

Licenciatura em Engenharia Informática e de Computadores

Programação na Internet, INVERNO de 2014/15



**ueixinhas**

Grupo 43

28499 – Victor Morais

31181 – Rui Dinis

33357– João Caselli

Docente:

Eng.º Carlos Guedes

Lisboa, 13 de Fevereiro de 2015

##### Conteúdo

##### Objectivo

Este trabalho teve como objectivo, aplicar e aprofundar os conhecimentos obtidos na cadeira de Programação e Internet, atrás do desenvolvimento de uma aplicação WEB.

##### Introdução

O Nome dado a aplicação desenvolvida foi *’Queixinhas’*, esta aplicação consiste em que diversos utilizadores possam interagir entre si através de publicações – queixinhas – possam comentar, votar, seguir, em suma, tudo o que permita uma discussão sobre o assunto apresentado.

A aplicação tem três tipos possíveis de utilizadores.

- Os utilizadores não registados, que tem acesso a um número limitado de queixinhas, em que destas só conseguem visualizar o seu título e a sua descrição.

- Os utilizadores registados, que tem acesso a ilimitado a todas as queixinhas, podem comentar, votar, seguir, criar, e fechar as suas próprias queixinhas. Podem também alterar os dados de utilizador (fornecidos no registo).

- Os utilizadores Gestores, que tem acesso a tudo a que um utilizador registado tem, mas tem ainda a função de gestão de aplicação, o que lhe permite entre outros, o arquivo de queixinhas, das quais não sejam criadores.

Todos os utilizadores que não se encontrem autenticados são considerados como utilizadores não registados.

A aplicação, tem ainda disponível um serviço de *‘contacto’,* para que sejam colocadas questões por qualquer utilizador, esta funcionalidade envia uma mensagem de email com o conteúdo escrito pelo utilizador para um email geral da aplicação, que é posteriormente acedido pelos gestores da aplicação.

Um utilizador, quando se regista é obrigado e inserir um endereço de email válido, no entanto a aplicação não verifica a existência do endereço inserido. Este endereço tem como objectivo poder ser utilizado na recuperação da palavra-passe ou do nome de utilizador, isto porque quando o utilizador tenta recuperar estes dados, é-lhe pedido o email com o qual se registou, pois será esse o endereço utilizado para envio dos dados em questão.

##### Desenvolvimento

Esta aplicação foi desenvolvida com recurso a NODEJS como tecnologia WEB, *PostgreSQL* como motor de base de dados relacional e *Bootstrap* como *framework* da componente visual.

Assim, e de forma a facilitar a explicação e as opções tomadas no desenvolvimento da aplicação, a parte de desenvolvimento pode-se dividir em três partes.

###### Componente Visual

Como já referido anteriormente a *framework* usada no desenvolvimento da componente visual foi o *BootStrap*.

O template base usado, foi uma versão modificada de *‘Business Casual’*, um template desenvolvido com recurso a elementos *Bootsrap*, que pode ser encontrado em:

- <http://startbootstrap.com/template-overviews/business-casual/>

Este template foi adaptado às necessidades da aplicação desenvolvida, no entanto, a base mantém-se a original.

Foram utilizados outros elementos, tais como imagens, facilmente encontradas no GOOGLE, e cujos respectivos direitos de autor não foram respeitados.

###### Server Side

A *framework* base usada para desenvolver a aplicação, foi o *ExpressJS* para *nodeJS*. Como *template engine*, foi usado o EJS, e como suporte para autenticação foi usado o *midleware* *PassportJS* .

A aplicação desenvolvida tem um sistema de rotas, que permite obter os recursos necessários ao normal funcionamento da aplicação. Existem dois tipos de rotas, as rotas que podem ser acedidas por qualquer utilizador, isto é, não existe necessidade de autenticação, não havendo necessidade da utilização do *midleware* de autenticação nessa rota. E as rotas que necessitam de autenticação, utilizando o *midleware* de forma a verificar se o utlizador se encontra ou não autenticado. Caso o utilizador não não se encontre autenticado, e na tentativa de acesso a uma destas rotas “protegidas”, o mesmo é reencaminhado para a página de login.

Na página de login, é possível também efectuar a recuperação da palavra passe, sendo esta recuperação feita através do email utilizado no registo. Esse email introduzido é verificado contra o email presente na base de dados e em caso afirmativo é enviado para o utilizador através desse mesmo email, o nome do utilizador e a respectiva palavra passe.

Para o envio do respectivo email, foi utlizado o *midleware* *request,* que permite a execução de um método *http*, sendo que neste caso especifico foi usado o método POST. É este o único sítio da aplicação onde este *midleware* é utilizado.

Para fornecer, suporte ao envio de email, foi usada a aplicação <http://mandrill.com/>, que após registo fornece um chave, que é enviada no *body* do pedido, de forma a autenticar o pedido. No pedido de contacto, dúvidas ou questões, que também procede ao envio de um email, mas neste caso para o gestores da aplicação, o pedido e efectuado com um pedido AJAX, não utilizando o *request*.

------------------------------------------------------------------------------------------------------------------------------

A aplicação foi estruturada recorrendo ás seguintes entidades, mapeadas em tabelas na base de dados.

Criamos uma entidade "queixinha" que representa a para com a entidade "comentário" as principais entidades da aplicação. A primeira agrega toda a informação pertencente a uma queixinha, sendo que ao especificado no enunciado acresentamos um campo de nome "state" que através de um booleano especifica o estado da queixinha (aberta ou fechada). De referir que uma queixinha tem sempre um utilizador "owner" que é o utilizador criador da queixinha. Um utilizador pode visualizar na aplicação uma listgem de todas as suas queixinhas. Esta funcionalidade será descrita mais á frente.

A entidade "comentário" representa os comentários que se podem adicionar a uma queixinha. Tambem aqui um comentário pertence ao utilizador que inseriu o mesmo, assim como um comentário foi inserido no ambito de uma determinada queixinha. Como tal estas duas relações foram modeladas nesta entidade.

Como o tema de uma queixinha insere-se numa determinada categoria de temas, criamos tambem a entidade "categoria" que não é mais do que metadata representada na base de dados na forma de uma tabela simples com o descritivo de cada categoria. A inserção, edição e remoção de categorias não foi prevista na aplicação, mas a ser seria realizada pelo utilizador gestor através de uma eventual aplicação de administração.

A entidade "dbuser" representa um utilizador da aplicação. Nesta tabela guardamos o username, password e email de um utilizador assim como o nome que quer usar no âmbito da aplicação, modelado pelo campo "nickname". Existe tambem um campo "role" que serve para destinguir os utilizadores gestores dos demais utilizadores. A razão de termos chamado a esta entidade dbuser e não user, prende-se com o facto de "user" ser uma palavra reservada em PostgreSQL. Daqui para a frente chamaremos simplesmente user á entidade dbuser.

A última entidade nativa que criamos foi a entidade "vote". Esta representa os votos dos utilizador numa determinada queixinha. Para isso temos nesta entidade como seria de esperar, as relações para um user e para uma queixinha. Acresentamos ainda o campo "value" que com um booleano representa o sentido do voto.

Por último criamos uma tabela auxiliar "queixinha\_dbuser". Esta destina-se a implementar a funcionalidade de seguimento de queixinhas pelos utilizadores. Assim quando um utilizador pretende seguir uma queixinha, ou seja ser notificado das alterações de estado da mesma, é inserido nesta tabela a relação entre o utilizador em questão e a queixinha a seguir. Um utilizador pode seguir tantas queixinhas quantas quiser. Nesta entidade criamos ainda um campo "dirty" que mais uma vez através de um booleano permite assinalar as queixinhas que sofreram alterações ainda não visualizadas pelo utilizador seguidor. Assim este estado começa por default a false, mudando para true quando a queixinha for adicionada de um comentário ou for alterada pelo proprietário da mesma. Assim que o utilizador seguidor observar a queixinha em questão (através de um pedido GET a esta queixinha especifica) o estado deste campo retorna a false.

A utilização destas entidades na aplicação é feita através de DAOs que agrupados na pasta “db” não só manipulam o acesso á base de daos destas mesmas entidades como estabelecem representações das mesmas no context NodeJs. Na pasta “routes” criámos rotas para atender aos aspectos funcionais da aplicação, sendo que a manipulação da informação necessária a cada pedido tratado nestas rotas é feito através dos DAOs antes descritos.

Ao descrevermos o funcionamento da aplicação iremos acompanhar essa descrição com referências aos promenores de implementação mais relevantes.

LOGIN

Depois de autenticados na aplicação podemos aceder a uma lista de todas as queixinhas fechadas existentes. Sendo esta lista paginada, foi necessário acresentar ao URI “/queixinhas/” o campo “page” que representa a página desta listagem. Assim a fim de efectuar esta paginação estabelecemos que no pedido inicial não é inserido o campo “page”, o que vai despoltar um pedido á base de dados da contagem de todas as queixinhas fechadas existentes através do método “getTotalNumberClosedQueixinhas” e posteriormente guardado numa variável global. Sendo o número de queixinhas por página imutável e igual a 5, esta contagem é passada á vista por forma a ser construido em cada pedido o sistema de paginação com o número de páginas certo.

Ao escolher uma queixinha especifica desta lista, somos redirecionados para os detalhes dessa queixinha incluindo todos os comentários efectuados até ao momento e esssa queixinha. Para criar esta vista é necessário recolher toda a informação relativa a esta queixinha. Aqui foi nossa opção efectuar o provisionamento desta informação do lado do servidor Web e não como uma função da base de dados. A razão desta nossa escolha foi o facto de que ainda que saibamos que seria mais acertado fazer esta operação do lado da BD isso não cairia no ânbito desta cadeira. Assim a função que trata da rota “queixinhas/id”, é uma função que executa inumeros pedidos á base de dados em sequência, a fim de obter informação acerca da própria queixinha, os comentários dessa queixinha, o numero total de votos, assim como saber se a queixinha já foi votada e se é seguida pelo utilizador em uso. Esta última informação tem relevância a fim e bloquear ou não as funcionalidades de seguir e votar uma queixinha.