

Universidade do Minho
Escola de Engenharia

Engenharia Web

Relatório - *Digital Me*

3 de Fevereiro de 2018

Mestrado Integrado em Engenharia de
Telecomunicações e Informática

2017/2018

Identificação do grupo de trabalho



João Henrique Vivas dos Santos Simões, A68461
a68461@alunos.uminho.pt



Rui Filipe Coelho da Silva, A68466
a68466@alunos.uminho.pt



Ricardo Batista Rodrigues, A68468
a68468@alunos.uminho.pt

1 Descrição da Aplicação

A aplicação desenvolvida (*Digital Me*) trata-se de uma espécie de diário digital, e como tal as operações sobre os dados têm uma cronologia associada, ou seja, a linha temporal será o eixo principal da aplicação.

Esta aplicação web foi desenvolvida em *JavaScript*, utilizando a *framework Express*. Para o armazenamento dos dados foi utilizada uma base de dados em *MongoDB*. O *frontend* da aplicação foi desenvolvido em *Pug*.

Inicialmente, antes do início do desenvolvimento da aplicação, foi necessário fazer um levantamento de todos os requisitos funcionais e não funcionais da aplicação. Nomeadamente, todos os tipos de itens que a aplicação iria suportar e definir o tipo de meta-informação associado a cada item.

Posto isto, o grupo de trabalho definiu seis tipos de publicações que a aplicação suportará: Fotos, Vídeos, Receitas, Eventos, Pensamentos e Registos Desportivos. Cada uma destas publicações pode ser pública ou privada (por defeito privada), sendo que as definidas como públicas podem ser visualizadas por todos os utilizadores e as privadas apenas pelo próprio utilizador que a publicou. Cada utilizador pode tornar as suas publicações públicas ou privadas sempre que desejar.

É de realçar que esta aplicação web tem suporte para múltiplos utilizadores, tendo uma área de Registo e LogIn. De seguida iremos fazer uma breve apresentação e explicação do funcionamento da aplicação desenvolvida.

1.1 Página Principal

Depois de se registar ou fazer *LogIn* com sucesso, cada utilizador da deparar-se-á com a *homepage* da aplicação, como mostra a figura 1.

Como referido anteriormente, a aplicação desenvolvida trata-se de um "diário digital" e por esse motivo as publicações dos utilizadores aparecem por ordem cronológica (das mais recentes para as mais antigas).

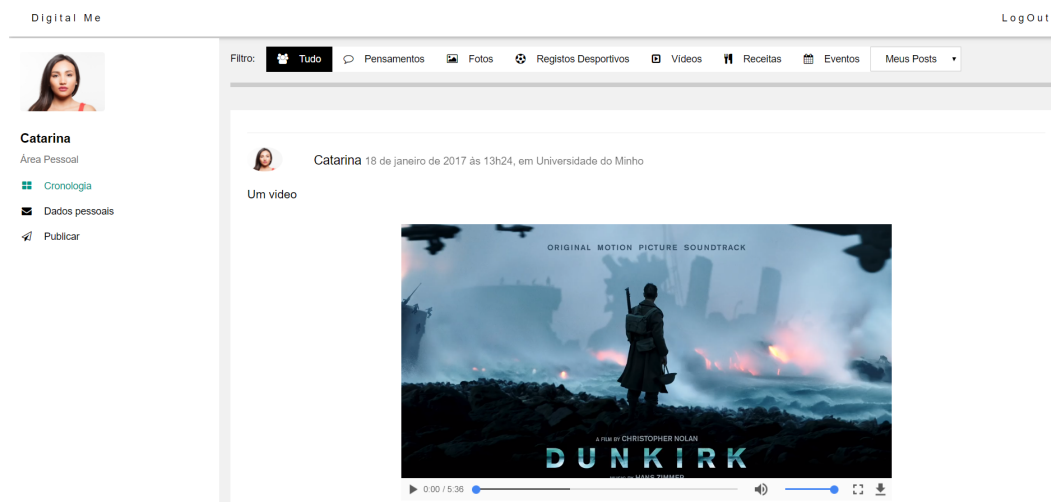


Figura 1: *Homepage* da aplicação.

Como podemos observar, o utilizador na sua *homepage* pode realizar enu-
 meras operações, nomeadamente, visualizar as publicações públicas de todos
 os utilizadores, filtrar as publicações por tipo (Fotos, Vídeos, Receitas, Even-
 tos, Pensamentos e Registos Desportivos), filtrar as suas publicações (Todas,
 Públicas e Privadas), fazer uma nova publicação, alterar os seus dados pesso-
 ais (foto, email, nome, *password* e género), comentar publicações e alterar a
 privacidade (Pública ou Privada) das suas publicações.

1.2 Publicação

Na figura 2 podemos observar o *template* de uma publicação do tipo Registo
 Desportivo. É de realçar que para cada tipo de publicação existe um *template*
 específico.



Figura 2: *Template* publicação -Registo Desportivo.

Como podemos observar, esta publicação tem uma data, um local e uma descrição associados (comum a todas as publicações) e um conjunto de campos específicos (tipo de desporto, duração e calorias gastas).

Como foi dito anteriormente, é possível comentar a publicação e podemos constatar que esta já possui um comentário. Por outro lado, como esta publicação foi feita pelo utilizador com sessão iniciada este pode alterar a privacidade da publicação (Pública/Privada) e eliminá-la se pretender.

1.3 Publicar

Quando o utilizador seleciona o botão de "Publicar" tem a possibilidade de escolher o tipo de publicação que pretende fazer. Na figura 3 podemos observar a realização de uma publicação do tipo Evento.

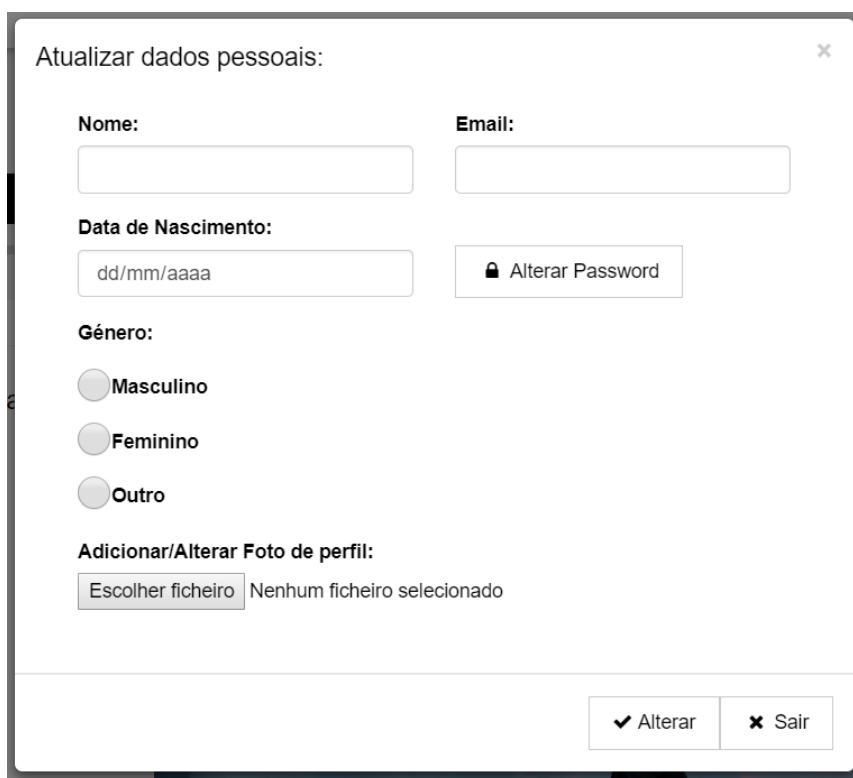
The screenshot shows a web form titled "Publicação:" with a close button (X) in the top right corner. Below the title is a horizontal navigation bar with icons and labels for different publication types: Pensamentos, Fotos, Registos Desportivos, Vídeos, Receitas, and Eventos (which is highlighted). The main section is titled "Publicar Evento:". It contains several input fields: "Nome do Evento:" with the value "Jantar Finalistas MIETI", "Data(dd/mm/aa):" with "06/04/2018", "Hora (hh:mm):" with "20:00", and "Localização:" with "Universidade do Minho". There is a "Descrição:" text area containing "Jantar de finalistas do curso de Engenharia de Telecomunicações e Informática 2017/2018". Below this is a file selection section "Escolher Foto:" with a button "Escolher Ficheiros" and the text "Nenhum ficheiro selecionado". To the right is a "Tipo de Evento:" dropdown menu set to "Convívio". At the bottom left is a "Duração do evento:" field with "03:00". At the bottom right is a "Privacidade:" dropdown menu set to "Privado". At the very bottom are two buttons: "✓ Publicar" and "✗ Cancelar".

Figura 3: Realização de uma publicação - Evento.

Como podemos observar, existem várias características específicas de cada tipo de publicação. É de realçar que é possível adicionar mais de que uma foto. Podemos observar que a privacidade da publicação é sempre Privada por defeito.

1.4 Dados Pessoais

Como foi referido anteriormente cada utilizador da aplicação tem a possibilidade de atualizar os seus dados pessoais a qualquer momento, como podemos observar na figura 4.



The image shows a web form titled "Atualizar dados pessoais:" with a close button (X) in the top right corner. The form contains several input fields and buttons:

- Nome:** A text input field.
- Email:** A text input field.
- Data de Nascimento:** A date input field with the placeholder "dd/mm/aaaa".
- Alterar Password:** A button with a lock icon.
- Género:** Three radio button options: "Masculino", "Feminino", and "Outro".
- Adicionar/Alterar Foto de perfil:** A section with a button labeled "Escolher ficheiro" and the text "Nenhum ficheiro selecionado".
- Bottom Buttons:** Two buttons at the bottom right: "✓ Alterar" and "✗ Sair".

Figura 4: Atualização dos dados pessoais.

Como é possível observar, o utilizador pode atualizar todos seus dados pessoais, nomeadamente, o nome, o email, a *password*, a data de nascimento, o género e a foto. Tem a possibilidade de alterar vários campos, ou apenas um. É de referir que sempre que o nome ou a foto são atualizados todos as publicações e comentários associados a esse utilizador são atualizados igualmente.

2 Implementação da Aplicação

Nesta secção irá ser descrito de forma resumida toda a implementação da aplicação web desenvolvida. Esta pode ser dividida em três partes: *front-end*, modelo de dados e *back-end*.

Primeiramente iremos abordar a componente de *front-end* do projeto, descrevendo partes de código que achamos relevantes. Posto isto, explicaremos a nossa abordagem ao problema em si e de que forma construímos o nosso modelo de dados. Por último iremos descrever o *back-end* da aplicação mostrando o tratamento da aplicação a vários pedidos HTTP.

2.1 *Front-end*

Como foi enunciado anteriormente o *front-end* da aplicação foi desenvolvido em Pug. Pug é um "*template engine*" para Node.js que permite introduzir dados e posteriormente produzir HTML.

De seguida iremos apresentar um excerto de código Pug que faz o *rendering* das publicações do tipo Pensamento.

```
1 each post in lposts
2   - var testpriv = {id1:""+user._id,id2:""+post.posted_by_id}
3   case post.post_type
4     // TEMPLATE PENSAMENTO
5     when 0: .w3-container.w3-white.w3-margin.w3-padding-
6             large
7             .w3-row.w3-margin-bottom
8             hr
9             .w3-col.l1.m3
10            img(src="/"+post.posted_by_pic, style="
11               width:50px; border-radius: 50%")
12            .w3-col.l16.m9
13            h4= post.posted_by+" "
14            span.w3-opacity.w3-medium= post.post_date
15            if post.posted_in
16            span.w3-opacity.w3-medium= ", em "+post.
17              posted_in
18            .w3-justify
19            p(style="font-size:120%")= post.thought_text
20            p.w3-left
21            .w3-section.w3-bottombar.w3-padding-16.
22            button-container
23            button.w3-button.w3-white.w3-border(type=
24              "button", data-toggle='modal', data-
25              target="#myModal" style="margin-top:10
26              px")
27            b
28            i.fa.fa-commenting-o
29            | Comentar
```

```

23         if testpriv.id1==testpriv.id2
24             .dropdown
25                 form(method="POST" action="/
26                     changeprivacy/"+post._id)
27                     select.w3-button.w3-white.w3-border(
28                         name="privacy" onchange="this.form
29                         .submit()" style="margin-top:10px;
30                         margin-left:10px;")
31                     option(value="priv") Privacidade
32                     option(value="priv") Privado
33                     option(value="pub") Publico
34                 form(action="/deletepost/"+post._id
35                     method="POST")
36                 button.w3-button.w3-white.w3-border(
37                     type="submit" style="margin-top:10
38                     px; margin-left:10px;")
39
40             b
41                 i.fa.fa-commenting-o
42                 | Remover Post
43
44         p.w3-right
45             button#myBtn.w3-button.w3-black(onclick="
46                 myFunction('"+post._id+"')")
47             b Comentar
48             span.w3-tag.w3-white= post.post_comments.
49                 length
50         p.w3-clear
51             .w3-row.w3-margin-bottom(style="display:
52                 none", id="" + post._id)
53             ul.list-unstyled
54                 each i in post.post_comments
55                     li
56                         .w3-row.w3-margin-bottom
57                         .w3-col.l2.m3
58                         img(src="/" + i.comment_by_pic, style
59                             ="width:90px")
60                         .w3-col.l10.m9
61                         h4= i.comment_by+ " "
62                         span.w3-opacity.w3-medium= i.
63                             comment_date
64                         p= i.comment_body

```

Excerto de Código 1: *Rendering* das publicações do tipo Pensamento

Neste excerto de código podemos constatar que é percorrida a lista (*lposts*) que contém todas as publicações da base de dados e, para cada publicação (*post*) é aplicado um *template* diferente conforme o tipo o seu tipo. Neste caso trata-se de uma publicação do tipo Pensamento. Para cada tipo de publicação foi atribuído um número (0 a 4) de forma a diferenciar cada uma.

Depois disso e, para cada tipo de publicação, é carregada a foto do utilizador que fez essa publicação, a data e o local. Neste caso como se trata de um Pensamento é apenas carregado o texto do pensamento num parágrafo. Em seguida é definido um botão "Comentar" onde, caso seja selecionado, é executado o *target* "#MyModal" onde é possível escrever um comentário à publicação.

Em seguida é verificado, através de uma condição, se o id do utilizador com sessão iniciada é igual ao id do utilizador que fez a publicação, caso seja é definido um *dropdown* onde é possível alterar a privacidade da publicação (Pública/Privada) e é ainda definido um botão onde o utilizar pode eliminar a publicação em causa.

Por fim, quando o botão "Comentários" é selecionado são listados todos os comentários da publicação caso existam. É percorrida a lista de todos os comentário à publicação (*post.post-comments*) e para cada um é definida uma lista onde é carregada a foto do utilizador que fez o comentário o seu nome e o conteúdo do comentário.

2.2 Modelo de dados

A construção do modelo de dados foi uma das primeiras tarefas a ser realizada. Primeiramente foram levantados os requisitos e foi estabelecido um *scope* base que nos serviu numa fase primordial da aplicação. Com o decorrer do projeto foram realizadas algumas adaptações e adições no que toca à meta informação, o resultado final foi o que se pode ser observado na fig. 5.

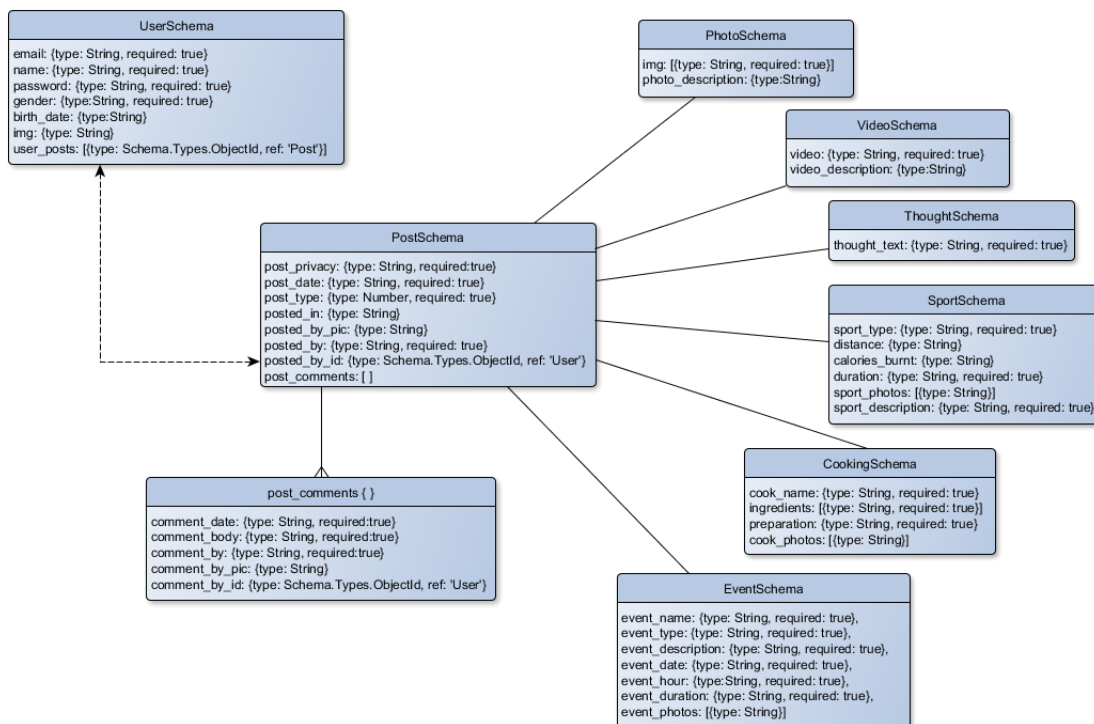


Figura 5: Modelo de dados da aplicação.

Um aspeto interessante foi a utilização do mecanismo de *discriminator*, isto permitiu criar um Schema genérico com todos os campos comuns a qualquer tipo de publicação. A partir daí foram criados os Schemas específicos tais como o SportSchema ou o EventSchema com os respetivos campos, herdando depois os campos do PostSchema.

2.3 *Back-end*

Na componente de *back-end* iremos mostrar primeiramente todos os URL's para cada tipo de pedido HTTP. Após isso iremos explicar um ou outro método que trata um pedido específico. Podemos observar a tabela seguinte:

Tabela 1: Tabela de URL's.

Método	URL	Descrição
GET	/	Carregamento da página de LogIn e Registo de utilizadores.
	/signup	LogOut e redireccionamento para a página de LogIn.
	/signout	LogOut e redireccionamento para a página de LogIn.
	/homepage	Carregamento da homepage com todos os posts públicos dos utilizadores
	/homepage/thoughts /homepage/pics /homepage/videos /homepage/sports /homepage/recipes /homepage/events /homepage/myposts	Carregamento da homepage com todos os posts públicos de um certo tipo (filtragem). O último URL consiste em listar apenas as publicação do utilizador, sendo que este pode listar os seus posts públicos ou privados ou todos.
	/login	Trata do LogIn do utilizador, em caso de sucesso realiza o redireccionamento para a /homepage em caso de erro /.
	/signup	Trata do Registo do utilizador, em caso de sucesso realiza o redireccionamento para a /homepage em caso de erro /.
POST	/homepage/post	Trata da acção de realizar uma publicação independente do seu tipo.
	/homepage/post/:id/comment	Permite ao utilizador postar um comentário numa publicação com o identificador id
	/changeprivacy/:idPost	Altera a privacidade da publicação com o identificador idPost
	/deletepost/:idPost	Remove o publicação com o identificador idPost
	/homepage/:id/change profdata	Altera os dados do perfil do utilizador com o identificador id

De seguida podemos observar alguns excertos de código que achamos relevantes mencionar. O primeiro representa o tratamento da aplicação a um pedido HTTP POST a um comentário feito a uma publicação.

```
1  /* Handle Comment on some Post */
2  router.post('/homepage/post/:idPost/comment', isAuthenticated
3    , (req, res, next)=>{
4    var date = new Date();
5    var comment = {comment_date: date, comment_body: req.body.
6      comment_body, comment_by: req.user.name, comment_by_pic:
7      req.user.img, comment_by_id: req.user._id}
8    Post.update({_id: req.params.idPost},$push: {post_comments:
9      comment}},
10     (err,result)=>{
11       if(!err)
12         console.log("Adicionei um comment ao post: "+ req.
13           params.idPost)
14       else
15         console.log("Erro a adicionar comentario: "+ err)
16     })
17     res.redirect('/homepage')
18   })
```

Excerto de Código 2: Tratamento de um comentário numa publicação

Neste excerto de código é criada uma variável com todos os campos preenchidos. Após isso é feita uma query que insere esta variável ao *post* com o id presente no *req.params*.

O próximo excerto representa como foram implementados os filtros por tipo de publicação. O método resume-se a fazer um *render* da *homepage* atribuindo um array de posts *lposts* que irá conter as publicações do tipo que se escolheu.

Os posts são do tipo correto e estão colocados por ordem cronológica, pois a query possui o *sort* e as *constraints* indicadas.

```
1  // filter recipes
2  router.get('/homepage/recipes', isAuthenticated, function(req
3    ,res){
4    console.log("Recipes")
5    Post.find({post_type:"4"}).where({post_privacy:"pub"}).sort
6      ({post_date: -1}).exec((err,doc)=>{
7      if(!err){
8        res.render('homepage', {lposts: doc, user:req.user})
9      }
10     }
11     else
12       res.render('error', {error: err})
13   })
14 // filter events
15 router.get('/homepage/events', isAuthenticated, function(req,
16   res){
17   console.log("Events")
```

```

16     Post.find({post_type:"5"}).where({post_privacy:"pub"}).sort
17       ({post_date: -1}).exec((err,doc)=>{
18         if(!err){
19           res.render('homepage', {lposts: doc, user:req.user})
20         }
21         else
22           res.render('error', {error: err})
23       })
24   })

```

Excerto de Código 3: Tratamento de filtragem de publicações