

# Ournote

Artur Queiroz

Luis Albuquerque

## **Resumo**

Trabalho prático de Desenvolvimento Web

# Conteúdo

<b>Introdução</b>	<b>3</b>
<b>Modelo</b>	<b>3</b>
Users . . . . .	3
Groups . . . . .	3
<b>Esquema</b>	<b>4</b>
POST . . . . .	5
PUT . . . . .	5
DELETE . . . . .	5
Interface . . . . .	5
GET . . . . .	5
POST . . . . .	5
PUT . . . . .	5
DELETE . . . . .	5
Cliente . . . . .	6
<b>Níveis de premissão</b>	<b>6</b>
<b>Conclusão</b>	<b>6</b>
Trabalho futuro . . . . .	6

## Introdução

Já existem muitas redes sociais, umas mais voltadas para partilhar fotos, outras para partilhar o estado das pessoas, e outras até mais focadas no mundo de trabalho, no entanto não existem assim tantas focadas em desenvolvimento e partilha de notas entre alunos. Com isso em conta, desenvolvemos Ournote, que é uma rede social focada em alunos, organizada em grupos e em páginas, onde o conteúdo pode ser organizado como num caderno, mas com a vantagem de aqui ser permitido facilmente trocar a ordem dos textos, imagens, ficheiros, etc. Os grupos são constituídos por páginas, e uma página esta organizada por uma meta linguagem, desenvolvida por nós, que permite distinguir títulos de diferentes importâncias, criar paragrafos, listas de paragrafos. imagens, eventos, pdfs, e até ficheiros distintos para fazer download.

## Modelo

No nosso modelo os nossos modelos foram distinguidos em duas classes, users e groups.

### Users

TODO: COMENTARIOS

```
name: String,
email : {
  type: String,
  unique: true,
  lowercase: true,
  required: true
},
password: {
  type: String,
  required: true
},
favourite: [ String ],
```

### Groups

TODO: COMENTARIOS

```
var eventSchema = new mongoose.Schema({
  title : String,
  data  : String
});
var card_Schema = new mongoose.Schema({
  p: String,
```

```

    img: String,
    pdf: String,
    h1: String,
    h2: String,
    h3: String,
    a: String,
    file : String,
    list : [String],
    comment : [String],
    event : eventSchema,
    tags : [String]
  });

var PATH = String;
var EMAIL = String;

var groupSchema = new mongoose.Schema({
  path : { //id
    type :String,
    unique: true,
    required: true
  },
  id_creator: EMAIL,
  name: {
    type: String,
    required: true
  },
  tags : [String],
  sub_groups : [ PATH ],
  read_perm : [ EMAIL ],
  write_perm : [ EMAIL ],
  page : [ card_Schema ],
});

```

## Esquema

Nós dividimos o trabalho em dois servidores (API e Interface) e clientes. ##  
API ### GET

```

/*?token=TOKEN                --- dados do grupo
/profile?token=TOKEN&email=EMAIL --- favoritos de um utilizador
/profile?token=TOKEN&email=EMAIL&tag=TAG --- Sitios que têm associado uma determinada tag
/user/:email?token=TOKEN      ---

```

## POST

/*?token=TOKEN&update=comment	--- adiciona um comentario a um objeto
/*?token=TOKEN&update=add	--- adiciona premissões a um ou mais utilizadores
/*?token=TOKEN&update=remove	--- remove premissões a um ou mais utilizadores
/*?token=TOKEN&type=file	--- upload de um ficheiro
/*?token=TOKEN	--- adiciona um novo grupo
/login?token=TOKEN&email=EMAIL	--- faz login
/favourite?token=TOKEN&email=EMAIL&path=PATH	--- adiciona uma página aos favoritos
/register	--- regista um utilizador

## PUT

/*?token=TOKEN&type=TYPE	--- acrescenta um objeto à página
--------------------------	-----------------------------------

## DELETE

/*?token=TOKEN	--- remove um objeto da página
----------------	--------------------------------

## Interface

### GET

/	--- responde com a página de login
/register	--- responde com a página para registar
/profile?tag=TAG	---
/logout	--- termina a sessão de um utilizador
/*	--- página de um grupo

### POST

/add_favourite?path=PATH	--- adiciona aos favoritos
/register	--- regista um utilizador
/login	--- faz login
root/*?update=add	--- acrescenta premissões
root/*?update=remove	--- remove premissões
root/*?update=comment	--- acrescenta um comentario
root/*?type=TYPE	--- acrescenta um ficheiro

### PUT

root/*	--- acrescenta um objeto
--------	--------------------------

### DELETE

root/*	--- elimina um ou mais objetos
--------	--------------------------------

## **Cliente**

## **Níveis de premissão**

## **Conclusão**

Ao longo desta jornada, encontramos diversos desafios, quer na discussão e esquematização da solução, como na própria implementação, como decidir qual seria a nossa abordagem e ao facto de termos usado javascript, e isso nos trazer muitos problemas na hora de debugging, uma vez que os erros muitas vezes ficam perdidos entre a API e a Interface, mas tendo o postman podemos testar individualmente a API, e com isso identificar mais facilmente a razão do problema. No fim de contas, conseguimos apresentar uma rede social funcional, com uma proposta bem original, que acreditamos que consiga ser útil para estudantes organizarem seus trabalhos, e a sua vida escolar.

## **Trabalho futuro**

No nosso trabalho, usamos uma meta linguagem para representar as páginas dos grupos, uma boa adição, seria adicionarmos elementos recursivas e outras funcionalidades, como canvas e suporte de videos, etc. Podiamos criar um parser, e com ajuda do pandoc convertermos as páginas em pdf, powerpoint, word, epub, etc. Infelizmente, não conseguimos fazer autenticação de ficheiros, apesar de termos tentado de diversas formas, acrescentar isso, seria fulcral para um bom funcionamento da aplicação. Outra coisa, menos importante, mas que seria mais conveniente para o utilizador, era ocultarmos as páginas que o utilizador não tem premissão de leitura. Neste momento, é possível acrescentar favoritos, mas não elimina-los, isso seria uma boa adição.