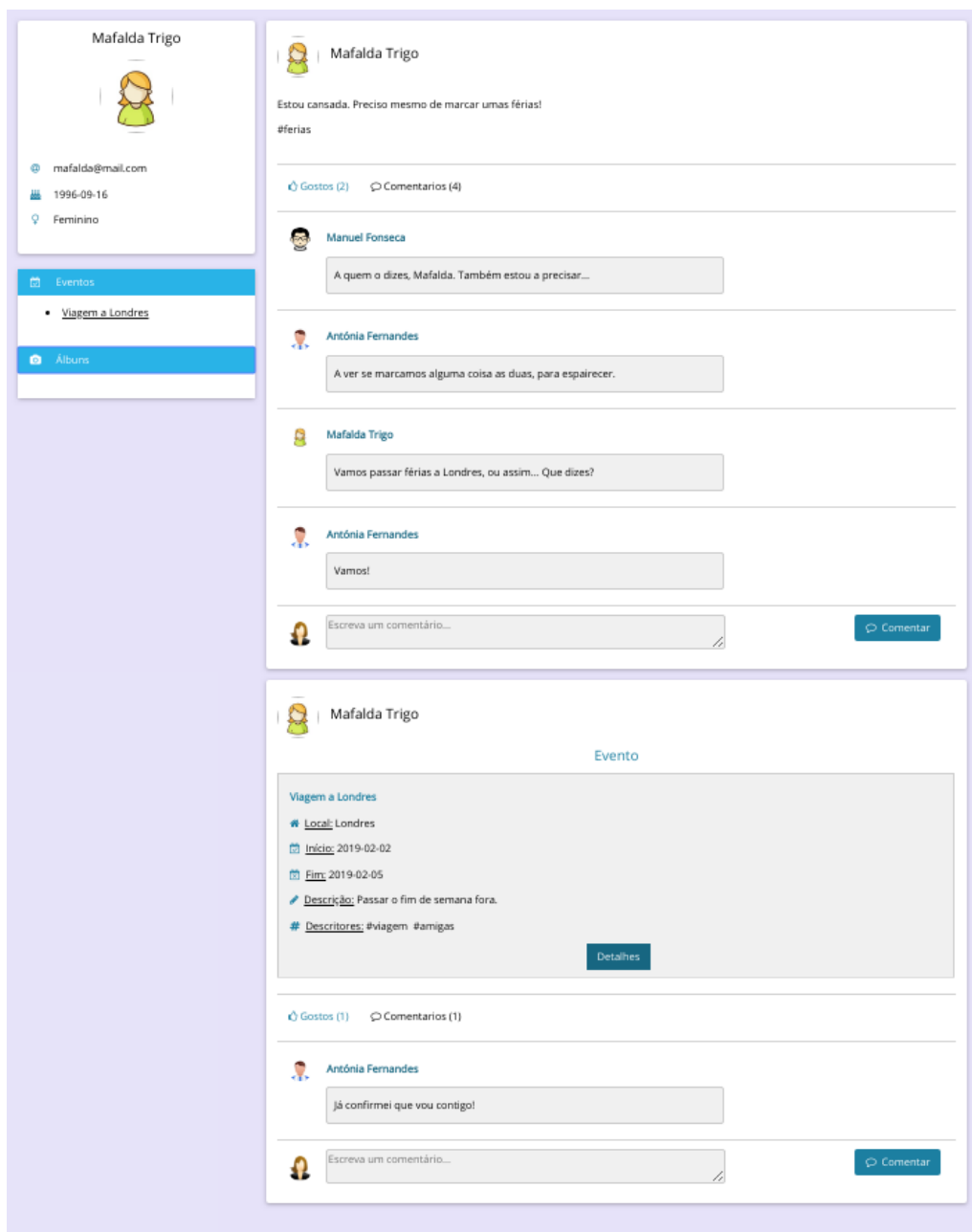


Figura 17: *Feed* de outro utilizador.

4 Conclusão

De uma perspetiva geral, consideramos que a realização deste exercício foi relativamente bem sucedida, visto que pensamos ter cumprido com o exercício proposto.

Como trabalho futuro poder-se-ia desenvolver um menu de notificações que avisasse os utilizadores de quando outros fizessem comentários, se adicionassem aos seus eventos ou tivessem deixado um “gosto” nas suas publicações. Para além disso, poderia ser implementada, futuramente, a opção de editar e remover comentários, assim como colocar “gosto” nestes.

Em suma, este trabalho permitiu consolidar melhor os conceitos lecionados nas aulas ao longo do semestre, ganhando, assim, prática e uma maior facilidade na utilização das ferramentas referidas na secção 1.

A Documentação da API RESTful

A.1 Autenticação

Rota	Formato	Resposta Sucesso	Resposta Erro	Descrição
POST /api/utilizador /register	{ email: String, password: String, nome: String, sexo: String }	{ email: String, password: String, nome: String, sexo: String }	{ erro: String }	Registrar utilizador
POST /api/utilizador /login	{ email: String, password: String }	{ email: String, password: String }	{ erro: String }	Autenticar utilizador
POST /api/utilizador /logout	{ }	{ }	{ erro: String }	Terminar sessão

Figura 18: API relativa à autenticação

A.2 Utilizadores

Rota	Formato	Resposta Sucesso	Resposta Erro	Descrição
GET /api/utilizadores	-	{ users: [User] }	{ erro: String }	Lista de utilizadores
GET /utilizador/:id	-	{ user: User }	{ erro: String }	Obter um utilizador
GET /utilizador /publicacoes/	-	{ pubs: [Item] }	{ erro: String }	Lista publicações relativas a todos utilizador segundo um descritor
POST /utilizador	{ email: String, password: String, nome: String, nasc: String, sexo: String }	{ email: String, password: String, nome: String, nasc: String, sexo: String }	{ erro: String }	Adicionar um utilizador
PUT /utilizador/:id	{ email: String, password: String, nome: String, nasc: String, morada: String, sexo: String }	{ email: String, password: String, nome: String, nasc: String, morada: String, sexo: String }	{ erro: String }	Atualizar informação de um utilizador
DELETE /utilizador/:id	-	{ email: String, password: String, nome: String, nasc: String, morada: String, sexo: String }	{ erro: String }	Remove um utilizador

Figura 19: API relativa aos utilizadores

A.3 Items

Rota	Formato	Resposta Sucesso	Resposta Erro	Descrição
GET /items	-	{ items: [Item] }	{ erro: String }	Lista de itens
GET /item/:id	-	{ item: Item }	{ erro: String }	Obter um item
GET /item/:id /comentarios	-	{ comentarios: [Comentario] }	{ erro: String }	Lista de comentários relativos a um item
POST /item/:id /comentario	{ id_utilizador: ObjectId, descricao: String }	{ descricao: String }	{ erro: String }	Adiciona um comentário a um item
PUT /item/:id /comentario/	{ descricao: String }	{ descricao: String, data: String, gostos: Number }	{ erro: String }	Editar um comentário
DELETE /item/:id /comentario/	-	{ descricao: String }	{ erro: String }	Remover um comentário
POST /item	{ titulo: String, tipo: String, local: String, elementos: [Elemento], privacidade: Boolean, descritores: [String], descricao: String }	{ titulo: String, tipo: String, local: String, elementos: [Elemento], privacidade: Boolean, gostos: Number, comentarios: [Comentario], descritores: [String], descricao: String }	{ erro: String }	Adicionar um item
POST /item/:id/like	-	{ titulo: String, tipo: String, local: String, elementos: [Elemento], privacidade: Boolean, gostos: Number, comentarios: [Comentario], descritores: [String], descricao: String }	{ erro: String }	Adicionar gosto

Figura 20: API relativa aos items

PUT /item/:id	{ titulo: String, tipo: String, local: String, elementos: [Elemento], privacidade: Boolean, descritores: [String], descricao: String }	{ titulo: String, tipo: String, local: String, elementos: [Elemento], privacidade: Boolean, gostos: Number, comentarios: [Comentario], descritores: [String], descricao: String }	{ erro: String }	Atualizar informação de um item
DELETE /item/:id/like	-	{ titulo: String, tipo: String, local: String, elementos: [Elemento], privacidade: Boolean, gostos: Number, comentarios: [Comentario], descritores: [String], descricao: String }	{ erro: String }	Remover gosto
DELETE /item/:id	-	{ titulo: String, tipo: String, local: String, elementos: [Elemento], privacidade: Boolean, gostos: Number, comentarios: [Comentario], descritores: [String], descricao: String }	{ erro: String }	Remove um item

Figura 21: API relativa aos itens