PROJETO FINAL - APLICAÇÃO PET SHOP

SCC0219 - Introdução ao Desenvolvimento Web

Kelvin Guilherme de Oliveira - 9293286

Lucas Yudi Sugi - 9293251

Renan Rodrigues - 9278132

Este relatório tem como objetivo a explicação dos pontos principais do projeto desenvolvido nessa etapa da aplicação Pet Shop, que consiste da implementação do *Server-Side* para o projeto cujo *Client-Side* foi desenvolvido previamente. Esta etapa do projeto utilizou *Node.js* em conjunto do *Express.js* para a criação do servidor, e o *CouchDB* para armazenamento dos dados.

VISÃO GERAL

A estrutura de arquivos final seguiu o padrão das etapas 1 e 2 do projeto. A parte do *Client-Side* se encontra no diretório ***public***, que é fornecido estaticamente pelo servidor para o navegador. Nessa pasta, estão os arquivos *.js*, *.html*, *.css* e as imagens, sendo os mesmos da segunda etapa.

Para utilizar alguns recursos que foram movidos para o servidor (como a manipulação dos dados), várias adaptações foram realizadas nos arquivos *.js*, permitindo-os acessar os recursos com *HTTP Requests* (*POST e GET*). Além dessas modificações, algumas correções foram feitas, como a atualização do nome do usuário na barra de navegação, quando editado, porém alguns recursos não estão mais funcionando, como o *upload* de imagens.

SERVER SIDE

O *Server-Side* da aplicação foi construído com *Node.js* e *Express.js*, e os arquivos necessários se encontram no diretório ***server***, que são: *database.js*, que possui uma espécie de “API” para manipulação da base de dados do projeto, e o arquivo *index.js*, que tem os métodos necessários para criar e ativar o servidor (porta ***8081***), definir o diretório *public* como estático, e tratar os *HTTP Requests* que

são feitos pelo *Client-Side* de maneira adequada, chamando as funções que manipulam o banco de dados conforme necessário.

Para facilitar o tratamento dos *HTTP Requests*, utilizou-se as funções prontas do *Express.js*, e, sendo assim, trabalhamos apenas com *requests* do tipo *GET* e *POST*, não utilizando os métodos *PUT* e *DELETE*, tratando tais casos internamente com as funções da *API* criada para a manipulação do banco de dados.

DATABASE

Conforme mencionado anteriormente, o banco de dados utilizado pela aplicação é o ***CouchDB***, que possui uma facilidade muito grande de realização de consultas, por utilizar a linguagem *JavaScript* para fazê-las. Sendo assim, a única linguagem de programação utilizada no projeto como um todo foi *JavaScript*.

O banco de dados foi estruturado de forma similar às etapas anteriores, ou seja, praticamente toda a estrutura de objetos criados anteriormente pode ser reutilizada, como mostrado a seguir:

- Animal = {email:, name:, breed:, age:, picture:}
- CartPurchase = {email:, idProduct:, qtd:}
- Product = {name:, brand:, price:, vality:, stock:, description:, picture:, soldAmount:, priority:}
- Scheduling = {email:, day:, month:, hour:, idAnimal:, idService:}
- Service = {name:, price:, description:, picture:, scheduledAmount:, priority:}
- Solicitation = {email:,situation:, tracking:, day:, month:, value:}
- User = {email:, password:, name:, cpf:, tel:, adress:, district:, city:, state:, country:, adm:}

Com relação ao trabalho anterior a maior diferença é que agora em Scheduling é armazenado o idAnimal (antes era o seu nome), isso permitiu uma maior consistência nos dados assim como maior liberdade para o usuário(agora é possível cadastrar um animal com o mesmo nome).

Deve-se citar que para uma maior produtividade foi utilizado um módulo chamado *node-couchdb* que permite realizar a manipulação do banco de forma

mais simplificada, não sendo necessário realizar diretamente requisições http com get,post,update e delete.

Assim, com ele foi possível utilizar funções que realizam inserções, atualizações e deleção na base, sendo que o único problema é que não existe nada pronto para criar uma view, logo, sua criação foi a única em que se utilizou requisições http com o módulo http do node.

Ademais, é importante dizer que em nosso modelo uma requisição de um dado ocorre da seguinte forma: O usuário requisita alguma informação enviando a requisição ao servidor sendo ele o responsável por checar qual a operação e dado desejado. Isso será passado para o banco de dados que analisa o que deve ser entregue e realiza o envio da informação diretamente para o cliente, ou seja, em nosso modelo a resposta do usuário não passa pelo servidor.

EXECUÇÃO E TESTE

Antes da execução da aplicação, deve-se garantir que o *CouchDB* está instalado na máquina, e que o seu servidor esteja configurado na porta ***5984***. Além disso, recomenda-se que possíveis tabelas (documentos do *CouchDB*) existentes sejam previamente excluídas para evitar conflitos com os documentos gerados na criação da base.

Ao extrair o arquivo *.zip* que contém os arquivos da aplicação, deve-se entrar no diretório raiz, e executar o comando ***npm install***, que irá buscar pelos *node_modules* necessários para a execução da aplicação e disponibilizá-los. Vale ressaltar que a versão do *Node.js* utilizada nos testes foi a *v8.11.0*, sendo testado nos sistemas operacionais *Windows 10* e no *Ubuntu 16.04* com sucesso.

Após a instalação ser efetuada, pode-se executar a aplicação com o comando ***npm start***, que irá construir o banco de dados na primeira vez que for executado, e disponibilizar a aplicação na porta ***8081*** do *localhost*.

Pode-se acessar o sistema e verificar as funcionalidades do **cliente** utilizando o usuário *teste@gmail.com* (*senha: 123456*), e as funcionalidades da parte **administrativa** (vide observação abaixo) por meio do usuário *admin@gmail.com (senha: admin123)*.

*** Obs.:** Infelizmente devido a alguns problemas ao longo do projeto com o *CouchDB* e ao “temido” final de semestre, só conseguimos finalizar a primeira parte do setor administrativo, que permite o registro de novos Clientes, além da visualização e exclusão de agendamentos, não sendo disponibilizada a gestão dos Produtos e Serviços conforme as versões anteriores do projeto.