

Ournote

Rede Social

Artur Queiroz

Luis Albuquerque

Resumo

Trabalho prático de Desenvolvimento Web

Conteúdo

Introdução	3
Modelo	3
Users	3
Groups	3
Esquema	5
POST	5
PUT	5
DELETE	5
Interface	5
GET	5
POST	5
PUT	6
DELETE	6
Cliente	6
Conclusão	6
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

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

Os *favourites* são um conjunto de caminhos de grupos

Groups

Cada evento tem um título e uma data

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

Uma *card* é um elemento de uma página pode ser um parágrafo, um título uma imagem, etc

```

var card_Schema = new mongoose.Schema({
  p: String,
  img: String,
  pdf: String,
  h1: String,
  h2: String,
  h3: String,
  a: String,
  file : String,
  list : [String],
  event : eventSchema,

```

Tudo acima, dentro de *card* são os elementos que uma página pode ter

```

  comment : [String],
  tags : [String]

```

Lista de comentários e uma lista de *tags*

```

});

```

```

var PATH = String;
var EMAIL = String;

```

Um *group* é algo que tem uma página com várias *cards* e outros sub-grupos, com *tags* e umas permissões, tanto de leitura como de escrita.

```

var groupSchema = new mongoose.Schema({
  path : {
    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

```
/root/*?token=TOKEN  
/profile?token=TOKEN&email=EMAIL  
/profile?token=TOKEN&email=EMAIL&tag=TAG  
/user/:email?token=TOKEN
```

POST

```
/root/*?token=TOKEN&update=comment  
/root/*?token=TOKEN&update=add  
/root/*?token=TOKEN&update=remove  
/root/*?token=TOKEN&type=file  
/root/*?token=TOKEN // add group  
  
/login?token=TOKEN&user=USER  
/favourite?token=TOKEN&email=EMAIL&path=PATH  
/register
```

PUT

```
/root/*?token=TOKEN&type=TYPE
```

DELETE

```
/root/*?token=TOKEN
```

Interface

GET

```
/  
/register  
/profile?tag=TAG  
/logout  
/root/*
```

POST

```
/add_favourite?path=PATH  
/register  
/login  
root/*?update=add  
root/*?update=remove  
root/*?update=comment
```

root/*?type=TYPE

PUT

root/*

DELETE

root/*

Cliente

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.