Desenvolvimento de Aplicações Web

Trabalho Prático

Relatório de Desenvolvimento

Miguel Oliveira pg41088 | Moisés Ramires a80499 | Pedro Moura pg41094

28 de Janeiro de 2020

1 Introdução

O seguinte relatório descreve o desenvolvimento do trabalho prático da UC Desenvolvimento de Aplicações Web, que tem como objetivo aplicar e consolidar os conhecimentos e competências adquiridos durante o decorrer do semestre.

A aplicação aqui descrita segue o modelo MVC (*Model View Controller*). Para o modelo utilizamos MongoDB, que usa documentos JSON para guardar informação, as interfaces foram desenvolvidas usando principalmente o Pug e o Bootstrap e o controlo foi implementado através de um servidor HTTP em NodeJS.

2 Aplicação

Foi-nos pedido o desenvolvimento de uma rede social para alunos onde fosse possível fazer várias coisas como partilhar material, discutir datas, etc. Visto isto, decidimos chamar à nossa aplicação **S4S** (*Students for Studends*).

2.1 Funcionalidades

A aplicação funciona à base de publicações e tópicos. Os utilizadores criam perfis que contêm informações acerca do nome, número de aluno e curso que frequentam. Uma vez criada a conta, um utilizador pode:

- Editar o perfil, podendo adicionar uma foto de perfil e subscrever a novos tópicos. Por predefinição, cada utilizador está subscrito ao tópico referente ao seu curso;
- Fazer publicações com determinados tópicos, com a possibilidade de anexar ficheiros;

- Adicionar/Remover tópicos para seguir ou que segue;
- Fazer e apagar comentários nas publicações;
- Consultar todas as publicações com um determinado tópico;
- Consultar as publicações mais recentes;
- Consultar informações relativas a outro utilizador, e as publicações feitas pelo mesmo;
- Pesquisar publicações pelo seu conteúdo;

3 Modelo

A nossa base de dados é composta por duas coleções: **users** e **posts**. Para interagir com ela, utilizámos a biblioteca *Mongoose*, que permite definir **schemas** e efetuar pedidos REST.

Os *schemas* que utilizamos são os seguintes:

- **User**

```
const UserSchema = new mongoose.Schema({
    id: { type: String, required: true },
    password: { type: String, required: true },
    full_name: { type: String, required: true },
    course: { type: String, required: true },
    subscribed_tags: { type: [String], required: true },
    profile_pic: { type: String, required: true, default: "no-profile.png" }
});
```

- **Post**

```
const FileSchema = new mongoose.Schema({
    name: String,
    mimetype: String,
    size: Number
});

const Comment = new mongoose.Schema({
    date: { type: Date, required: true },
    content: { type: String, required: true },
    from: { type: String, required: true }
});

const PostSchema = new mongoose.Schema({
```

```
    date: { type: Date, required: true, default: Date.now },
    user_id: { type: String, required: true },
    title: { type: String, required: true },
    tags: { type: [String], required: true },
    content: { type: String, required: true },
    attachments: { type: [File], required: true },
    comments: { type: [Comment], required: true }
});
```

4 Interfaces

As nossas *views* foram desenvolvidas, principalmente, com recurso ao motor **Pug** e à *framework* **Bootstrap**, bem como pequenas bibliotecas de modo a implementar uma interface agradável com a qual o utilizador possa interagir.

5 Controlo

A nossa aplicação contém um index e uma api, onde estão criadas as seguintes rotas.

Index(GET):

- / - páginal inicial da app onde o utilizador pode inicar sessão ou registar-se
- /register - páginal onde o utilizador se pode registar
- /login - páginal onde o utilizador pode iniciar sessão
- /logout - página onde o utilizador encerra sessão
- /posts -página com posts por tag e pesquisa
- /user/:userid - páginal de um utilizador autenticado, com os posts do mesmo
- /post/:idpost - páginal correspondente a um post com id = idpost
- /file/:filename - página correspondete ao download de um ficheiro de id = filename

Index(POST):

- /register - registo de um utilizador
- /login - login de um utilizador
- /user/profile_pic - inserção da imagem de um utilizador
- /post/:idpost/comment - adição de um comentário ao post

- /post - criação de um post

Index(PUT):

- /user/:userid/subscribed_tags - update das tags que o utilizador segue

Index(DELETE):

- /post/:idpost/comment/:idcomment - eliminação de um comentário

API, /api (GET):

- /user/:userid - devolve o utilizador de id = userid
- /posts/:userid - devolve os posts do utilizador de id = userid
- /post/:postid - devolve o post de id = postid
- /posts - devolve posts por tag ou pesquisa, ou por um limite, caso não seja disponibilizadas tags ou uma pesquisa

API, /api (POST):

- /post - cria um post
- /post/:idpost/comment - cria um comentario
- /user/ - cria um utilizador

API, /api (PUT):

- /user/:userid/subscribed_tags - altera as tags de um utilizador
- /user/:userid/profile_pic - altera a imagem de um utilizador

API, /api (DELETE):

- '/post/:idpost/comment/:idcomment - remove um comentário

6 Conclusão

Neste projeto criamos uma aplicação na qual é possivel a partilha de ficheiros e publicações, tal como comentar e seguir tags específicas. Os utilizadores podem alterar a sua foto e tem acesso a um chat, com o qual não nos encontramos satisfeitos devido a este funcionar apenas como uma sala geral para troca de mensagens. De um ponto de vista crítico estamos satisfeitos com o resultado do trabalho prático, porém existem aspectos práticos do projecto que podem ser melhorados, como o chat ter opção de mensgens individuais,e mais salas disponiveis, por exemplo uma por curso.