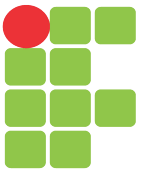


Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte (IFRN)
Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas (TADS)

LEONARDO NATÉRCIO LIBÂNIO LACERDA

RELATÓRIO DE ESTÁGIO EM DESENVOLVIMENTO DE SISTEMAS WEB NA EMPRESA SOIRTEC

Pau dos Ferros - RN
2018

LEONARDO NATÉRCIO LIBÂNIO LACERDA

RELATÓRIO DE ESTÁGIO EM DESENVOLVIMENTO EM SISTEMAS WEB NA EMPRESA SOIRTEC

Relatório de estágio apresentado ao curso superior de tecnologia em Análise e Desenvolvimento de Sistemas (TADS) do Instituto Federal de Educação, Ciência e Tecnologia (IFRN) como requisito para a obtenção do título de tecnólogo em análise e desenvolvimento de sistemas.

Orientador (a): Prof. Dr. Aluisio Igor Rêgo Fontes

Pau dos Ferros - RN
2018

RESUMO

O presente relatório descreve as atividades que foram desenvolvidas durante o período de estágio supervisionado, requisito para conclusão do curso de Tecnologia em Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte (IFRN), campus de Pau dos Ferros - RN. O estágio foi realizado num período de três meses na empresa Soirtec Tecnologia e Serviços de Informática LTDA - ME, e, teve como objetivo geral: desenvolver e manter, preventivamente, sistemas *web* propostos pela empresa, e como objetivos específicos: a) trabalhar com padrões de projetos; b) desenvolver módulos e componentes no projeto utilizando *frameworks* com a linguagem *JavaScript* e *TypeScript*; c) integrar os mesmos com o *Firebase* e uma API REST desenvolvida pela empresa e, d) realizar testes unitários utilizando o padrão BDD. Os objetivos foram cumpridos, ao desenvolver funcionalidades para um sistema de venda de carros e a implementação de um sistema de auditoria para um *software* de segurança, o que, ao término desta experiência, pôde-se obter um *feedback* positivo por parte dos clientes e da própria empresa.

Palavras-chave: Framework. JavaScript. Desenvolvimento Web.

SUMÁRIO

1 INTRODUÇÃO

Com o passar dos anos, as aplicações *web* evoluíram de simples *websites*, cujo o propósito era apenas a navegação, para verdadeiros sistemas de informação altamente complexos, com dados e transações realizadas imensuravelmente durante cada acesso voltado para a implementação de processos de negócios. Houve a necessidade de se adotar um processo de software sistemático que ajudasse a gerenciar o ciclo de vida de tais aplicações que vieram a surgir naturalmente durante todo o processo de desenvolvimento e implantação/produção.

Os padrões de sistemas baseados nos fundamentos do desenvolvimento web vem sofrendo diversas atualizações todos os dias com a criação de novos *frameworks* baseados na linguagem *JavaScript* (*Angular, VueJS e React*) utilizando o padrão da *ES6* (*EcmaScript 6* e *TypeScript*) e padrão de desenvolvimento *MVC* (*Model, View, Control*). Devido a esta diferença de novos recursos de atualizações diárias e de formas de acesso, é necessário e importante que os recursos tecnológicos empregados nestes sistemas e os *layouts* de páginas sejam adaptados à essas variações. Portanto, é considerável o estudo e o emprego de *frameworks* atuais no desenvolvimento web.

A prática do estágio foi realizada na empresa Soirtec Tecnologia e Serviços de Informática LTDA - ME, que fica situada na Rua João de Aquino, no Edifício Djalma de Freitas, Nº 101,na centro Pau dos Ferros/RN, próximo ao prédio da Justiça Federal; servindo como requisito avaliativo do componente curricular Trabalho de Conclusão de Curso, do curso Tecnólogo em Análise e Desenvolvimento de sistemas do Instituto Federal do Rio Grande do Norte, campus de Pau dos Ferros/RN. Teve como objetivo geral: desenvolver e manter sistemas *web* (capazes de evitar erros e falhas) propostos pela empresa, onde, o mesmo envolveu os seguintes objetivos específicos: a) trabalhar com padrões de projetos; b) desenvolver módulos e componentes no projeto utilizando *frameworks* com a linguagem *JavaScript* e *TypeScript*; c) integrar os mesmos com o *Firebase* e uma API REST desenvolvida pela empresa e, d) realizar testes unitários utilizando o padrão BDD.

O trabalho está organizado por tópicos, onde, num primeiro momento foi realizado um levantamento bibliográfico para serem descritas as ferramentas utilizadas; em seguida, o relato de como se deu a realização deste estágio, apresentando todos os pontos positivos e negativos que se fizeram presentes durante este processo de desenvolvimento das atividades propostas, e, por fim, as nossas considerações finais.

2 FERRAMENTAS UTILIZADAS NA PRÁTICA DO ESTÁGIO

2.1 A IDE *VSCODE* E SUAS FACILIDADES

O *VSCode* é uma *IDE* criada pela *Microsoft* e muito popular para desenvolver projetos com os principais *frameworks* em atividade/atuais.

A *IDE* vem com diversos *plugins* pré-instalados para compilar os códigos e destacar as sintaxes de acordo com cada tipo de linguagem que é utilizada na fase de desenvolvimento dos sistemas. Assim acontece com os *frameworks web*, o que facilita no trabalho do desenvolvedor diversos truques utilizados para melhorar o desempenho e a produção na hora de codificar e também de enviar os códigos para repositórios, que são as conhecidas extensões.

Para padronizar o código, por exemplo, podemos utilizar o *TSLint,* onde o mesmo varre toda a estrutura do projeto em busca de erros que podem ser acometidos por falta de percepção ou pelo simples fato de uma função não existir naquela linguagem que está sendo utilizada. Outra extensão bastante utilizada é a *vscode-icons,* para padronizar a estrutura de pastas e arquivos do seu projeto na *IDE*. Existem diversas extensões e *plugins* importantes que podem ser adicionados ao *VSCode* para facilitar o trabalho de desenvolvimento e melhorias na qualidade do código.

2.2 FRAMEWORKS BASEADOS NA LINGUAGEM JAVASCRIPT

O *JavaScript* vem se tornando uma linguagem bastante utilizada entre os desenvolvedores *front-ends*, desde o uso de seus *scripts* puros, para realizar uma

determinada ação no sistema, manipulando-os, sem nenhuma mudança em suas funções primárias; até o seu uso com bibliotecas prontas para facilitar o envolvimento do usuário com a linguagem, dispensando os grandes códigos.

Através do notável avanço das tecnologias e a facilidade do *JavaScript* acessar objetos e manipular estruturas dentro do *DOM* (*Document Object Model*), diversos *frameworks* foram desenvolvidos para facilitar ainda mais o desenvolvimento de sistemas *web*. Segundo Mattsson (1996 apud SILVA E LUCENA, 2000, p. 5):

> [...] um framework é um conjunto de classes que representam o design abstrato de uma família de problemas relacionados. Um framework é uma arquitetura projetada para possibilitar a máxima reutilização. Ele é representado por uma coleção de conjuntos abstratos e concretos de classes, encapsuladas de maneira que subclasses possam ser especializadas para uma dada aplicação.

Portanto, foram utilizados os *frameworks Angular 6, VueJS e React* em determinados projetos e manutenções para a empresa onde foi desenvolvido o estágio supervisionado. Mais abaixo iremos dissertar um pouco sobre cada *framework*.

Para a utilização destes *frameworks* deve-se antes dissertar sobre como e quando estes são inicializados após um projeto estar em andamento, e para isto, utiliza-se um servidor local para este serviço, chamado *Node.js*.

O *Node.js* trata-se de uma plataforma utilizada para o desenvolvimento de aplicações *server-side* que se baseiam em rede e utiliza como padrão o *JavaScript* e o *JavaScript Engine*, ou seja, com o *Node.js* podemos criar uma pluralidade de aplicações *web* utilizando apenas códigos em *JavaScript,* sendo este, ideal para o desenvolvimento de funcionalidades que trabalham em tempo real, onde pode haver uma troca intensa de dados através de dispositivos distribuídos. E, por este e outros motivos, os *frameworks CLI* (*comand-line interface*) dependem do *Node.js* para rodar/inicializar os projetos em um servidor local, automaticamente.

Para incrementar ainda mais e deixar as aplicações robustas utilizamos o NPM**,** como o principal gerenciador de pacotes, no qual facilita o trabalho dos

desenvolvedores usando *plugins* pré-compilados gerando módulos com bibliotecas prontas para serem utilizadas dentro do projeto.

2.2.1 *Angular*

O *Angular* trata-se de uma plataforma e *framework* para construção da interface de aplicações usando *HTML*, *CSS* e, principalmente, *JavaScript*, criada pelos desenvolvedores da *Google*. Dentre os principais elementos básicos do desenvolvimento web orientados a componentes, podemos destacar os componentes, *templates*, diretivas, roteamento, módulos, serviços, injeção de dependências e ferramentas de infraestrutura que automatizam tarefas, como a de executar os testes unitários de uma aplicação.

Desta forma, o Angular nos ajuda a criar *Single-Page Applications* com uma melhor qualidade e produtividade.

Como linguagem no desenvolvimento de aplicações em *Angular*, podemos utilizar *JavaScript*, *TypeScript* ou *Dart*. Fica a critério do programador e da necessidade do projeto a ser desenvolvido.

2.2.2 *VueJs*

Criado por *Evan You*, que trabalhou em vários projetos do *Angular.js* durante o tempo que trabalhou no *Google*. Segundo a *State of JS Survey* (2018) o *Vue.js* é uma contraparte leve do *Angular.js* e também está provando ser uma alternativa atraente para ele, e se tornou o projeto *GitHub* de *front-end* mais popular em 2018.

Uma das principais razões pela qual os desenvolvedores, ultimamente, priorizam o *Vue,* é devido a sua estrutura progressiva. Isso significa que ele se adapta facilmente às necessidades do desenvolvedor, começando de 3 linhas até o gerenciamento de toda a sua camada de visualização. O *VueJS* pode ser rapidamente integrado em um aplicativo por meio da tag "*script*", onde, gradualmente, começa a explorar o espaço.

Mas, o *Vue* funciona especialmente porque une as melhores escolhas e possibilidades que os *frameworks* como *Angular.js*, *React.js* e *Knockout* fornecem,

tornando-se a melhor versão de todos os *frameworks* juntos em um pacote limpo e organizado.

2.2.3 *React*

Segundo a *State of JS Survey* (2018) o *React* é outra biblioteca *JavaScript* popular que alimenta o *Facebook*. Desenvolvido e aberto pelo *Facebook* em 2013, rapidamente alcançou destaque para o desenvolvimento de grandes aplicativos da *web* que envolvem o processamento dinâmico de dados.

A *State of JS Survey* (2018) ainda destaca que, um dos principais casos em que o React encontra uso extensivo é quando os desenvolvedores precisam desmembrar códigos complexos e reutilizá-los em um ambiente livre de erros que tem a capacidade de manipular dados em tempo real. Elementos relacionados à ganchos do ciclo de vida, como decoradores, ajudam a melhorar a experiência do usuário.

O *React* mostra a simplicidade em termos de sua sintaxe que envolve muitas habilidades de escrita em HTML. Ao contrário do *Angular*, que requer uma curva de aprendizado muito grande para ser proficiente em *TypeScript*, a dependência do *React* em HTML o torna uma das bibliotecas *JavaScript* mais fáceis de usar.

2.3 *FIREBASE*

O *Firebase* é um serviço em nuvem muito utilizado por desenvolvedores de aplicações móveis e trata-se de um *back-end* completo também para aplicações *mobile* (*Android e iOS*) e aplicações *web*, que contém um visual limpo e de uma usabilidade simples, o *Firebase* é uma plataforma dedicada e contém *SDK* para a construção de aplicativos. Atualmente, o serviço suporta o desenvolvimento de aplicações nas linguagens de programação *Java, Node.js, C++, Javascript, Objective-C e Swift*.

A plataforma possui diversas funcionalidades que podem deixar o trabalho de desenvolvimento mais simples. Tais como as funcionalidades do *Cloud Firestore* e também o *Realtime Database* que podem ser utilizadas para propósitos iguais ou não, depende muita da estrutura de dados com que você deseja trabalhar. Pode usar ambos os bancos de dados no mesmo *app* ou projeto do *Firebase*. Esses bancos de dados *NoSQL* podem armazenar os mesmos tipos de dados, e as bibliotecas de cliente funcionam de maneira semelhante.

2.3.1 *Cloud Firestore*

O *Cloud Firestore* é o novo banco de dados principal do *Firebase* para o desenvolvimento de aplicativos para dispositivos móveis. Ele oferece resultados ainda melhores que o *Realtime Database* com um novo modelo de dados mais intuitivo. O *Cloud Firestore* também tem consultas mais avançadas e rápidas e melhor escalabilidade que o *Realtime Database*. Nele armazenam-se dados em documentos organizados em coleções como: dados simples são fáceis de armazenar em documentos, que são muito semelhantes aos da árvore *JSON*, dados complexos e hierárquicos são mais fáceis de organizar em escala, utilizando subcoleções dentro dos documentos e também requer menos desnormalização e desdobramento de dados.

2.3.2 *Realtime Database*

É o banco de dados original do *Firebase*. Trata-se ainda de uma solução eficiente e de baixa latência para aplicativos móveis que exigem estados sincronizados entre clientes em tempo real. Este também armazena dados como uma grande árvore *JSON*: dados simples são muito fáceis de armazenar e dados complexos e hierárquicos são mais difíceis de organizar em escala.

2.4 TESTES EM BDD (*BEHAVIOR DRIVEN DEVELOPMENT*)

Esta metodologia é útil em projetos de software ágeis, que são construídos em várias iterações e estão sofrendo alterações ao longo do seu ciclo de vida. Quanto maior o projeto, mais difícil será a comunicação. Segundo MORAES (2016) o BDD envolve práticas ágeis e foi projetado para torná-las mais acessíveis e eficazes para as equipes ágeis na entrega de um software. Trata-se de uma abordagem que funciona muito bem com uma metodologia ágil, encorajando desenvolvedores, pessoas de qualidade, não técnicas e de negócios em um projeto de software. O foco em *BDD* é a linguagem e as interações usadas no processo de desenvolvimento de software. Desenvolvedores que se beneficiam destas técnicas escrevem os testes em sua língua nativa em combinação com a linguagem ubíqua (*Ubiquitous Language*). Isso permite que eles foquem em por que o código deve ser criado, ao invés de detalhes técnicos, e ainda possibilita uma comunicação eficiente entre as equipes de desenvolvimento e testes.

Segundo Duarte e Fernandes (2016) o BDD é uma metodologia "outside-in". Ele começa do lado de fora identificando os resultados de negócios e, em seguida, detalha o conjunto de recursos que atingirá esses resultados. Cada recurso é capturado como uma “história”, que define o escopo do recurso junto com seus critérios de aceitação.

Diante do pressuposto, a ferramenta utilizada para a realização de testes guiados a comportamento foi o *Jest*. A seguir abordaremos sobre as facilidades e melhorias que o projeto adquire após a utilização da ferramenta nos testes.

2.4.1 *Jest* - Testes simplificados

Sempre foi complicado escrever testes para *front-end* devido a pouca experiência com projetos voltados a grandes empresas de *softwares*. Bem como, não existia um padrão ou uma ferramenta ideal e simples para se configurar e realizar estes testes. E, normalmente, testes para *front-end* são deixados de lado pela falta de clareza de como serão feitos, ou até mesmo pelas empresas pouco cobrarem de seus encarregados.

O JavaScript tem uma variedade de bibliotecas de testes que contém vários recursos excelentes. Essas bibliotecas ajudam à organizarmos nosso conjunto de

testes de maneira robusta e sustentável. Muitas dessas bibliotecas realizam o mesmo domínio de tarefas, mas com diferentes abordagens.

Existem muitas ferramentas para testes voltadas a linguagem *JavaScript* e elas sempre estiveram disponíveis na *internet*, como o *QUnit, Jasmine, Mocha, Karma* e algumas outras.

Averiguando de perto, estas ferramentas até conseguem resolver um determinado problema, mas, existem muitas aplicações *web* que são complexas, com inúmeros e distintos meios de chegar a mesma resposta. E para juntar todas estas ferramentas e colocar para trabalhar todas juntas levaria muito tempo e não ficaria compatível para a produção do projeto.

Sendo assim, o *Jest* torna-se rápido e prático, com opções de escolher seus testes manipulando-os através de expressões regulares ou o próprio nome do arquivo em questão. Além de tudo, tem um ótimo suporte para *stack trace* e o *source map* simples e prático de entender.

3 RELATO DE EXPERIÊNCIA

3.1 Caracterização da empresa e objetivos do estágio

O estágio referente ao Trabalho de Conclusão de Curso (TCC) do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, do Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte (IFRN), teve como campo de atuação a empresa Soirtec Tecnologia e Serviços de Informática LTDA - ME, que fica situada na Rua João de Aquino, no Edifício Djalma de Freitas, Nº 101,na centro Pau dos Ferros/RN, próximo ao prédio da Justiça Federal. A empresa é uma filial da matriz localizada em São Bernardo do Campo - SP, desde 2015 vem atuando com o desenvolvimento de sistemas corporativos, sites e aplicativos para empresas voltadas ao ramo de fintechs, mas, também presta serviços para startups de diversas áreas.

A Soirtec vem firmando espaço em Pau dos Ferros/RN, buscando uma melhor acomodação e bem-estar de seus desenvolvedores e, consequentemente, interação para que haja um significativo trabalho em equipe. A filial conta com uma

sala para acomodar até oito desenvolvedores, além de um cômodo para descanso e realizar refeições, um banheiro e uma pequena copa.

O fluxo de trabalho da empresa é baseado na metodologia de desenvolvimento ágil SCRUM, com jornada de trabalho realizada de segunda à sexta, e *daily's* diárias às 10h (horário de Brasília), onde são discutidas as atividades que foram realizadas no dia anterior e o que será feito no dia atual, e, quinzenalmente, há o fechamento das *sprints* dos projetos, para atualizar os ambientes de desenvolvimento, onde o que foi trabalhado no ambiente *dev* passa para o ambiente *staging* e posteriormente para a "produção", tendo assim, total controle sobre cada processo de desenvolvimento dos *softwares*. A cada trinta dias de trabalho se realiza uma reunião individual avaliativa, onde são levantados os pontos positivos e negativos do desenvolvedor e,a partir desta avaliação, elencar possibilidades de melhoramentos. A cada três meses, a depender do desempenho e progressão do funcionário da empresa, poderá, ou não, ser promovido a um nível mais elevado.

Para alcançar o objetivo geral de desenvolver e manter sistemas *web (*capazes de evitar erros e falhas) propostos pela empresa, o estágio se deu a partir dos seguintes objetivos específicos: trabalhar com padrões de projetos, desenvolvendo módulos e componentes no projeto utilizando *frameworks* com a linguagem *JavaScript* e *TypeScript*, integrando os mesmos com o *Firebase* e com uma *API REST* desenvolvida pela empresa. Realizou-se testes utilizando o padrão *BDD* com a ferramenta *Jest*, utilização do gerenciador de projetos *Trello* para garantir qualidade e desempenho das entregas juntamente com a implementação da metodologia ágil *SCRUM*, onde, desenvolvemos boas práticas de como melhorar o desempenho na produção de códigos e gerenciar atividades para agilizar o processo de desenvolvimento, utilizando-se, também, o *Gitlab* para controle de versão destas atividades.

3.2 Atividades desenvolvidas

3.2.1 Desenvolvimento e Manutenção de Sistema de Venda de Carros

O desenvolvimento e manutenção do sistema de venda de carros, que já era um projeto que vinha sendo desenvolvido para um dos clientes da empresa, e já contava com diversas funções e páginas elaboradas pela equipe de funcionários. Porém, havia algumas funcionalidades para serem implementadas, que seriam um *chat* para a troca de mensagens, uma listagem de carros com paginação, onde os carros cadastrados vinham de um banco de dados através de uma chamada via *GET* da *API* e também outras páginas individuais com os detalhes e informações de cada veículo para venda juntamente com os dados de contatos para facilitar o envolvimento do cliente/comprador com o vendedor.

A seguir iremos detalhar um pouco mais sobre cada funcionalidade que foi implementada no sistema para que fosse melhorado de forma que atendesse o público alvo, também sempre visando os padrões de responsividade e usabilidade em diversos modelos de dispositivos (*desktop e mobile*).

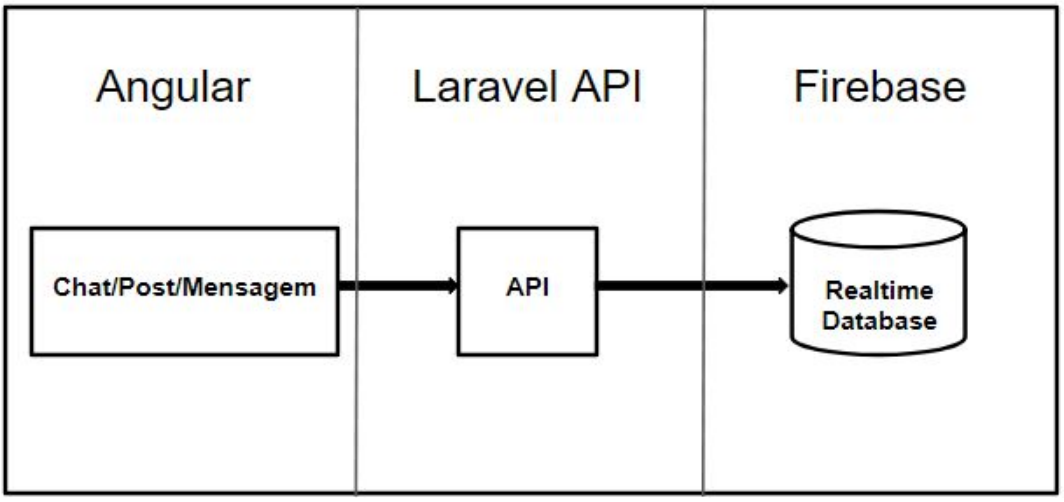
O *chat* foi uma das primeiras atualizações necessárias para manter a comunicação de um vendedor com um comprador. O *chat* desenvolvido tinha, também, uma série de regras baseadas em favoritar um veículo para criar uma conexão entre dois ou mais usuários. Criando conexões diretas, triangulares e compra/venda.

- Conexões diretas: uma conexão entre dois indivíduos, onde um favoritou o veículo de outro;
- Conexões triangulares: é uma conexão entre 3 indivíduos, onde os 3 favoritaram outros 2 anúncios dos seus semelhantes;
- Conexões compra/venda: onde um usuário somente favorita um anúncio e é aberto um *chat* simples entre ele e o vendedor do veículo;

Cada *chat* tem tipos de ligações que são estabelecidas durante a ação favoritar de cada anúncio, e existe uma validação feita pelo *back-end* na *API* que identifica quem favoritou o anúncio de quem, criando, logo após, um tipo específico de *chat*, sendo ele do tipo direto, compra/venda ou triangular como foram especificados acima.

Para o envio e recebimento de mensagens nesta funcionalidade utilizamos o *Realtime Database* para fazer toda a sincronização e troca de dados em tempo real. Para estabelecer esta conexão no *Angular*, nós utilizamos um pacote do próprio *core* do projeto, onde assim, pudemos criar uma ponte com o banco de dados facilitando ainda mais o trabalho. Desta forma, o *chat* foi desenvolvido e funciona perfeitamente de ambos os tipos supracitados. Na figura abaixo podemos observar como é realizado o *POST* de uma mensagem em nosso banco de dados no *Firebase*.

Figura 1 - Esquema de envio de *POST* das mensagens do *chat*, que passa por um *endpoint* na API onde são tratados os dados e enviados para o *Realtime Database*.

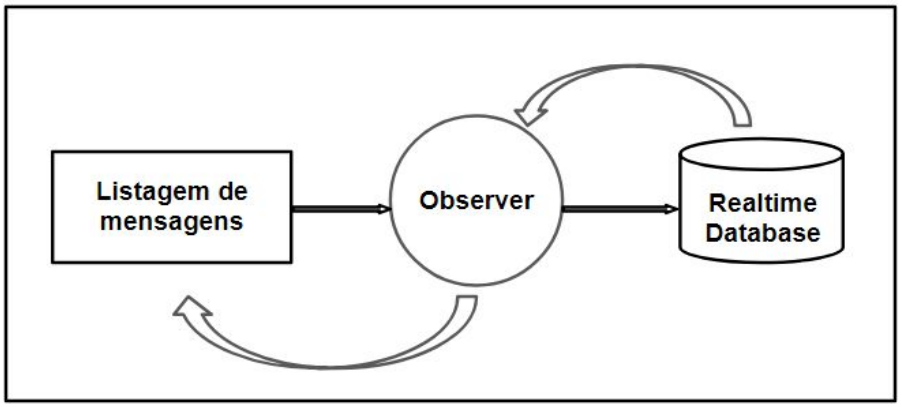


Fonte: elaborada pelo autor

A listagem das mensagens recebidas é realizada através de uma consulta no *Realtime Database*, em que a busca é inicializada no carregamento da página, e posteriormente, uma variável declarada como um *array,* e, sendo ela um *Observer,* fica observando o comportamento do banco, e, caso chegue uma nova mensagem, a listagem é rapidamente atualizada, tornando o *chat* em uma

aplicação *realtime*. Logo abaixo na figura 2 mostraremos o funcionamento desta implementação.

Figura 2 - O *array* de mensagens é um *observer,* que fica verificando no *Realtime Database* se tem mensagem nova, se sim, ele atualiza o *array* de mensagens.



Fonte: elaborada pelo autor

Uma das outras funcionalidades foi desenvolver uma página com a listagem de carros juntamente com um filtro de pesquisa que detalha toda a busca por um veículo específico.

Para desenvolver a página com listagem, foi utilizado o laço comum do *Angular* que é o **ngFor* e, para a paginação, um módulo (*pagination-controls*) que pode ser baixado utilizando o gerenciador de pacotes do *Node*, o *NPM*.

Para o filtro de pesquisa utilizamos um *JSON* pré-configurado e funções de buscas aninhadas dentro do código desenvolvido em *TypeScript* do componente de busca, fazendo assim uma busca assíncrona na página sem dar um *refresh* para carregar o conteúdo pesquisado.

Figura 3 - Arquivo *JSON* utilizando para o filtro de busca.

```
[
  {
    "carro" :
    [
      {
        "title" : "Blindagem",
        "options" :  [
          {
            "name": "Blindagem nível 1",
            "control": "optionalCar_1"
          },
          {
            "name": "Blindagem nível 2",
            "control": "optionalCar_2"
          },
          {
            "name": "Blindagem nível 3",
            "control": "optionalCar_3"
```

Fonte: elaborada pelo autor

Uma última particularidade inserida no sistema de venda de carros foi a página de detalhes individual de cada anúncio. Nela há todas as informações do veículo, juntamente com uma nuvem de *tags* (funcionalidade elaborada por mim) que foi inserida para facilitar no visual do *layout*, um carrossel de fotos do veículo anunciado, além de, uma função que foi implementada, também, para mostrar as informações do vendedor logo após o comprador favoritar aquele anúncio, mostra, ainda, demais informações do anúncio em geral, todas estas informações são recebidas através de chamadas na *API REST* do sistema.

Figura 4 - Nuvem de *tags* e demais informações do sobre o veículo.

Fonte: elaborada pelo autor

3.2.2 Auditoria de Sistema de Segurança

O desenvolvimento da auditoria de um sistema de segurança, deu-se a partir da criação de uma listagem dos recursos que são obtidos após o dispositivo mobile ser utilizado, gerando assim, informações que são gravadas em um banco de dados e, enfim, listados neste sistema *web*. Todos os módulos desta parte do sistema são criados em forma de componentes que podem ser inseridos em qualquer outro projeto, dando liberdade para o reuso do código.

Para este projeto, foi utilizado o *Cloud Firestore* para fazer as chamadas de dados (*gets*) e apresentar na camada de visualização todas as informações recebidas pelo banco de dados do *Firebase*.

Toda e qualquer mudança no banco de dados *Cloud Firestore*, o sistema recebe no *front-end*, tornando-se assim, um sistema *realtime*.

Neste mesmo sistema de auditoria, o usuário que tem todo o controle sobre os demais dispositivos cadastrados no sistema, pode filtrar entre datas quanto tempo ele deseja receber dados vindos destes dispositivos para gerar mais informações na auditoria.

3.2.2 Testes automatizados com *JEST*

Durante o desenvolvimento das funcionalidades, o grupo de desenvolvimento trabalha fazendo *reviews* a todo momento e buscando possíveis erros nos sistemas, para manter um código de qualidade e um sistema estável para o cliente. A ferramenta que foi utilizada para os testes de funcionalidades durante a fase final deste estágio foi o *JEST*, ferramenta de simples manuseio e fácil de manter.

Pela facilidade de uso, o *JEST* foi escolhido também por ter diversos pontos positivos, como: testes paralelos, cobertura de código e *sandbox* (para impedir que variáveis globais afetem testes posteriores).

Foram realizados testes, com *login*, rotas, funções diversas dos componentes e também na parte de *storage* (armazenamento local) dos

navegadores, como mostra a figura abaixo, onde testamos o “capturar, inserir e remover” itens do localStorage.

Figura 5 - Testando itens do localStorage com mock.

```
const mock = () => {
  let storage = {};
  return {
    getItem: key => key in storage ? storage[key] : null,
    setItem: (key, value) => storage[key] = value || '',
    removeItem: key => delete storage[key],
    clear: () => storage = {},
  };
};

Object.defineProperty(window, 'localStorage', {value: mock()});
Object.defineProperty(window, 'sessionStorage', {value: mock()});
Object.defineProperty(window, 'getComputedStyle', {
```

Fonte: elaborada pelo autor

Alguns testes seguiram o padrão BDD (Desenvolvimento guiado a testes), onde a partir dos testes realizados, iríamos implementando as seguintes situações de comportamento das funcionalidades.

4 CONCLUSÃO

O período de estágio supervisionado na empresa Soirtec, foi primordial para o aperfeiçoamento do trabalho e desempenho de habilidades que foram construídas ao longo da trajetória acadêmica, tanto na parte de solução problemas com lógicas e/ou algoritmos, como na prática de implementação e codificação com novos recursos do desenvolvimento web que se tem disponíveis atualmente, novas plataformas, novas *IDE’s*, novos *frameworks*, novos bancos de dados e tantas outras ferramentas que vêm para somar e facilitar o trabalho dos desenvolvedores de *software*, seja *web* ou *mobile*.

Foi significativo trabalhar com recursos e ferramentas estudados anteriormente em alguns componentes curriculares da academia, pondo em prática essas aprendizagens, tendo uma visão de como realmente funciona em um espaço

onde há o contato direto com cobranças rígidas de clientes e contratação de serviços reais.

Durante todo o processo do estágio foram utilizadas ferramentas que pudessem ser viáveis para cada propósito dos projetos e que facilitassem também a vida útil destes *softwares*.

A escolha da IDE para o desenvolvimento foi fulcral para o projeto, bem como a escolha da linguagem *web*, da arquitetura do projeto, da metodologia de desenvolvimento utilizada, e outras especificidades. Tudo isso, engloba a padronização e estrutura de códigos utilizados pela empresa durante todo o processo de criação dos projetos.

Durante o desenvolvimento dos projetos da empresa, os erros que iam sendo cometidos, seja por falta de experiência ou não, a equipe de funcionários dispôs de assistência e orientações adequadas para a resolução dos erros e assim obter uma melhor qualidade dos códigos. Este suporte e o trabalho em equipe permitiu uma considerável aprendizagem com sujeitos mais experientes da área, contribuindo para a construção de novos conhecimentos relacionados aos conhecimentos produzidos durante a graduação, permitindo, assim, ao término da experiência do estágio uma maior bagagem de habilidades para uma carreira profissional promissora.

Quando se tem um mercado de trabalho bastante competitivo, ter a oportunidade de praticar as habilidades adquiridas no curso em uma instituição da área de tecnologia em uma prática de estágio, ampliam-se as visões e as ambições desse mercado. É uma experiência que permite um crescimento profissional e pessoal.

Todas as atividades propostas foram desenvolvidas dentro do prazo pré-estabelecido pelo orientador viabilizando alcançar os objetivos propostos para a realização do estágio de desenvolver e manter, preventivamente, sistemas *web* propostos pela empresa, o que, ao término desta experiência, pôde-se obter um *feedback* positivo. Dessa maneira saiu como previsto para a conclusão de mais esta etapa para garantir a formação do curso Tecnólogo em Análise e Desenvolvimento de Sistemas.

REFERÊNCIAS

SILVA, V. T.; LUCENA, C. J. P. **ContentNet: um framework para interoperabilidade de conteúdos educacionais utilizando a plataforma IMS**. Disponível em: <http://www.lbd.dcc.ufmg.br/colecoes/rbie/6/1/003.pdf>. Acesso em: 23 dez. 2018.

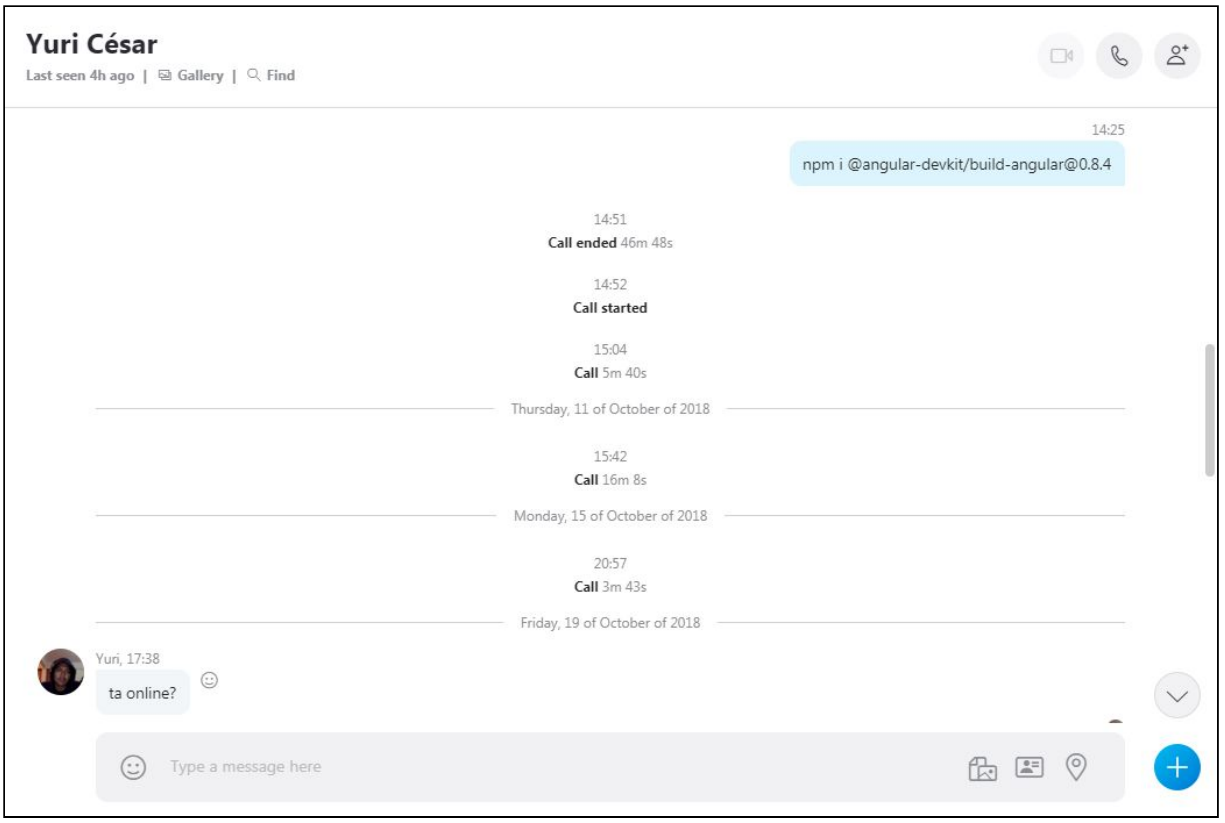
FRONT-END Frameworks - Overview. Disponível em: <https://2018.stateofjs.com/front-end-frameworks/overview>. Acesso em: 23 dez. 2018.

MORAES, L. C. P. **Um estudo empírico sobre o uso do BDD e seu apoio a Engenharia de Requisitos**. 2016. 139 p. Dissertação (Programa de Pós-graduação em Ciência da Computação) - Faculdade de Informática, Pontifícia Universidade Católica do Rio Grande do Sul, Porto Alegre, 2016. Disponível em: <http://tede2.pucrs.br/tede2/bitstream/tede/7043/2/DIS_LAURIANE_CORREA_PEREIRA_MORAES_COMPLETO.pdf>. Acesso em: 26 dez. 2018.

DUARTE, César; FERNANDES, Amílcar. **Behavior-Driven Development**. 2016. 4 p. Mestrado Integrado (Engenharia Informática e Computação) - Faculdade de Engenharia, Universidade do Porto, Porto - PORTUGAL, 2016. Disponível em: <https://paginas.fe.up.pt/~ei06089/files/Behavior-Driven%20Development.pdf>. Acesso em: 26 dez. 2018.

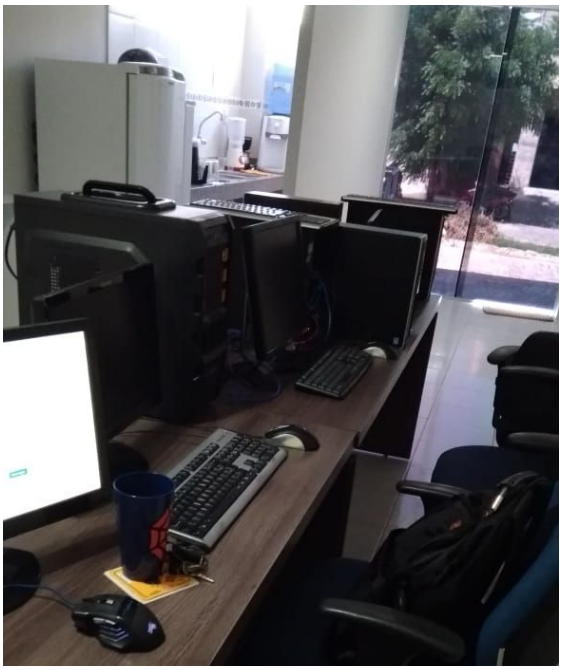
ANEXOS

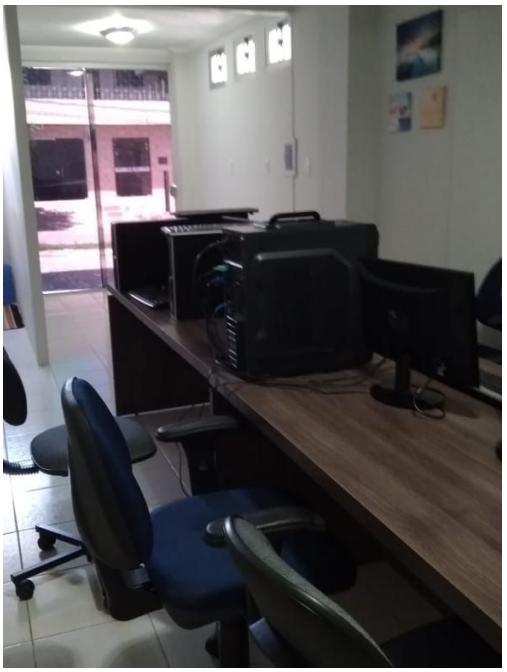
ANEXO A - Reuniões no Skype com o supervisor Yuri César



ANEXO B - Foto da frente da filial da Soirtec em Pau dos Ferros/RN

ANEXO C - Fotos da Sala de trabalho





ANEXO D - Foto da Copa

ANEXO E - Fotos do cômodo para descanso e refeição