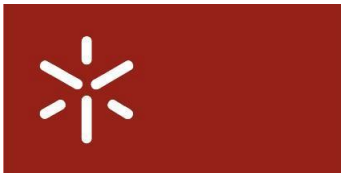

Universidade do Minho

Mestrado Integrado em Engenharia Informática

Desenvolvimento de Aplicações WEB

4º Ano, 1º Semestre

Social Bike Store

Bruno Magalhães (75377)

Joel Rodrigues (79068)

Luís Guimarães (77004)

29 de Janeiro de 2019

Resumo

O presente documento diz respeito à documentação do trabalho prático proposto na unidade curricular de Desenvolvimento de Aplicações WEB, uma unidade curricular complementar lecionada no 4º ano do Mestrado Integrado em Engenharia Informática da Universidade do Minho.

O trabalho proposto pelo docente consiste no desenvolvimento de uma aplicação web que implemente um repositório digital de registo de ideias, fotos e outros objetos com o intuito de partilhar ou armazenar digitalmente para consulta futura.

Conteúdo

1 Introdução

O presente documento diz respeito à documentação do trabalho prático proposta na unidade curricular de Desenvolvimento de Aplicações WEB, uma unidade curricular complementar lecionada no 4º ano do Mestrado Integrado em Engenharia Informática da Universidade do Minho.

O trabalho proposto pelo docente consiste no desenvolvimento de uma aplicação web que implemente um repositório digital de registo de ideias, fotos e outros objetos com o intuito de partilhar ou armazenar digitalmente para consulta futura.

A aplicação deverá ser constituída por um frontend, backend, persistência de dados em base de dados, ficheiros e lógica de controlo em *JavaScript*.

O intuito da aplicação é suportar uma *persona* digital do utilizador e atuará como um diário digital do mesmo. Isto quer dizer que todos os dados guardados deverão ter associados uma marca temporal de modo a serem apresentados num *feed* cronológico que pode ser consultado pelos restantes utilizadores da aplicação.

A aplicação desenvolvida tem como objetivo a partilha social, e possível venda, de motos. Ao longo deste documento estará documentado todo o processo de desenvolvimento desta aplicação, começando pela definição do domínio e motivações por detrás do projeto, prosseguido da definição dos vários componentes e sua implementação.

2 Domínio

Com o aumento significativo do preço dos combustiveis e dos impostos sobre a venda e circulação dos automóveis, muitas pessoas têm vindo a procurar alternativas para a sua mobilidade diária. Uma das soluções passa por adquirir um veículo de duas rodas que para além dos baixos custos de aquisição e manutenção, comparativamente a um autómovel, apresenta também vantagens nas capacidades de deslocação sendo possivel evitar filas e estacionar em qualquer lugar.

De acordo com os dados presentes na Associação Automovel de Portugal (ACAP), em 2018 foram matriculadas 28338 motas em Portugal, o que representa um aumento de 14,2% face ao valor de 2017. Desta forma, estima-se que o mercado dos motociclos cresca também em 2019, com mais individuos a optar por adquirir um destes veiculos.

O mercado online da venda de motociclos novos e usados em Portugal é algo muito pouco explorado. Opções como a plataforma *Standvirtual* onde é necessário efetuar pagamentos para poder efetuar anuncios reduz a utilização para as concessionárias e alguns individuos particulares. Já na plataforma *OLX*, apesar de ser possivel anunciar a venda de motociclos, não é uma plataforma dedicada ao efeito o que por vezes torna a experiência do utilizador menos ideal.

Surge então a ideia de desenvolver uma aplicação web que permite ao utilizador partilhar anuncios de venda de motociclos, assim como interagir com outros utilizadores. Nasce assim o SBS, *Social Bike Store*.

3 Estrutura da Aplicação

A aplicação foi desenvolvida com o auxílio da *framework* `express` disponível para `node`, e apresenta uma arquitetura semelhante a MVC (Model-View-Controller). A principal vantagem desta arquitetura é a divisão em módulos lógicos, repartindo lógicamente as diversas funcionalidades da aplicação.

Ao longo deste capítulo serão descritos os Modelos da aplicação e o seu formato; a API de dados, os seus controladores e roteadores; o módulo responsável pela autenticação dos utilizadores; e, por último, as diversas vistas que compõem a interface com o utilizador.

3.1 Modelos

Os modelos da aplicação são abstrações dos dados que se pretendem armazenar, descrevendo o que cada um representa e qual o domínio que alberga. Estes representam os objetos manipulados pela aplicação, e que se pretende que sejam persistidos. Para obter essa persistência, foi utilizada uma base de dados `mongoDB`, composta por três coleções que se pretendem armazenar: os utilizadores , `(users)`, as motos, `(bikes)`, e as publicações `(posts)`.

3.1.1 Utilizadores

Os utilizadores serão armazenados na coleção `users` e representam a *persona* digital do utilizador da aplicação. Este deverá ser composto por um *email* único, *password* de acesso à aplicação e dados pessoais como: nome, data de nascimento, número de telemóvel, fotografia, morada e nacionalidade. Por último, e destinado à venda das motos partilhadas, um *rating* que ajudará os restantes utilizadores a decidir um possível negócio. Assim sendo, e para armazenamento em base de dados, foi definido um *schema* para o utilizador:

```
var UserSchema = new Schema({
    email: {type: String, required: true, unique:true},
    password: {type: String, required: true},
    name: {type: String, required: true},
    birth: {type: String, required: true},
    cellPhone: {type: String, required: false},
    picture: {type: String, required: true},
    city: {type: String, required: true},
    district: {type: String, required: true},
    nationality: {type: String, required: true},
    rating: {type: String, required: true},
})

module.exports = mongoose.model('User', UserSchema, 'users');
```

3.1.2 Motos

As motos serão armazenadas na coleção `bikes` e representam uma única moto na aplicação, contendo a sua marca, modelo, ano e mês da compra, cilindrada, potência, quilómetros percorridos e estado atual. Assim sendo, o *schema* para armazenamento em base de dados é o seguinte:

```
var BikeSchema = new Schema({
   make : {type:String, required:true},
   model : {type:String, required:true},
   year : {type:Number, required:true},
   month : {type:String, required:true},
   cylinderCapacity: {type:Number,required:true},
   power:{type:Number,required:true},
   kilometers:{type:Number,required:true},
   condition:{type:String,required:true}
})

module.exports= mongoose.model("Bike",BikeSchema,"bikes")
```

3.1.3 Publicações

As publicações serão armazenadas na coleção `posts` e representam as várias entradas que o utilizador desejar guardar. Estas publicações incidem na partilha de diversas motos, permitindo aos restantes utilizadores ter acesso a estas entradas para deixar a sua opinião, podendo também ser destinada à venda das mesmas. Como cada publicação é feita por um utilizador sobre uma moto, esta deverá armazenar tanto o identificador do objeto `user`, como o identificador do objeto `bike`, bem como as várias opiniões deixadas pelos restantes utilizadores. Tem ainda armazenada a data de publicação, uma foto e dois atributos: "likes"e "dislikes"deixados pelos vários utilizadores da aplicação. Assim sendo, para armazenamento em base de dados, foi definido o seguinte *schema*:

```
var OpinionSchema = new Schema({
    user : {type: Schema.Types.ObjectId, ref: 'User' },
    text : {type:String, required:true}
 })

var PostSchema = new Schema({
    user: {type: Schema.Types.ObjectId, ref: 'User' },
    bike: {type: Schema.Types.ObjectId, ref: 'Bike' },
    picture: {type: String, required: true},
    postDate: {type: String, required: true},
    likes: {type: Number, required: true, default:0},
    dislikes: {type: Number, required: true, default:0},
    opinions: [OpinionSchema],
    state: {type: String, required: true, default:'Active'}
})

module.exports = mongoose.model('Post', PostSchema, 'posts');
```

3.2 API de dados

A API desenvolvida tem o intuito de abstrair a manipulação de dados da delegação das vistas. Assim, os *endpoints* apenas devolvem a informação pedida - em formato json - ou erros, seja esse o caso. Para todos os modelos foram implementadas as operações CRUD para garantir as funcionalidades minimas da aplicação. A API apresenta então os seguintes *endpoints*:

3.2.1 Utilizadores

- **GET /api/users/** - Lista utilizadores;
- **GET /api/users/:id** - Devolve utilizador por id;
- **GET /api/users?email** - Devolve utilizador por email;
- **POST /api/users/** - Insere um novo utilizador;
- **POST /api/users/:id/editPicture** - Atualiza a foto do utilizador com identificador id;

3.2.2 Motas

- **GET /api/bikes/** - Lista motas;
- **GET /api/bikes/:id** - Devolve mota com id;
- **GET /api/bikes?make** - Lista motas por marca;
- **GET /api/bikes?model** - Lista motas por Modelo;
- **POST /api/bikes/** - Insere uma nova mota;

3.2.3 Publicações

- **GET /api/posts/** - Lista todos os posts por ordem cronológica descendente;
- **GET /api/posts/:userid** - Lista todos os posts de um utilizador;
- **GET /api/posts?make** - Lista posts por marca de moto;
- **GET /api/posts?model** - Lista posts por Modelo de moto;
- **GET /api/posts?make&model** - Lista posts por marca e Modelo de moto;
- **POST /api/posts/** - Insere um novo post;
- **POST /api/posts/like/:id** - Like no post;
- **POST /api/posts/dislike/:id** - Dislike no post;
- **POST /api/posts/opinion/:id** - Regista uma nova opinião num post;

3.3 Autenticação

A autenticação é efetuada utilizando o módulo **Passport**. Assim que um utilizador entra no website, é gerada uma session que é persistida no *session store* baseado no sistema de ficheiros. Utilizamos o *Loki store* para persistir os dados das sessões, visto este não ter problema de implementação nas plataformas Windows. Durante a autenticação do utilizador, caso esta seja terminada com sucesso, é gerado e persistido - na session store - um Json Web Token que será posteriormente utilizado para verificar a autenticação dos utilizadores. Desta forma, e após um utilizador se autenticar, este pode continuar a utilizar a plataforma sem se voltar a introduzir as suas credencias. Caso a *tab* seja fechada, o utilizador pode voltar a aceder ao website sem ter de inserir as credenciais. O mesmo não se verifica caso o utilizador feche a janela do *browser* visto que a session está implementada para durar apenas até isso acontecer. Implementamos também uma estratégia de autenticação com o Facebook, o que permite a um utilizador se autenticar utilizando a sua conta de facebook. Com isso, o nosso servidor irá utilizar a API do facebook para requerer os respetivos campos de registo e criar um perfil na nossa plataforma.

3.4 Vistas

Ao estilo de qualquer rede social mas sem ser necessário registar primeiro, assim que um utilizador acede à plataforma obtém uma lista de todos as publicações, ordenados por ordem cronológica. Nela consegue verificar comentários e outras informações como likes e dislikes de uma publicação e perfis de outros utilizares. No entanto, apenas consegue realizar algum tipo de acção se efetuar um registo ou se se autenticar. Nessa mesma página inicial, o utilizador tem a possibilidade de proceder para a página de registo ou para a página de Login. Uma vez autenticado, um utilizador é redirecionado de novo para a página principal, mas desta vez sem restrições nas ações. Este pode ainda consultar o seu perfil e mudar a sua foto.

4 Importação / Exportação de dados

A aplicação oferece a funcionalidade de exportar e importar todos os dados guardados. A exportação dos dados gera um ficheiro `zip` que contém um ficheiro `json` com todos os dados guardados na base de dados, bem como duas pastas com as fotos de perfil dos utilizadores e as fotos submetidas nas publicações.

De modo a exportar os dados da base de dados, numa primeira fase, é utilizado o comando `mongoexport` para gerar os ficheiros respetivos a cada coleção. De seguida, cada um dos ficheiros é iterado de modo a copiar as imagens para uma pasta no ficheiro zipado e redefinir os paths. De seguida, as três coleções são juntas num só ficheiro json com o formato:

```
{
    "users":[User],
    "bikes":[Bike],
    "posts":[Post]
}
```

O nome de cada imagem é redifinido para ser único e facilmente identificável. Cada imagem de um post tem agora o nome `"post"+ post._id` e cada imagem de um utilizador tem o nome `"user"+ user._id`. O módulo utilizado para produzir o zip foi o `jszip`.

Para importação dos dados, foi primeiro definido um `schema`, para que os dados inseridos no json respeitem o formato de exportação. É disponibilizada uma página para proceder à importação, em que o utilizador fornece um ficheiro `json` com os dados e dois ficheiros `zip` com as fotos dos utilizadores e dos posts. As referências das imagens não têm de seguir o formato definido acima, mas as imagens referidas no json têm de estar corretamente definidas nos restantes zips.

Espelhando a exportação, o primeiro passo da importação é redfinir os paths das imagens para a sua localização do lado do servidor, separar os três arrays em coleções distintas, e proceder à execução do comando `mongoimport`.

5 Conclusão e trabalho futuro

O desenvolvimento de uma aplicação web como o **Social Bike Store** é um processo iterativo e incremental onde o tempo de desenvolvimento define o grau de complexidade da aplicação. O estado da aplicação atual é tal o que foi possível de desenvolver com o tempo disponível para realizar este projeto. O grau de dificuldade do desenvolvimento não é elevado, porém são enumeras as vezes em que se leva mais tempo do que o esperado devido a erros inesperados e de confusa resolução. Aquando da data final da entrega, consideramos que a aplicação desenvolvida apresenta especificações interessantes, aplicando algumas das tecnologias aprendidas nas aulas e outras aprendidas de forma extra curricular. O resultado final está de acordo com o esperado da distribuição e calendarização das tarefas e objetivos.

6 Referências

Referências

[1] Vendas de Motociclos, Ciclomotores e e Quadriciclos - http://www.autoinforma.pt/index.php?MIT=0&template_id=316&xpto=1&a[]=01,,36458,,,0,