

Universidade De São Paulo - USP
Instituto de Ciências Matemáticas e de Computação - ICMC

Trabalho 3

Introdução ao Desenvolvimento Web

Funcionalidade do Servidor da Aplicação Pet Shop

Bruno Bacelar Abe	9292858
Paulo R. C. Barbosa	9779475
Marcelo Tabacnik	7573232

Professor:
Prof. Dr. Dilvan Moreira

São Carlos, Junho de 2018

Introdução	3
Testes	4
Implementação	12
Estrutura do projeto	12
Servidor	12
Cliente	13
Instruções de execução	14
Erros segunda etapa	16
Conclusão	17

Introdução

A partir da primeira parte do projeto, criação do mockup, foi criada uma aplicação web no modelo *Single Page Application* (SPA). Nesta última etapa, foi definida a criação do lado do servidor, bem como a criação de um banco de dados NoSQL.

No lado do cliente, foram usadas as linguagens HTML (para construção das páginas) e CSS (para sua estilização). Além disso, foram utilizados os frameworks JQuery (para a atualização das páginas) e Bootstrap (para customização do HTML, bem como para garantir responsividade).

No lado do servidor, foi utilizado o NodeJS, assim pudemos desenvolver ambos cliente e servidor utilizando JavaScript. Além disso, foi utilizado MongoDB para armazenamento NoSQL dos dados.

Dentro do site, é possível logar-se como usuário comum, ou administrador.

O usuário comum tem a capacidade de adicionar animais, marcar serviços para os animais que estão cadastrados, assim como comprar produtos. Todas as atividades relacionadas aos animais cadastrados podem ser acessadas pelo usuário. Além disso, o usuário será o único com acesso ao carrinho de compras, posto que é o único que pode comprar.

O administrador tem poderes maiores dentro da aplicação, podendo adicionar novos administradores, produtos, serviços e usuários comuns, além de gerenciar todo o site (produtos, serviços e usuários). A aplicação sempre começa com um usuário administrador padrão, para que todo o resto possa ser adicionado.

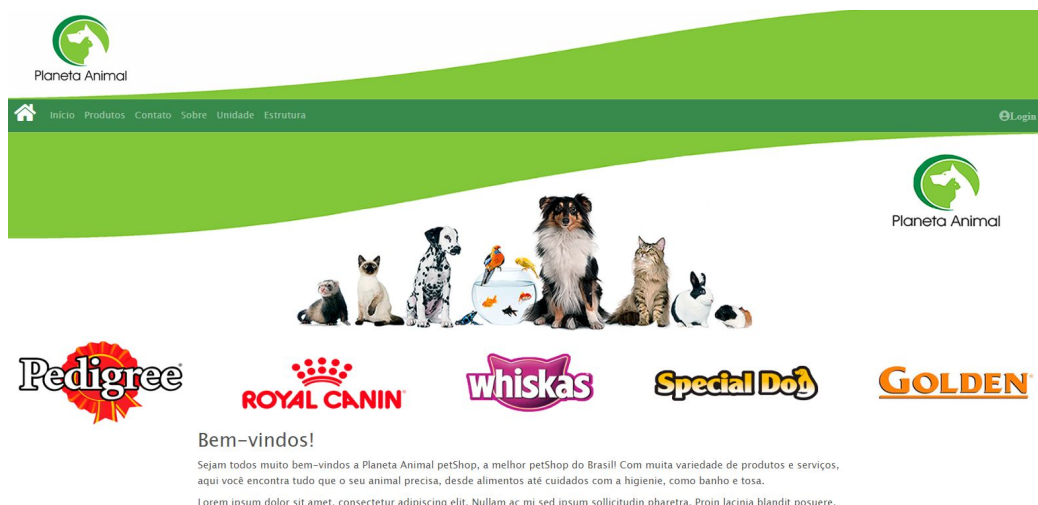
Nesta etapa, fez-se uso de servidores locais (criado automaticamente pela IDE WebStorm, que foi recomendada pelo monitor PAE), para hospedagem da aplicação e de suas mídias. Estamos utilizando o Node.js juntamente com o MongoDB no backend da aplicação, reproduzindo um servidor RESTful, que atende aos chamados do protocolo HTTP (como GET, POST, DELETE e PUT).

Para a execução do trabalho, basta ligar o servidor e acessar o localhost da porta previamente configurada (no nosso caso, a porta 3000).

Todo o projeto foi desenvolvido em uma plataforma Windows (Windows 10, x64), usando as versões mais atualizadas do *mongoDB*, *node.js*, *express* e o navegador browser usado foi o Google Chrome. Testes também foram feitos no Linux (Ubuntu 16.0 LTS).

Testes

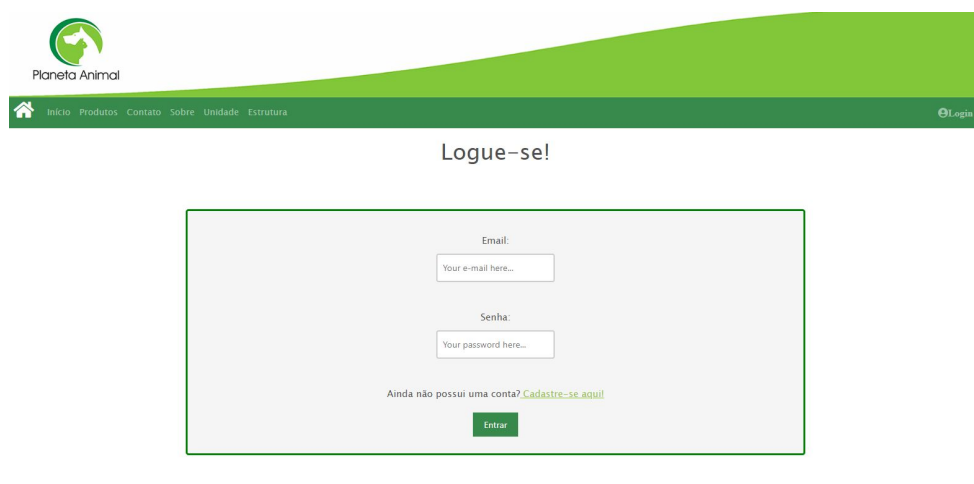
Uma vez que a SPA for executada, o usuário será direcionado para a tela inicial da PetShop, recebendo uma mensagem de Boas-vindas, além de poder logar-se ou acessar outros campos do site pelo menu de navegação.



(Imagem 1)

Como é possível ver, a barra de navegação possibilita que o usuário vá para diversos setores do site, que já foram descritos na primeira etapa, de forma clara e simplificada.

Após acessar a página inicial, é interessante que haja o login para acessar a conta, ou para cadastro. Ambas as ações podem ser realizadas clicando-se na aba "Login". Muitas funcionalidades, como acesso ao carrinho de compras, compra de produtos e agendamentos só são possíveis mediante o login.



(Imagem 2)

Nessa etapa, é possível identificar-se como usuário normal, ou usuário administrador. Assim, um acesso será feito ao banco de dados e caso a conta exista, o usuário será redirecionado para sua tela. Caso o usuário normal queira se cadastrar, basta clicar em “cadastre-se aqui”.

Uma vez cadastrado e/ou logado, o usuário terá acesso a sua página pessoal, onde poderá cadastrar seus animais, assim como listá-los.



Alterar foto

Adicionar Animal

Listar Animais

Meus Dados

Nome:

Email:

Telefone:

Endereço

Rua:

Número:

Bairro:

Cartão

Número:

Bandeira:

Sair

Salvar

(Imagem 3)

É interessante verificar que, após o login, a aba “Login” transforma-se em “Conta”, e a partir dela é possível acessar a conta do usuário sempre que necessário. O usuário também pode mudar seus dados (menos o email, que está sendo usado como chave primária), cadastrar o cartão de crédito e personalizar sua foto de perfil.

Próximo a foto, é possível adicionar um animal, ou listá-lo. Caso nenhum animal esteja cadastrado, uma lista em branco aparecerá.



Dados do animal

Nome:

Peso:

Dono:

Raça:

Raça Pai:

Raça Mae:

Serviços Reservados

(Imagem 4)

A imagem 4 ilustra a página de “listar animais”. Nesta página é possível modificar os dados do animal, alterar foto de perfil, deletá-lo, ou ir para o animal seguinte, além de mostrar todos os serviços relacionados à esse animal em específico.

Serviços Reservados



Banho

Data marcada: 2018-07-18T11:00:00.000Z

Valor: 30



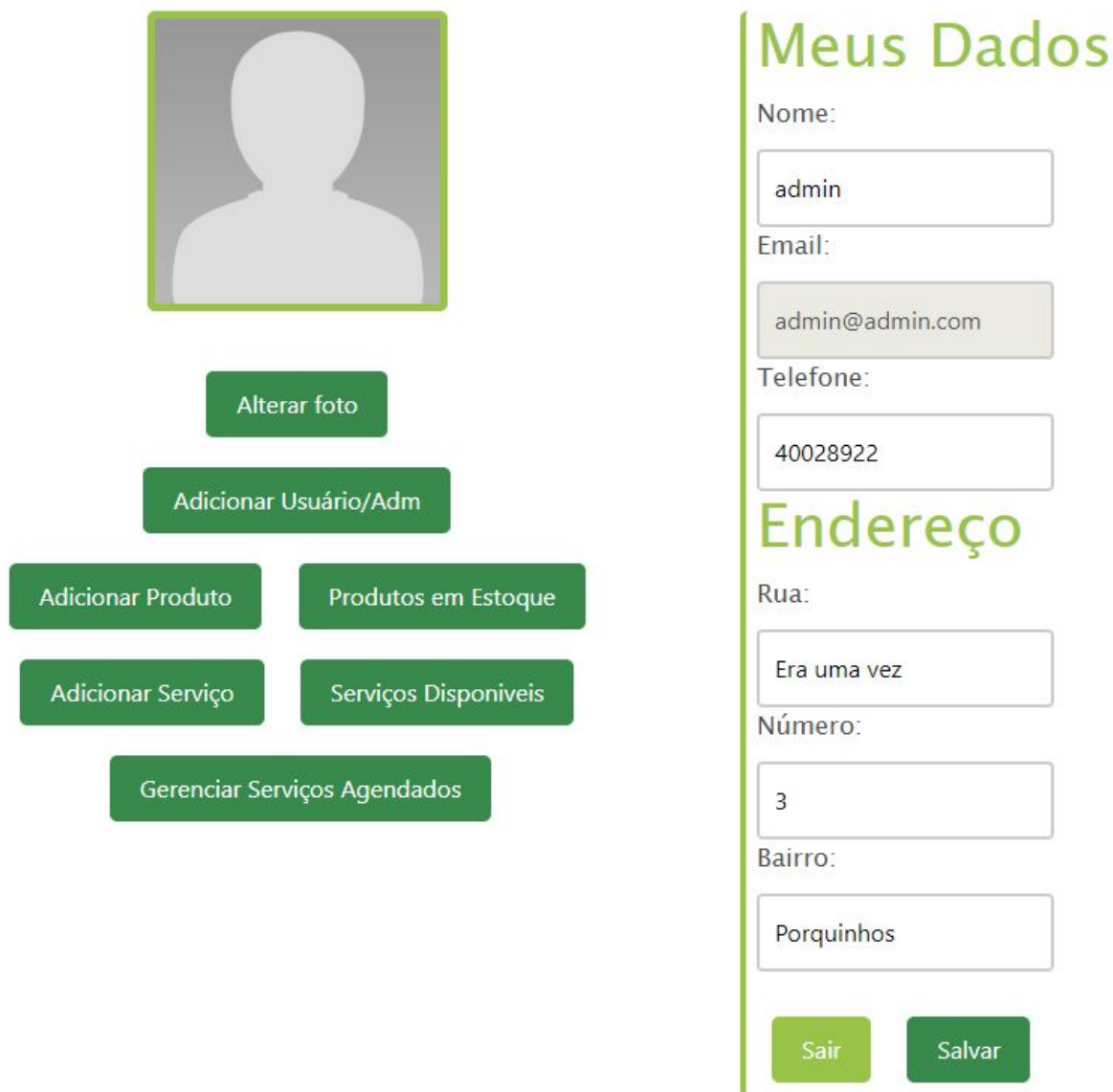
Tosa

Data marcada: 2018-07-18T11:00:00.000Z


Valor: 36

[Deletar](#)[Salvar](#)[Voltar](#)[Próximo](#)

A imagem 5 (pode ser vista a seguir), mostra a tela do administrador, que funciona de maneira semelhante ao do usuário normal, porém, as opções do mesmo são outras (listar serviços, estoque, gerenciar usuários, etc).



The image shows a user profile management interface. On the left, there is a placeholder for a user profile picture. Below it are several green buttons: 'Alterar foto', 'Adicionar Usuário/Adm', 'Adicionar Produto', 'Produtos em Estoque', 'Adicionar Serviço', 'Serviços Disponíveis', and 'Gerenciar Serviços Agendados'. On the right, there is a section titled 'Meus Dados' with input fields for 'Nome' (containing 'admin'), 'Email' (containing 'admin@admin.com'), and 'Telefone' (containing '40028922'). Below this is a section titled 'Endereço' with input fields for 'Rua' (containing 'Era uma vez'), 'Número' (containing '3'), and 'Bairro' (containing 'Porquinhos'). At the bottom right are two green buttons: 'Sair' and 'Salvar'.



Alterar foto

Adicionar Usuário/Adm

Adicionar ProdutoProdutos em Estoque

Adicionar ServiçoServiços Disponíveis

Gerenciar Serviços Agendados

Meus Dados

Nome:

Email:

Telefone:

Endereço

Rua:

Número:

Bairro:

SairSalvar

(Imagem 5)



É importante prestar muita atenção na hora do cadastro de usuários, pois o campo “tipo” deve ser preenchido da maneira correta (“normal”, para usuários normais, e “admin, para Administradores). Presume-se que, como essa é uma área restrita aos administradores do sistema, o cadastro de usuários será feito sempre da maneira correta.

Para o cadastro de produtos, é necessário uma atenção especial para o “código de barras” (que é a chave primária do mesmo no banco), para que não haja cadastro de mais de um produto com o mesmo código. Já para o cadastro de serviços, a chave primária consiste no nome do serviço (nunca há dois serviços com o mesmo nome). Ambos podem ser cadastrados com, ou sem, foto (As fotos, quando selecionadas, DEVEM estar na pasta “Imagens”, para que haja o funcionamento correto. Esse comportamento foi autorizado pelo monitor PAE). Caso

a imagem apresente um erro, o atributo alt será mostrado, e caso nenhuma foto seja selecionada, a imagem “Sem Imagem” irá aparecer.

Após o cadastro de produtos e/ou serviço, é possível acessar a tela de “Produtos” (localizada na lista de navegação) e compra-los.

Produtos

 <p>Ração Golden</p> <p>R\$54</p> <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit.atur! Lorem ipsum dolor sit amet.</p> <p>Em estoque 12</p> <p>Quantidade: <input type="text" value="0"/></p>	 <p>Cama</p> <p>R\$123</p> <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit.atur! Lorem ipsum dolor sit amet.</p> <p>Em estoque 2</p> <p>Quantidade: <input type="text" value="0"/></p>	<p>produto</p> <p>Osso de brinquedo</p> <p>R\$12</p> <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit.atur! Lorem ipsum dolor sit amet.</p> <p>Em estoque 1</p> <p>Quantidade: <input type="text" value="0"/></p>
--	--	---

Comprar

(Imagem 6)

Uma vez que a quantidade de itens seja selecionada, clica-se no botão “Comprar”, e o usuário é redirecionado para sua respectiva conta.

O processo para marcar serviços é análogo, porém, é necessário selecionar o nome do animal que receberá o serviço, assim como a data e hora do serviço.

Serviços



Banho

R\$30

Nome do animal(apenas para usuarios):

Nina ▼

Data:

Horário: ▼

(Imagem 7)

Após a seleção do produto e/ou serviço, o usuário é direcionado para o carrinho, onde pode finalizar a compra.

No servidor, há o uso de *logs* para identificação do que está acontecendo. No log, normalmente pode-se ver o documento salvo e seu *schema*, na imagem a seguir, podemos ver o comportamento para o login de um usuário normal e um usuário administrador.

<pre> Login - console node false [{ foto: 'Imagens\\gatoRacao.jpg', isAdmin: false, _id: 5b40ee913995940444c32bc5, password: '123', email: 'messengerabe@hotmail.com', numCartao: 0, bandeiraCartao: '1', idAdmin: 0, __v: 0, bairro: '1', nome: 'aaa', numCasa: 1, rua: '1', tel: 1 }] </pre>	<pre> Login - console node true true [{ foto: 'Imagens\\perfilFoto.jpg', isAdmin: true, _id: 5b41366a806c3623e8e53c7c, password: '123', nome: 'admin', email: 'admin@admin.com', tel: 2, rua: '6', bairro: '1', numCasa: 1, numCartao: 0, bandeiraCartao: ' ', idAdmin: 1, __v: 0 }] </pre>
--	---

(Imagem 8)

É importante salientar que podem existir diferenças nos produtos, nomes e animais cadastrados das imagens deste relatório se comparadas com o que está na base entregue. Houve a exclusão e adição de informações durante a implementação do trabalho, buscando encontrar e solucionar o maior número de falhas possíveis.

Implementação

Estrutura do projeto

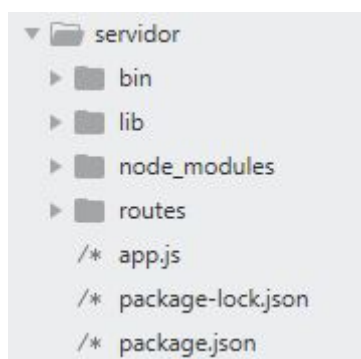
O projeto possui 3 pastas principais, cliente, data e servidor, cada uma alocando uma parte importante para o funcionamento do trabalho.

1. Cliente: Abriga todos os HTMLs, CSSs e Javascripts do cliente;
2. data: Abriga os dados da base de dados;
3. Servidor: Abriga todos os Javascript do servidor.

Servidor

Como servidor foi utilizado a ferramenta *node.js*, juntamente com o framework *express*. O uso do *express* se deu pela facilidade que o mesmo oferece no tratamento das requisições HTTP, aumentando a produtividade na criação do trabalho.

Como banco de dados, como foi dito anteriormente, foi usado o banco NoSQL *mongoDB*. A escolha do *mongoDB* deu-se pela facilidade trazida por ele na manipulação das coleções (muitas vezes assemelhando-se ao esquema de tabelas). No banco de dados, são definidos os *schemas* utilizados pelas tabelas (para melhor organização), assim como as rotas.



(Imagem 9)

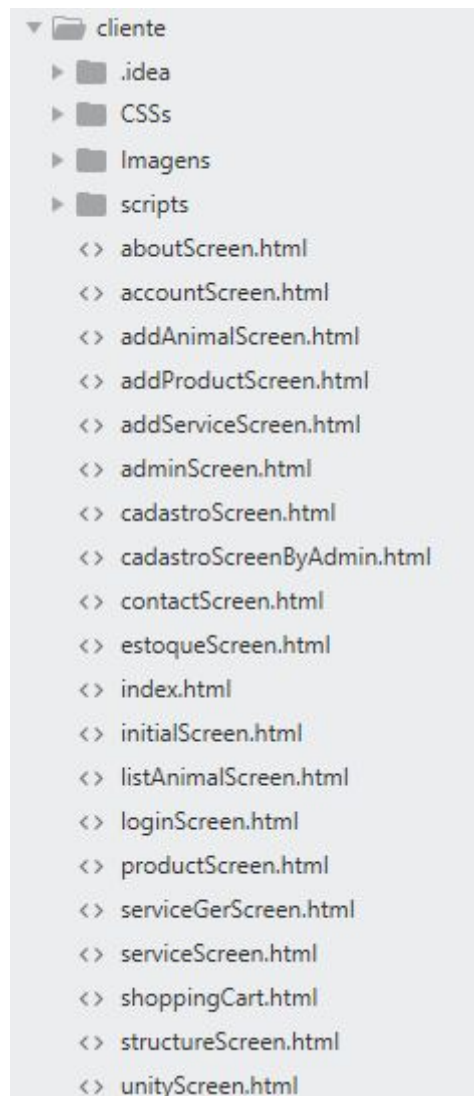
Como pode ser visto na imagem 9, há 4 pastas presentes na parte do servidor (bin, lib, node_modules e routes).

1. bin: Criado automaticamente pelo *express*, é responsável por configurações padrões da aplicação (como a porta do localhost);
2. lib: Possui todos os *schemas* utilizados no trabalho (animals, haveService, product, service e user). Os *schemas* são utilizados no tratamento das requisições HTTP;
3. no_modules: Criada automaticamente pelo *node.js*;
4. route: Trata as requisições HTTP vindas do cliente.

Cliente

A organização da pasta é a mesma da segunda etapa, possuindo todos os HTMLs, CSSs e Javascript do site.

Todas as requisições partem do cliente, sendo respondidas e atendidas pelo servidor. Por exemplo, em uma requisição do tipo GET (o cliente pede um dado para o servidor), o servidor recebe a requisição, trata-a e se for o caso, redireciona para o *mongoDB* para recuperar o dado em questão. As outras requisições funcionam de maneira análoga.



(Imagem 10)

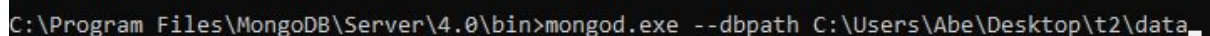
Instruções de execução

Para executar corretamente o trabalho, é necessário que as seguintes ferramentas estejam corretamente instaladas e funcionais:

1. *MongoDB*;
2. *Node.js* e *Express*;
3. WebStorm (ou servidor estático equivalente);

O *mongoDB*, *Node.js* e *Express* podem ser instalados a partir dos executáveis encontrados nos respectivos sites dos desenvolvedores.

Na execução do *mongoDB*, é importante que haja a definição de onde está a base de dados. Como estaremos utilizando o localhost, o monitor deve direcionar o banco de dados para a pasta *data*, presente no projeto.

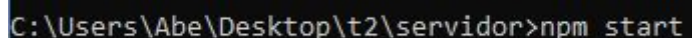


```
C:\Program Files\MongoDB\Server\4.0\bin>mongod.exe --dbpath C:\Users\Abe\Desktop\t2\data_
```

(Imagem 11)

A imagem 11 mostra como deve ser realizada a execução e direcionamento do *mongoDB*. O executável *mongod.exe* (que se encontra no diretório de instalação do banco) é o responsável pela inicialização do banco, enquanto que a flag *--dbpath* (path) é responsável pela definição de qual base do *mongoDB* está sendo usada pela aplicação. Caso a correção seja feita no Linux, dentro da pasta do projeto, execute o comando *mongod --dbpath ./data*.

Após a instalação do *node.js*, entre na pasta *servidor* e execute o seguinte comando: *npm install*, após a finalização do comando, uma nova pasta deve ter sido criada (*node_modules*). Execute o comando *npm install express*, *npm install express --save* e *npm install mongoose* respectivamente. Para a execução do servidor, é necessário que o comando *npm start* seja realizado dentro da pasta *servidor* (que contém os arquivos necessários para a inicialização do *node.js*).



```
C:\Users\Abe\Desktop\t2\servidor>npm start
```

(Imagem 12)

Por fim, para a inicialização do lado do cliente, é necessário que haja a criação de um servidor para o cliente. No nosso caso, foi utilizado a IDE WebStorm (recomendada pelo próprio PAE durante a execução da parte 2 do projeto). Essa IDE cria este servidor de maneira automática, não necessitando que o usuário crie o servidor manualmente.

Há dois usuários disponíveis para os testes, sendo um deles administrador e o outro um usuário normal.

O usuário administrador tem como usuário e senha, respectivamente, admin@admin.com e 123. Já o usuário normal, tem como usuário e senha, respectivamente, messengerabe@hotmail.com e 123 . O usuário normal já possui animais cadastrados, que podem ser vistos em sua respectiva aba.

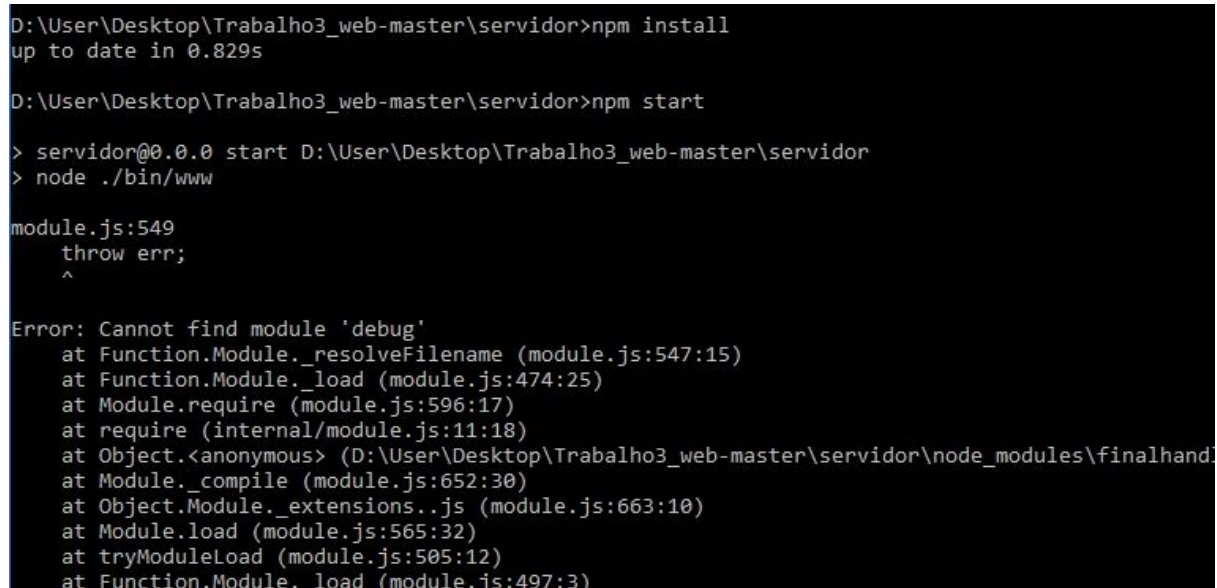
Caso deseje, o PAE pode criar novos usuários e administradores para mais testes;

Com todos os pré-requisitos preparados, é possível fazer uso da aplicação Web que utiliza uma arquitetura RESTful em seu *backend*.

Possíveis erros ao executar

Há alguns erros que, possivelmente, podem acontecer ao executar o trabalho - e aqui estão as soluções.

Caso, ao executar o *node.js*, receba essa mensagem de texto (ou similar) - presente na imagem a seguir - basta excluir a pasta *node_modules*, e executar o comando *npm install* na pasta do servidor.



```
D:\User\Desktop\Trabalho3_web-master\servidor>npm install
up to date in 0.829s

D:\User\Desktop\Trabalho3_web-master\servidor>npm start

> servidor@0.0.0 start D:\User\Desktop\Trabalho3_web-master\servidor
> node ./bin/www

module.js:549
    throw err;
    ^

Error: Cannot find module 'debug'
    at Function.Module._resolveFilename (module.js:547:15)
    at Function.Module._load (module.js:474:25)
    at Module.require (module.js:596:17)
    at require (internal/module.js:11:18)
    at Object.<anonymous> (D:\User\Desktop\Trabalho3_web-master\servidor\node_modules\finalhandl
    at Module._compile (module.js:652:30)
    at Object.Module._extensions..js (module.js:663:10)
    at Module.load (module.js:565:32)
    at tryModuleLoad (module.js:505:12)
    at Function.Module._load (module.js:497:3)
```

(Imagem 12)

É provável que outro erro aconteça, se o trabalho for executado novamente (acusando falta do *mongoose*), para solucionar-lo basta instalar o *mongoose* com o comando *npm install mongoose*.

Apenas esses erros foram encontrados durante a execução de testes do trabalho, funcionando em todos os computadores dos membros do grupo.

Erros segunda etapa

Após a correção da segunda etapa, alguns erros foram apontados pelo monitor PAE no feedback, entre eles:

1. Não verificação de e-mails válidos;
2. Falta de produtos e serviços previamente definidos;
3. Falta de oferecimento de horários para atendimento;
4. Opção do usuário de selecionar seus animais para atendimento;
5. Falta de uma carrinho de compras;
6. Não solicitação de dados financeiros no momento do pagamento;

Todos os itens acima foram arrumados, mesmo que - muitos desses - não estivessem, de fato, na especificação da segunda etapa do trabalho, pois a mesma foi feita de maneira muito simplória.

Todo o HTML foi tratado por validadores de HTML (<https://validator.w3.org/> , que foi sugerido pelo próprio monitor).

Fizemos uso do Bootstrap, para garantir responsividade de tela dentro da aplicação proposta.

No caso das adaptações, para validação do e-mail foi utilizado uma expressão regular regex. Foram definidos produtos logo de início, oferecemos 10 horários diferentes para cada dia, menos aos domingos, momento em que a reserva de serviços é impedida. Além disso, no momento da reserva existe um select, caso o usuário não esteja conectado, nada será exibido , caso contrário os animais do usuário serão mostrados.

O carrinho de compras foi criado e apenas usuários logados terão acesso.

Conclusão

A primeira etapa do trabalho consistia em construir uma aplicação fixa, sem muita interação e/ou tratamento de eventos. Para que o mockup da primeira etapa pudesse se tornar usável, adequando-se às tendências da internet moderna, houve o uso de ferramentas baseadas em Javascript.

A segunda etapa do trabalho consistia na construção de uma aplicação com um “servidor falso”, além de toda a “animação” da página por meio do Javascript (no nosso caso, JQuery), que faz o controle de toda a interação com o usuário. Em suma, toda a construção da SPA foi desafiadora, muito por conta da falta de familiaridade com a linguagem escolhida (JQuery), que, apesar de nova, detém uma curva de aprendizado muito alta, além disso houve a necessidade de aprendizado do framework Bootstrap.

Para a última etapa do projeto, utilizou-se do servidor Node.js e a base de dados NoSQL MongoDB para armazenamento.

Caso haja algum problema com o acesso do trabalho, na pasta zipada, é possível encontrá-lo no seguinte endereço do github: https://github.com/abe2602/Trabalho3_web.