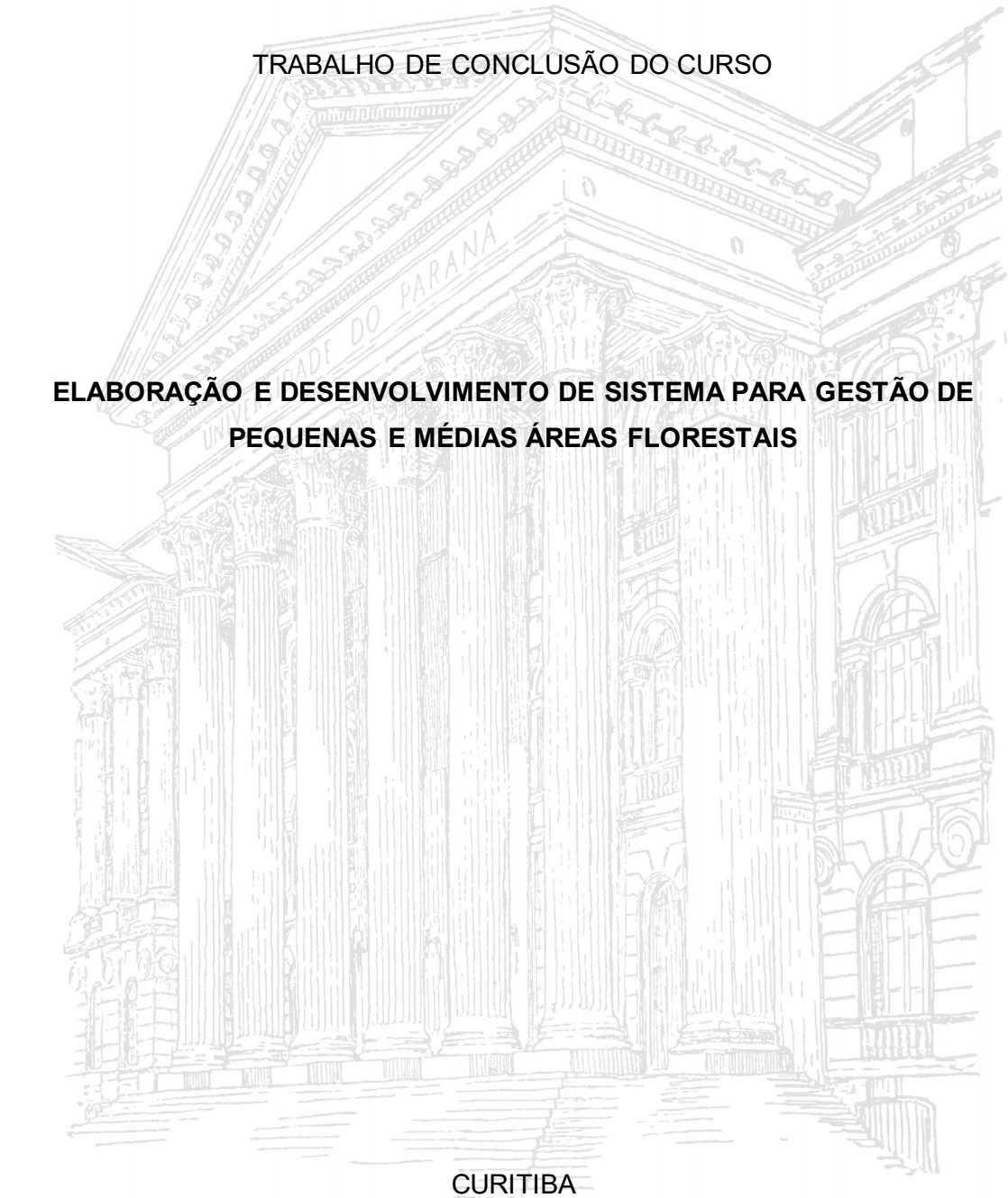
UNIVERSIDADE FEDERAL DO PARANÁ
SETOR DE CIÊNCIAS AGRÁRIAS
CURSO DE ENGENHARIA FLORESTAL

VÍTOR CARDOSO BERTOLUCCI

TRABALHO DE CONCLUSÃO DO CURSO



ELABORAÇÃO E DESENVOLVIMENTO DE SISTEMA PARA GESTÃO DE PEQUENAS E MÉDIAS ÁREAS FLORESTAIS

CURITIBA
2019

VÍTOR CARDOSO BERTOLUCCI

ELABORAÇÃO E DESENVOLVIMENTO DE SISTEMA DE GESTÃO PARA PEQUENAS E MÉDIAS ÁREAS FLORESTAIS

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia Florestal, Setor de Ciências Agrárias, Universidade Federal do Paraná, como requisito para a conclusão da disciplina ENGF010 e requisito parcial para obtenção do título de Engenheiro Florestal.

Orientador: Prof. Dr. Julio Eduardo Arce.

CURITIBA

2019

AGRADECIMENTOS

A Deus, por tudo que ele me proporciona e oferece.

À minha família, em especial aos meus pais Claudio Augusto Bertolucci e Claudia Tucunduva Cardoso Bertolucci, e ao meu irmão Gabriel Cardoso Bertolucci, pelo apoio, carinho e paciência durante toda a graduação.

À Caroline Papaléo Siqueira pelo apoio e inspiração durante os últimos anos da graduação.

Ao meu orientador Julio Eduardo Arce pela prontidão e ensinamentos que me proporcionou ao longo do curso e durante a realização do Trabalho de Conclusão de Curso e do Estágio Obrigatório.

A Marcos Bailo, Vanderson Fernandes e Richard Respondovesk pela oportunidade de aprendizado e desenvolvimento que possibilitaram a realização do Trabalho de Conclusão de Curso.

Aos meus amigos Alexandre Baumel dos Santos, Winicius Augusto Schaeffer, César Luizon Padilha, Jean Carlos Ribeiro, Fabio Cordeiro de Brito, João Felipe Martins, Thiago Garcia de Almeida e Marcelo Luvizotto.

Aos meus colegas do curso de Engenharia Florestal da Universidade Federal do Paraná.

Aos colegas de profissão de Engenharia Florestal.

Aos professores do curso de Engenharia Florestal da Universidade Federal do Paraná.

DADOS DO ACADÊMICO

Nome do aluno: Vítor Cardoso Bertolucci

GRR: 20152431

RESUMO

As mudanças no setor produtivo desencadeadas pela evolução tecnológica trouxeram um aumento na competitividade do mercado em geral, trazendo a necessidade de constante evolução e aperfeiçoamento das atividades produtivas. Uma das conhecidas formas de otimizar a produção de uma empresa é através do controle dos consumos, despesas e operações realizadas. Nas empresas de base florestal, costuma-se realizar estes tipos de controle com fichas de campo impressas nas quais as informações são inseridas e, posteriormente, digitadas em planilha eletrônica ou simplesmente armazenadas para consultas. O presente trabalho teve como objetivo digitalizar o processo de coleta de dados em campo e armazenar os dados coletados em uma base de dados, através do desenvolvimento de um sistema computadorizado. O sistema foi desenvolvido com uma versão *web* - onde são realizados os cadastros da empresa, os boletins de campo e as consultas das informações gerenciais produzidas – e uma versão *mobile*, por meio da qual são realizados os apontamentos em campo. Os módulos de Cadastro Florestal e Silvicultura encontram-se em pleno funcionamento e continuam sendo desenvolvidos, abrindo um leque de possibilidades para desenvolvimento futuro de outros módulos importantes ao processo florestal e integrações entre eles.

Palavras chaves: Cadastro Florestal. Silvicultura. Controle. Digitalização. Floresta 4.0.

ABSTRACT

The changes in the productive sector triggered by technological evolution brought an increase in the market competitiveness in general, bringing the need for constant evolution and improvement of productive activities. One of the well-known ways to optimize a company's production is through control of consumptions, expenses and operations performed. In forest-based companies, these types of control are usually performed with printed field sheets in which information is entered and then typed into a spreadsheet or simply stored for consultation. The present work aimed to digitize the field data collection process and store the collected data in a database, through the development of a computerized system. The system was developed with a web version - where the company information, the field bulletins and the consultation of produced management information are registered - and a mobile version, through which the field notes are made. The Silviculture and Forestry Registration modules are fully operational and are still being developed, opening up a range of possibilities for future development of other important modules to the forestry process and integrations between them.

Keywords: Forestry Registration. Silviculture. Control. Digitization. Forest 4.0.

SUMÁRIO

LISTA DE FIGURAS

1 INTRODUÇÃO

A coleta e o armazenamento de dados de uma empresa são cruciais para se exercer o controle de suas atividades, desde o consumo de insumos, a utilização de materiais e serviços realizados, até o controle da qualidade e de nível gerencial, onde os dados são usados para análises de eficiência, produtividade, custos e de estratégias a serem seguidas, servindo como alicerce para a tomada de decisão.

Para que qualquer atividade econômica seja rentável, ela deverá possuir um estilo de gestão compatível com suas características organizacionais para que esta estrutura possa garantir padrões de competitividade dentro da indústria na qual ela atua. A eficiência de uma administração dentro de qualquer negócio depende, entre vários fatores, de um suporte capaz de prover informações contábeis relevantes para as diversas decisões gerenciais, atualizando de maneira sistemática os diversos usuários destas informações (CALLADO, 1999).

A coleta de dados é a base para qualquer tipo de controle de atividades e, portanto, é importante que ela seja feita de maneira organizada e padronizada, de modo a minimizar a chance de registro de dados incorretos. Uma maneira de se atingir isto é por meio de sistemas que forneçam campos de entrada de dados com bloqueio a caracteres desnecessários, validação dos dados informados e, sempre que possível, repostas pré-definidas em listas. Este modelo, além de evitar erros, também evita que a mesma informação seja registrada diversas vezes com nomenclaturas diferentes, o que impossibilita o agrupamento das informações para análises.

As grandes empresas, em quase todos os casos, fazem uso de ERPs (*Enterprise Resource Planning*), que são grandes sistemas para gerenciamento e gestão de tudo que acontece nelas. Estes sistemas atendem perfeitamente a necessidade de uma administração de custos, controle fiscal e controle de estoque, porém, não são específicos para empresas florestais, deixando algumas lacunas importantes que devem ser preenchidos por outros sistemas. Para preencher estas lacunas, algumas empresas recorrem aos sistemas de gestão florestal, que são preparados para gerir o cadastro florestal e os processos florestais como silvicultura, colheita e inventário, além de, em alguns casos, ter comunicação direta com o ERP da empresa, unificando as informações.

Os sistemas citados compõem uma solução para o controle e a gestão dos processos florestais, porém, eles são de custo elevado e, portanto, inacessíveis para muitas empresas. Por não terem condições de adquirir estes sistemas, as pequenas empresas buscam outras maneiras de fazer seu controle, muitas vezes consistindo no uso de fichas impressas de campo e de planilhas eletrônicas. Neste modelo, são registradas nas fichas de campo as atividades diárias, com informações como: tipo e quantidade de insumos utilizados, equipamentos utilizados, área trabalhada, denominação da área trabalhada (em muitos casos as áreas são divididas até o nível de talhões e estes possuem um número ou um nome), equipe que realizou o serviço, dentre outras. Estas fichas são então enviadas para um escritório onde são digitadas em planilhas eletrônicas que atuam como um banco de dados.

Este modelo funciona, porém, apresenta importantes pontos negativos, como: grande dispêndio de esforço para ler, interpretar e digitar as informações recebidas; suscetibilidade ao registro de informações imprecisas ou incorretas, uma vez que são escritas manualmente; dificuldade de consumo das informações armazenadas, sendo necessário localizar e ler todas as planilhas salvas. Colocados estes pontos, pode-se concluir que há espaço para melhoria através da adoção de sistemas informatizados para substituir estes processos de coleta e armazenamento de dados.

2 OBJETIVOS

2.1 OBJETIVO GERAL

Elaborar e desenvolver um sistema informatizado para coleta de dados e gestão florestal acessível às pequenas empresas, visando auxiliá-las a reduzir o esforço despendido com a coleta e armazenamento de dados, além de reduzir o erro no registro de informações e simplificar a consulta e análise dos dados produzidos pela empresa.

2.2 OBJETIVOS ESPECÍFICOS

- Desenvolver uma versão *web* do sistema para gerenciar os processos de Cadastro Florestal e Silvicultura, além de receber, armazenar e organizar as informações obtidas em campo.
- Desenvolver uma versão *mobile* do sistema para realizar a coleta de dados em campo através de formulários, que funcione com ou sem acesso à internet e envie as informações para a versão *web*.
- Desenvolver uma API (*Application Programming Interface)* para gerenciar a comunicação entre as versões *web* e *mobile* com o banco de dados do sistema.

3 MATERIAL E MÉTODOS

A proposta do trabalho é elaborar e desenvolver um sistema simples para gestão de empresas florestais, que supra a necessidade da coleta de dados em campo e permita aos usuários fazer uso destes dados para ter uma visão geral dos processos e tomar decisões.

Nesta primeira etapa foram desenvolvidos os módulos de Cadastro Florestal e Silvicultura, ficando os demais processos florestais como objetivos de trabalhos consequentes. Todo o sistema foi desenvolvido para a plataforma *web*, podendo ser acessado através de qualquer navegador (Google Chrome, Mozilla Firefox, Microsoft Edge, etc.), com exceção da parte de coleta de dados, para a qual será desenvolvida uma aplicação *mobile* compatível com os sistemas operacionais Android e iOS.

3.1 CADASTRO FLORESTAL

O módulo de Cadastro Florestal prove telas para a inserção de dados, onde o usuário pode cadastrar as informações importantes referentes à sua empresa que serão utilizadas pelo sistema, tais como as regiões dentre as quais as áreas da empresa estão divididas; as fazendas que a empresa possui; os talhões delimitados de cada fazenda; as atividades e operações que a empresa realiza; os insumos e os equipamentos que a empresa utiliza, bem como seus custos; o custo da mão de obra para cada atividade; e outras informações importantes para a gestão florestal.

Além destas, o módulo de Cadastro Florestal também armazena informações fundamentais para o planejamento da silvicultura e da colheita florestal, a nível de talhão, como área, espécie e material genético plantados, ano de plantio, ciclo, rotação, ano de intervenção e espaçamento. Estas informações devem ser preenchidas de forma manual pelos usuários, conforme as alterações forem ocorrendo, porém, vislumbra-se uma automação deste processo através das informações recebidas pelo aplicativo de coleta em trabalhos futuros.

3.2 SILVICULTURA

A silvicultura é o processo florestal que mais requer controle e planejamento, porém, por ter como alvo pequenas empresas e áreas florestais, o presente trabalho se limita a desenvolver somente as funcionalidades de cadastro e coleta de dados, por serem básicas para o controle e gestão. Em trabalhos futuros, com a inserção de novos módulos, o módulo de silvicultura será aprimorado.

O cadastro das informações pertinentes ao módulo de silvicultura pode ser realizado a partir do módulo de cadastro florestal e os apontamentos de atividades realizadas serão feitos no aplicativo de coleta. Na versão *web*, o módulo de silvicultura permite que o usuário consulte todos os apontamentos realizados, tendo acesso a todas as informações registradas e a informações extras, como as coordenadas de onde o apontamento foi realizado, com o local exibido em mapa, o horário real do apontamento, o nome do usuário que realizou o apontamento e informações sobre o dispositivo usado para realizar o apontamento.

3.3 APLICATIVO DE COLETA

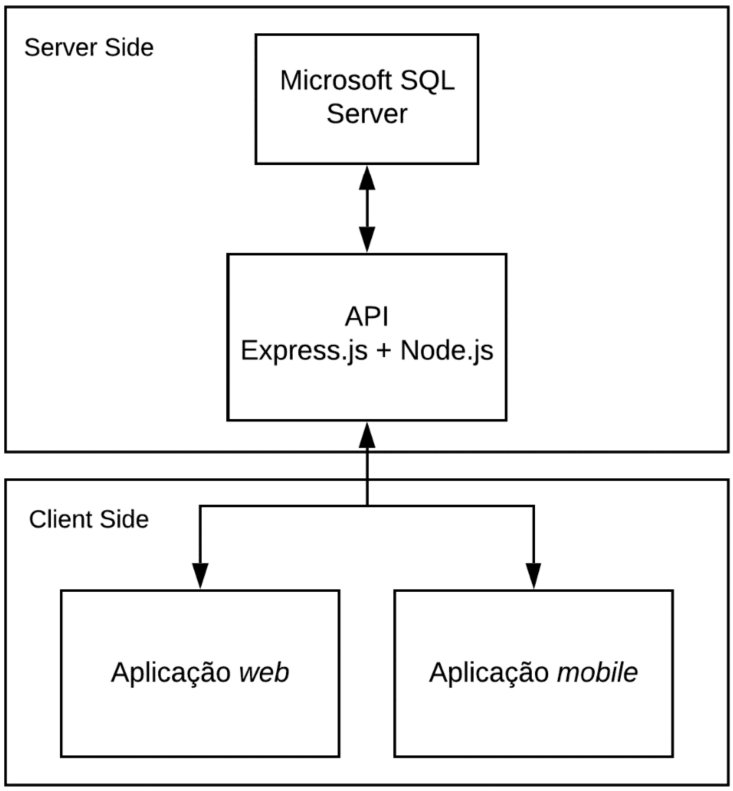
O aplicativo de coleta de dados faz uso das informações registradas no módulo de Cadastro Florestal para gerar formulários de campo. Nestes formulários é possível informar onde foi realizada a atividade, através da hierarquia região, fazenda e talhão, qual foi a atividade realizada, quantas pessoas participaram da atividade, quais insumos foram utilizados e suas quantidades, a área (espaço em hectares) na qual a atividade foi realizada, os horários de início e fim da execução da atividade e outras informações que poderão ser configuradas pelo usuário. Além destas informações, também é possível o registro de imagens e mensagens de áudio para auxiliar no esclarecimento de quaisquer eventualidades que venham a ocorrer no campo.

Considerando a realidade das áreas florestais no Brasil, onde em sua grande maioria o acesso à internet é indisponível ou limitado, o aplicativo será desenvolvido para funcionar *offline*. Desta forma, os usuários podem fazer o envio de formulários de campo mesmo sem estarem conectados. Além disso, o aplicativo deve ser de fácil uso, com botões e textos grandes e de fácil interpretação, para que possa ser utilizado por pessoas com pouca familiaridade a aplicações *mobile*.

3.4 ARQUITETURA

O sistema é dividido em duas partes: o lado-servidor (*server side* ou *backend*) e o lado-cliente (*client side* ou *frontend*) (FIGURA 1). No lado-servidor são realizados as consultas e os registros de dados no banco de dados Microsoft SQL Server. O lado-cliente, por sua vez, é o que está disponível aos usuários e é subdividido entre a aplicação *web* e a aplicação *mobile*. Ambas as aplicações do lado-cliente consomem a mesma API através de requisições.

FIGURA 1 – VISÃO GERAL DA ARQUITETURA DO SISTEMA.



Fonte: O autor.

3.4.1 Arquitetura do lado-servidor (*server side* ou *backend*)

O lado-servidor do sistema foi implementado em Node.js e exposto através de uma API criada com um servidor Express. A API acessa uma instância de banco de dados relacional do Microsoft SQL Server e expõe rotas HTTP para realizar operações CRUD (*Create, Read, Update and Delete*), isto é, criação, consulta, atualização e deleção de dados. Estas rotas seguem o estilo arquitetural REST (Transferência Representacional de Estado).

3.4.2 Arquitetura dos clientes (*client side* ou *frontend*)

A aplicação *web* foi desenvolvida através da biblioteca de JavaScript ReactJS e utiliza o Bootstrap como *framework* de CSS para prover o *layout* da aplicação. Os usuários podem acessar a aplicação *web* através de qualquer navegador de internet como, por exemplo, Google Chrome, Mozilla Firefox e Microsoft Edge.

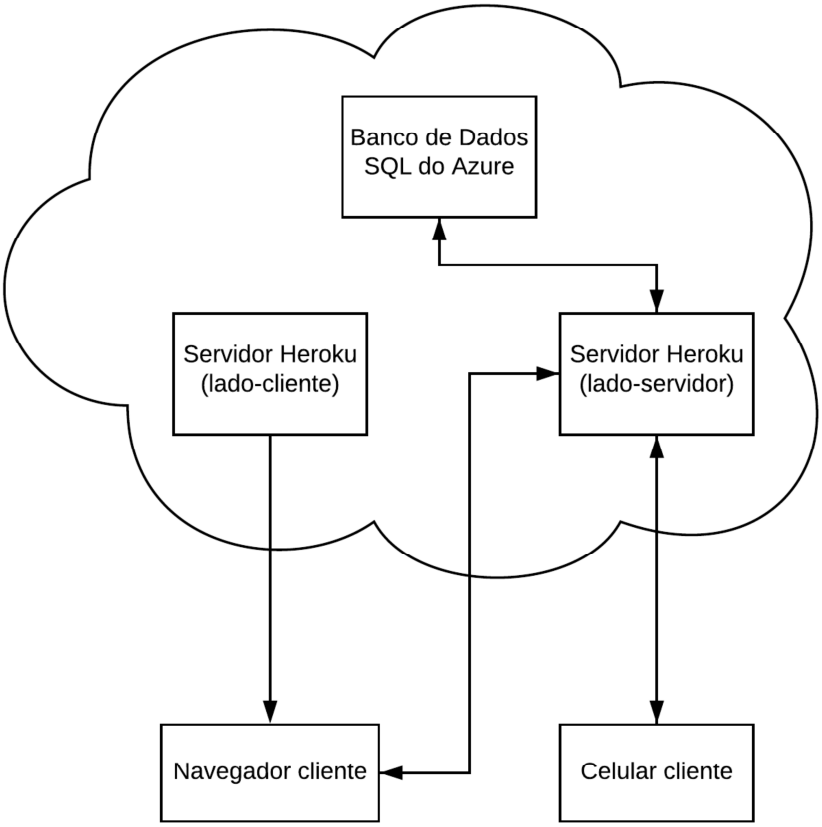
A aplicação *mobile* será desenvolvida com a *framework* React Native, que tem como base a mesma biblioteca JavaScript ReactJS, e estará disponível para os sistemas operacionais Android e iOS. Ambas as aplicações fazem requisições à API exposta pelo servidor para utilizar e manipular dados.

3.4.3 Arquitetura de implantação

O lado-servidor e o lado-cliente *web* do sistema foram publicados em duas máquinas virtuais Linux hospedadas na plataforma Heroku, no formato de plataforma como serviço (*Platform as a service*). Este formato elimina a necessidade de gerenciar, configurar e dar manutenção a uma máquina para rodar um servidor, transferindo todas estas responsabilidades para o prestador de serviço, neste caso, o Heroku. Os códigos de ambas as partes são mantidos em repositórios privados do GitHub e o processo de implantação do sistema (*deploy*) ocorre a partir do envio destes repositórios aos servidores do Heroku, que automatizam o processo de compilação e publicação.

O banco de dados do servidor também foi mantido no formato banco de dados como um serviço, sendo publicado através do serviço Banco de dados SQL do Azure, oferecido pela Microsoft Azure.

FIGURA 2 - ARQUITETURA DE IMPLANTAÇÃO DO SISTEMA.



O esquema da arquitetura de implantação está disposto na Figura 2.

Fonte: O autor.

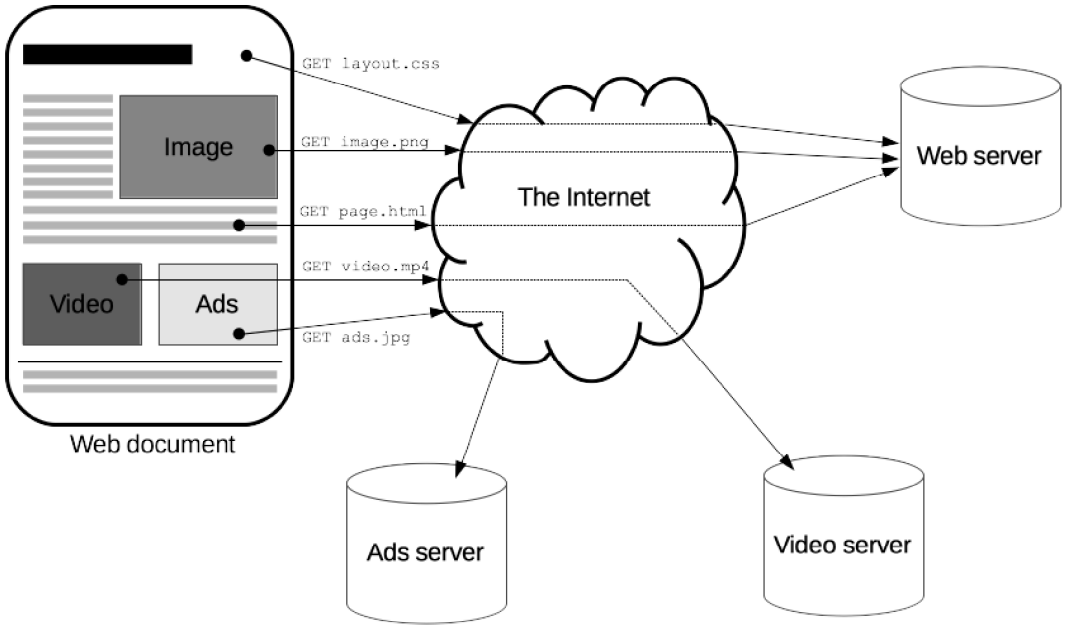
4 REVISÃO DA LITERATURA

4.1 PROTOCOLO HTTP

O *Hypertext Transfer Protocol* (HTTP), em português, Protocolo de Transferência de Hipertexto, é um protocolo de comunicação (na camada de aplicação segundo o Modelo OSI da ISO) utilizado para sistemas de informação de hipermídia, distribuídos e colaborativos (CARVALHO, 2014).

Conforme a documentação (Uma visão geral do HTTP - Mozilla Developer Network, 2019), HTTP é um protocolo (*protocol*) que permite a obtenção de recursos, tais como documentos HTML. É a base de qualquer troca de dados na Web e um protocolo cliente-servidor, o que significa que as requisições são iniciadas pelo destinatário, geralmente um navegador da Web. Um documento completo é reconstruído a partir dos diferentes sub documentos obtidos, como por exemplo texto, descrição do layout, imagens, vídeos, scripts e etc (FIGURA 3).

FIGURA 3 - REQUISIÇÃO DE DADOS ATRAVÉS DO PROTOCOLO HTTP.



Fonte: Disponível em <https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Overview>

Ainda conforme a documentação, clientes e servidores se comunicam trocando mensagens individuais (em oposição a um fluxo de dados). As mensagens enviadas pelo cliente, geralmente um navegador da Web, são chamadas de solicitações *(requests)*, ou também requisições, e as mensagens enviadas pelo servidor como resposta são chamadas de respostas *(responses)*.

Toda a comunicação entre as versões *web* e *mobile* e a API do sistema aqui apresentado se dá através do protocolo HTTP.

4.2 LINGUAGEM DE MARCAÇÃO DE HIPERTEXTO (HTML)

HTML é uma Linguagem de Marcação de Hipertexto, que é uma linguagem para produzir páginas web que serão interpretadas pelos navegadores. Os documentos HTML são documentos de textos simples, estáticos e que podem ser criados e editados em qualquer editor de texto comum (BATISTA, 2015).

No presente trabalho, a linguagem HTML foi utilizada de forma embarcada nos arquivos JavaScript, facilidade concedida pela biblioteca React.js, descrita a seguir.

4.3 FOLHA DE ESTILO EM CASCATA (CSS)

A linguagem HTML demarca os elementos de uma página *web*, porém é limitada quanto à determinação de sua aparência. Para controlar a aparência das páginas utiliza-se a linguagem CSS (*Cascading Style Sheets*) que descreve como seus elementos serão mostrados na tela.

4.4 LINGUAGEM JAVASCRIPT

A linguagem de programação principal adotada para o presente projeto é o JavaScript, usada para as três camadas do sistema: *web*, *mobile* e API.

JavaScript® (às vezes abreviado para JS) é uma linguagem leve, interpretada e baseada em objetos com funções de primeira classe, mais conhecida como a linguagem de script para páginas Web, mas usada também em vários outros ambientes sem browser, tais como node.js, Apache CouchDB e Adobe Acrobat. O

JavaScript é uma linguagem baseada em protótipos, multi-paradigma e dinâmica, suportando estilos de orientação a objetos, imperativos e declarativos (como por exemplo a programação funcional) (Mozilla Developer Network, 2019).

4.5 BIBLIOTECA REACTJS

ReactJS é uma biblioteca de JavaScript para construir interfaces de usuário, desenvolvida pelo Facebook e mantida como projeto *open source*. Hoje, ReactJS é considerada como a *framework* mais popular para desenvolvimento em JavaScript e vem sendo adotada e buscada por cada vez mais desenvolvedores e empresas.

4.5.1 Funcionalidades chave do ReactJS

- **JSX**

JSX é uma abreviação para JavaScript XML, uma extensão de sintaxe para a linguagem JavaScript que é semelhante ao XML ou HTML. Esta sintaxe permite que códigos HTML e JavaScript coexistam em um único bloco, facilitando a implementação visual e lógica de componentes, de forma dinâmica.

- **Componentes**

Aplicações desenvolvidas com ReactJS são construídas através da junção de múltiplos componentes, e cada componente possui sua própria lógica e controles. Estes componentes são reutilizáveis, tornando a aplicação altamente escalável e mais fácil de ser mantida.

- **Arquitetura flux**

Flux é um padrão de arquitetura de aplicação unidirecional que complementa a composição de componentes do ReactJS. Os benefícios de um fluxo de dados unidirecional garantem maior controle através da aplicação, tornando-a mais eficiente e flexível.

- **DOM virtual**

O Modelo de Objeto de Documento **(*DOM*)** é uma interface de programação para documentos HTML, XML e SVG. Ele fornece uma representação estruturada do documento como uma árvore. O DOM define métodos que permitem acesso à árvore, para que eles possam alterar a estrutura, estilo e conteúdo do documento. O DOM fornece uma representação do documento como um grupo estruturado de nós e objetos, possuindo várias propriedades e métodos. Os nós também podem ter manipuladores de eventos que lhe são inerentes, e uma vez que um evento é acionado, os manipuladores de eventos são executados. Essencialmente, ele conecta páginas web a scripts ou linguagens de programação. (Mozilla Developer Network, 2019).

Um DOM virtual é uma representação do objeto DOM original, atuando como uma ligação de dados unidirecional. Sempre que uma modificação ocorre na aplicação, toda a interface é renderizada novamente na representação virtual do DOM. Em seguida, ele busca por diferenças entre a representação antiga do DOM e o novo DOM virtual, permitindo que o DOM original atualize somente as coisas que sofreram alteração. Isso faz com que a aplicação seja mais rápida e não desperdice memória.

4.6 BOOTSTRAP

O Bootstrap é o mais popular *framework front-end* de código-fonte aberto para desenvolvimento de componentes de interface para aplicações *web* usando HTML, CSS e JavaScript. Ele fornece diversas classes de estilo pré-montadas que auxiliam no desenvolvimento do *layout* da aplicação, poupando o desenvolvedor de reescrevê-las.

Além disso, o Boostrap fornece um sistema de *grid* responsivo, que se adapta a todos os tipos de tela e uma grande biblioteca de componentes, como botões, tabelas, ícones, barras de navegação, cartões, dentre vários outros, prontos para serem utilizados.

4.7 MICROSOFT SQL SERVER

O Microsoft SQL Server é um sistema gerenciador de banco de dados relacional desenvolvido pela Microsoft. Sendo um banco de dados relacional, ele é usado para armazenar dados em uma estrutura definida de tabelas, colunas e linhas. É altamente escalável e largamente utilizado por empresas de todos os tamanhos, sendo adequado para armazenar dados sobre um único computador até sobre milhões de usuários.

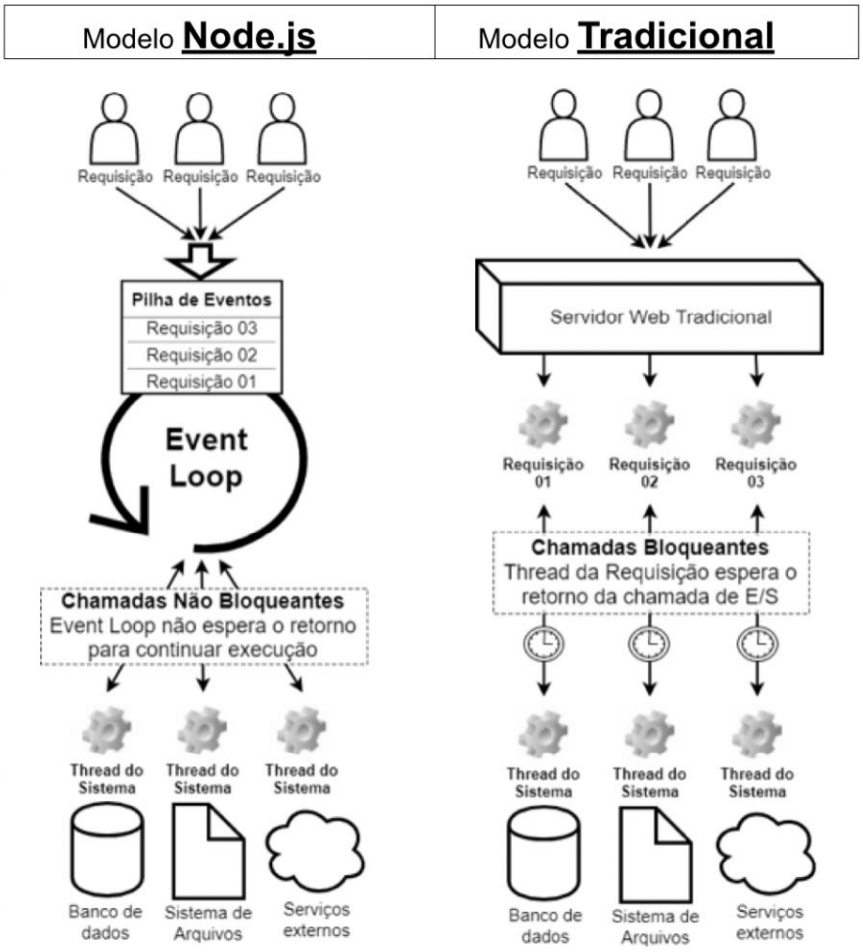
A linguagem usada pelo Microsoft SQL Server é o T-SQL, uma extensão da linguagem SQL, também desenvolvida pela Microsoft, que oferece a vantagem de programação procedural.

4.8 NODE.JS

Node.js é uma plataforma construída sobre o motor JavaScript do Google Chrome para facilmente construir aplicações de rede rápidas e escaláveis. Node.js usa um modelo de I/O direcionado a evento não bloqueante que o torna leve e eficiente, ideal para aplicações em tempo real com troca intensa de dados através de dispositivos distribuídos (NodeBR, 2016).

Comumente, as aplicações de rede criam um novo thread para cada conexão, que são bloqueados enquanto aguardam o resultado de uma requisição. Embora isto não seja um problema para aplicações pequenas e médias, pode gerar lentidão e travamentos em aplicações com grande número de conexões. O Node.js permite que muitos usuários possam enviar requisições ao servidor simultaneamente sem bloqueios, graças ao seu *Event Loop* (FIGURA 4).

FIGURA 4 - COMPARAÇÃO ENTRE UM SERVIDOR NODE.JS E UM SERVIDOR TRADICIONAL.



Fonte: Disponível em < https://www.opus-software.com.br/node-js/>

Além das vantagens de performance e escalabilidade citadas, o Node.js também possui uma vasta gama de bibliotecas pré-montadas que agilizam e ajudam a evitar falhas no desenvolvimento do servidor. O fato de ele ser escrito usando JavaScript também pode ser considerado uma vantagem, pois permite que tanto o servidor quanto as aplicações *front-end* sejam desenvolvidas usando a mesma linguagem de programação.

4.9 EXPRESS.JS

Express é um *framework* de desenvolvimento de aplicações web para o Node.js. Ele é responsável por gerenciar todas as requisições enviadas à API do sistema e definir as configurações comuns da aplicação web, como a porta a ser usada para conexão e a localização dos modelos que são usados para renderizar a resposta. Com ele, também é possível adicionar *middlewares* a qualquer ponto da requisição para interceptá-la e tratá-la.

4.10 HEROKU

Heroku é uma plataforma como serviço na nuvem que suporta diversas linguagens de programação. Ele permite que os desenvolvedores permaneçam focados unicamente no desenvolvimento, cuidando de toda a parte de infraestrutura. Ao alugar uma máquina virtual no Heroku, ela já vem com todas as configurações necessárias para publicar um servidor e um repositório git para o qual é possível enviar o código da aplicação. Sempre que uma atualização de código é enviada para o repositório, o código é automaticamente compilado e publicado.

Este serviço evita o trabalho de configuração, manutenção e gerenciamento de servidores para rodar uma aplicação e, por consequência, confere grande escalabilidade ao projeto, permitindo *upgrades* de máquina sem esforço.

4.11 GIT

Git é um sistema de controle de versões distribuídas para arquivos. Ele é amplamente utilizado por desenvolvedores para gerenciar o desenvolvimento de todo tipo de aplicação, pois permite o controle e histórico de alterações, versionamento de arquivos, dentre inúmeras outras facilidades e funcionalidades.

4.12 GITHUB

O GitHub é uma plataforma de hospedagem de códigos-fonte que utiliza o git para controle de versão. Ele permite a criação de repositórios git nos quais os usuários podem trabalhar em conjunto, mantendo todo o conteúdo sincronizado.

Seus repositórios podem ser abertos para o desenvolvimento de projetos de código aberto, ou privados, permitindo acesso somente a usuários autorizados.

4.13 MICROSOFT AZURE

O Microsoft Azure é uma plataforma destinada à execução de aplicativos e serviços na nuvem. Um de seus produtos é o Banco de Dados SQL, um banco de dados como serviço. Com este serviço, é possível hospedar um banco de dados relacional dimensionável e inteligente na nuvem, sem a necessidade de montar um servidor e toda a infraestrutura necessária para mantê-lo funcional e seguro.

4.14 REACT NATIVE

O React Native é um *framework* desenvolvido pelo Facebook que consiste em uma série de ferramentas que viabilizam a criação de aplicações móveis nativas para as plataformas iOS e Android, utilizando como base a biblioteca ReactJS, o que possibilita que todo o código seja escrito em JavaScript e tire proveito de todas as facilidades oferecidas pela biblioteca.

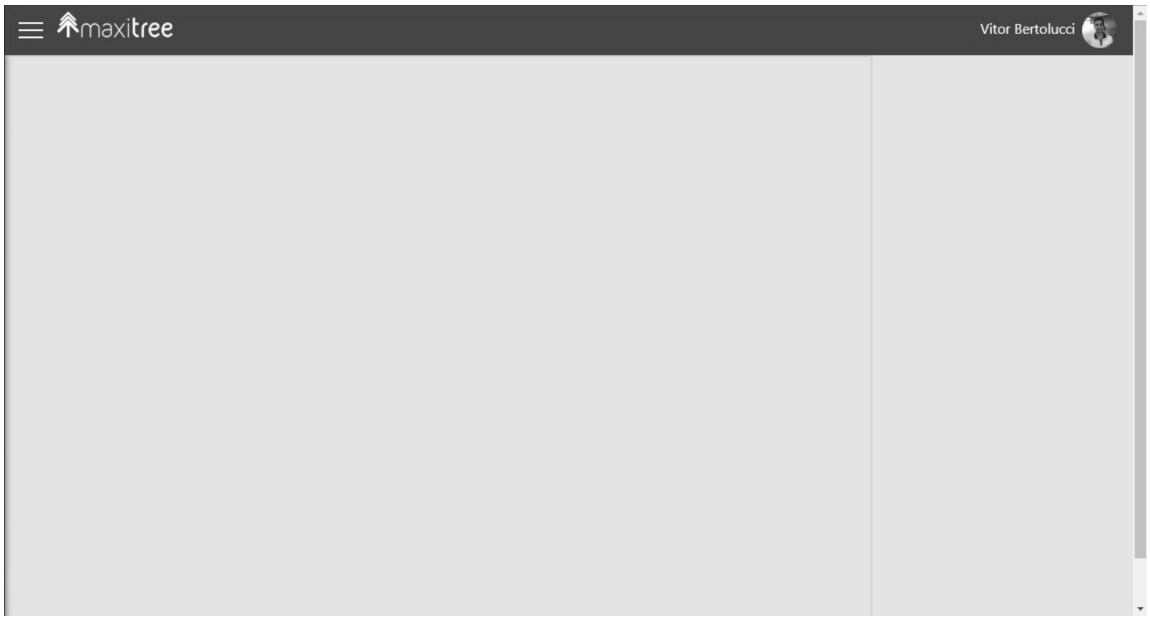
Diferente de outras alternativas para aplicações mobile que utilizam *webviews* para renderizar a aplicação, o React Native invoca as APIs de renderização nativas em ObjectiveC, para iOS, ou Java para Android. Portanto, a aplicação será renderizada utilizando componentes nativos da plataforma (EISENMAN, 2016 apud RAQUEL, 2017).

Além disto, o React Native traz uma importante vantagem ao desenvolvimento da aplicação que é a opção de recarregar (*reload*) o conteúdo do código sem a necessidade de recompilar todo o projeto, tornando os testes muito mais ágeis.

5 RESULTADOS E DISCUSSÃO

Após estudar e selecionar as tecnologias que seriam utilizadas no projeto, iniciou-se a fase de desenvolvimento. A primeira etapa desta fase foi desenvolver a base da interface para a versão web. Foram feitos diversos protótipos e, por fim, decidiu-se por utilizar um modelo de página com barra de navegação superior e um menu em *slider* lateral, que será responsável pela navegação entre os módulos do sistema (FIGURA 5).

FIGURA 5 - MODELO DAS PÁGINAS DA VERSÃO WEB DO SISTEMA.

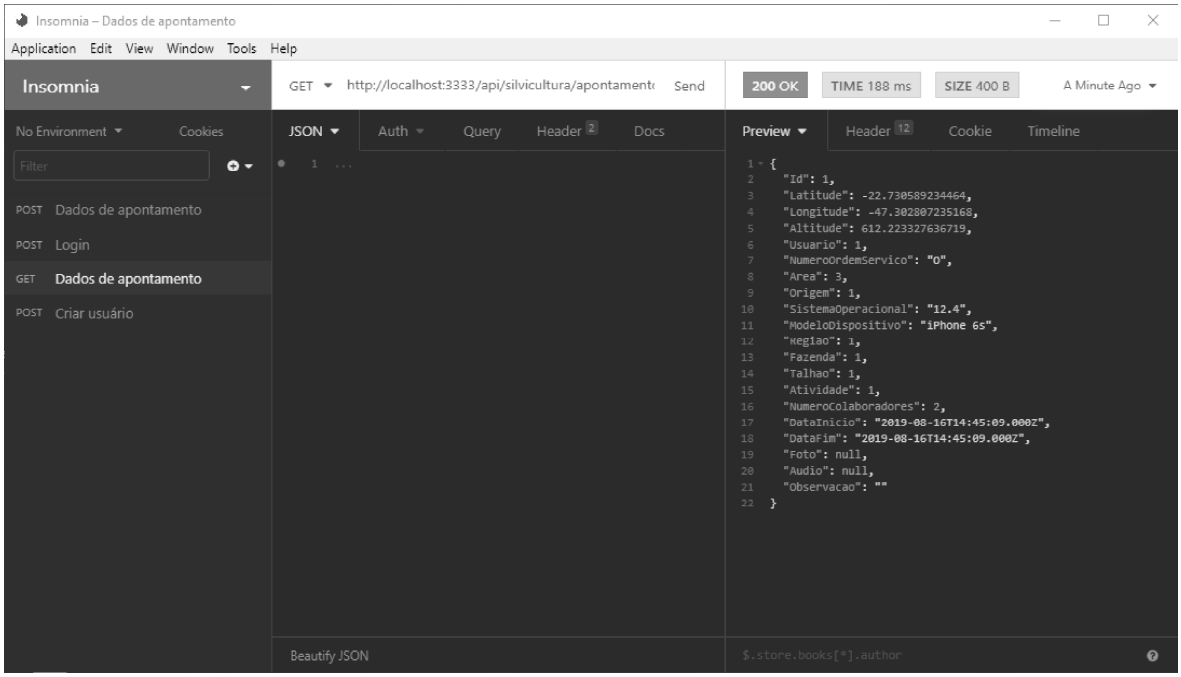


Fonte: o autor.

O menu em *slider* lateral construirá as árvores de navegação a partir dos módulos cadastrados em banco de dados e, portanto, se fez necessário iniciar paralelamente a etapa de modelagem do banco de dados. Esta etapa demandou muito estudo e reflexão, pois o modelo de dados definido regeria todo o sistema e, portanto, todos os componentes que seriam posteriormente desenvolvidos deveriam seguir este modelo.

Com a modelagem de dados parcial feita, deu-se início ao desenvolvimento da API Node.js, elaborando sua estrutura e algumas rotas iniciais de teste. Para testar as rotas foi utilizado um aplicativo chamado Insomnia, um cliente REST para desktop que permite configurar e executar uma coleção de requisições à API (FIGURA 6).

FIGURA 6 - TELA DO CLIENTE REST INSOMNIA.



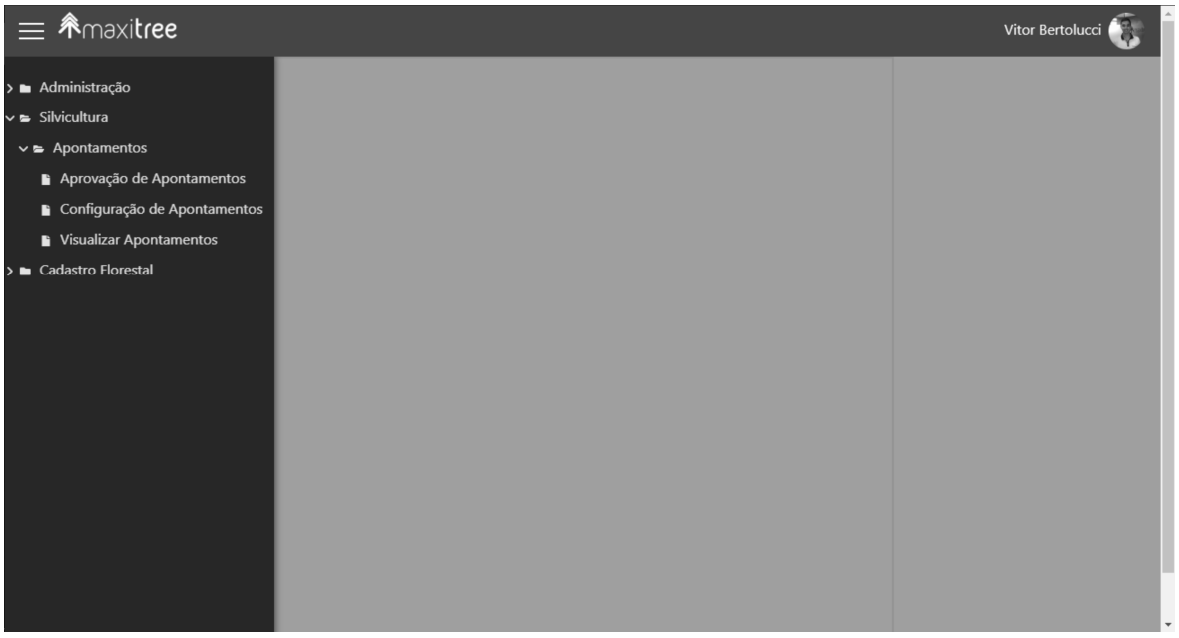
Fonte: o autor.

Através das requisições realizadas a partir do Insomnia, foi criado um usuário de administrador do sistema e as primeiras rotas implementadas e expostas foram as rotas de autenticação. Ao acessar a página do sistema, o usuário é levado a uma tela de login (FIGURA 7), sem o qual não é possível acessar nenhuma outra página. O login consiste em um endereço de e-mail e uma senha, que é armazenada com criptografia. Cada usuário possuirá um conjunto definido de permissões, que determinarão quais telas do sistema ele poderá acessar, e essas informações são obtidas através de requisições à uma outra rota exposta pela API. Em posse dessas informações, o menu em *slider* lateral pode finalmente construir as árvores de navegação (FIGURA 8).

FIGURA 7 - TELA DE LOGIN DO SISTEMA (VERSÃO WEB).

Fonte: o autor.

FIGURA 8 - MENU EM SLIDER LATERAL COM ÁRVORES DE NAVEGAÇÃO CONSTRUÍDAS.



Fonte: o autor.

Como um dos objetivos do sistema é a coleta de dados em campo e estes são importantes, iniciou-se o desenvolvimento do aplicativo *mobile*, ficando assim o projeto com 4 partes em desenvolvimento simultaneamente: versão *web*, banco de dados, API e versão *mobile*. A primeira tela desenvolvida para o aplicativo foi, também, a tela de login (FIGURA 9), que usa a mesma rota de autenticação usada pela versão *web*. Respeitando a necessidade de o aplicativo ser acessível a usuários pouco familiarizados com tecnologia, ele foi desenvolvido para ser o mais simples possível, com poucas opções de clique e botões grandes.

FIGURA 9 - TELA DE LOGIN DO SISTEMA (VERSÃO MOBILE).

Fonte: o autor.

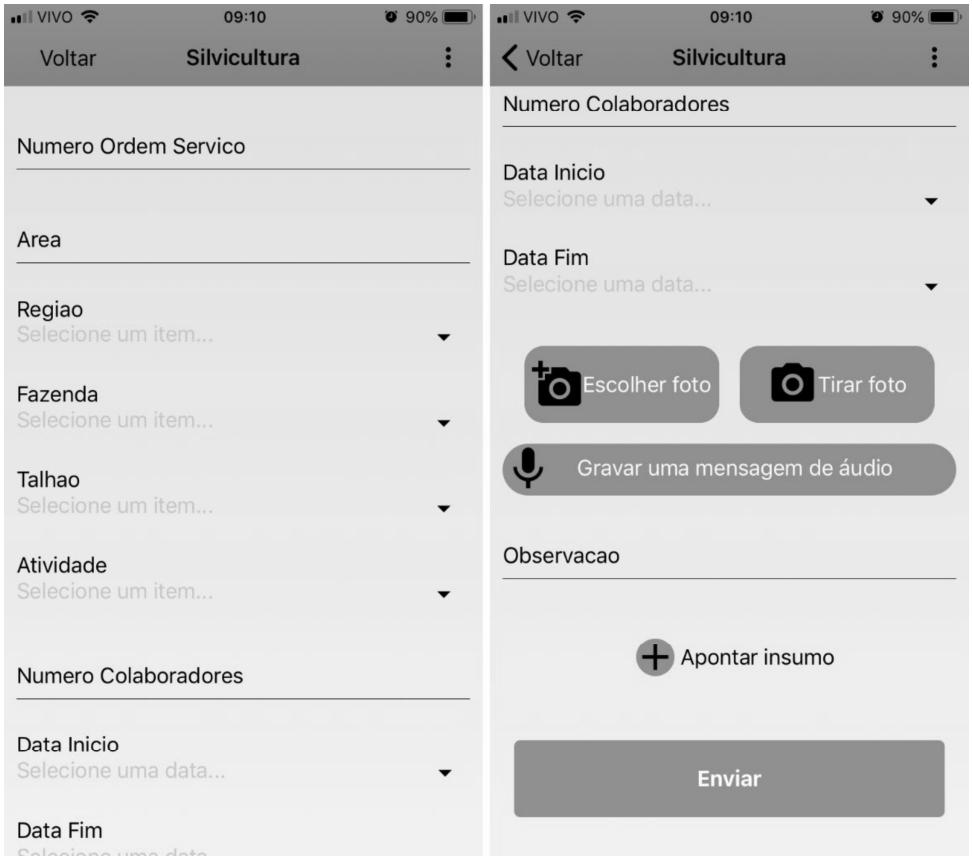
Após a autenticação, o usuário é encaminhado para a tela inicial do aplicativo, que exibe os módulos aos quais ele tem acesso (FIGURA 10). Cada módulo do sistema representará um formulário de campo no aplicativo. O primeiro

formulário a ser desenvolvido foi o de silvicultura (FIGURA 11) e, durante seu desenvolvimento, foram levantadas importantes alterações ao modelo de dados necessárias para seu funcionamento. Os campos escolhidos para compor este formulário foram: número da ordem de serviço, área em ha na qual a atividade foi realizada, região, fazenda e talhão onde a atividade foi realizada, atividade realizada, operação realizada, número de colaboradores que atuaram na atividade, horários de início e término da atividade, insumos utilizados e suas quantidades (em forma de sub formulário), fotografia da atividade, mensagem de áudio e observações. Alguns destes campos possuem relações entre si e todos possuem relações com outras tabelas de dados do sistema, que constituirão informações importantes para a gestão. Para os primeiros testes, foram registrados dados para todos estes campos diretamente no banco de dados.

FIGURA 10 - TELA INICIAL DO APLICATIVO.

Fonte: o autor.

FIGURA 11 - FORMULÁRIO DE SILVICULTURA NO APLICATIVO.

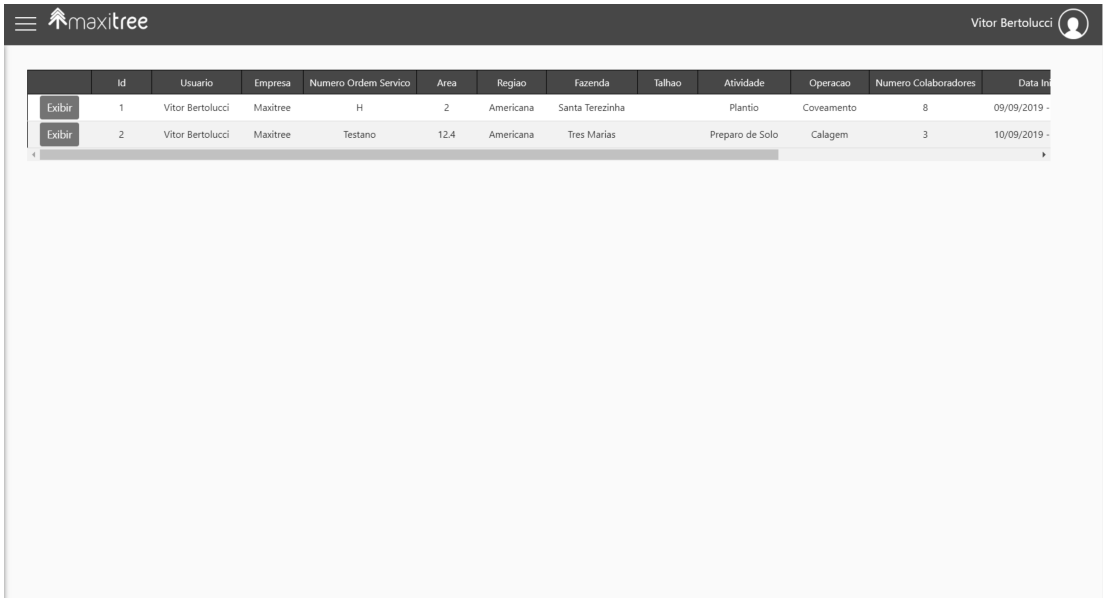


Fonte: o autor.

O envio do formulário foi desenvolvido para funcionar com ou sem acesso à internet, permitindo que o aplicativo se adeque ao uso no campo. Desta forma, se o usuário realizar o apontamento estando sem conexão, os dados apontados ficarão armazenados no dispositivo e serão enviados quando a conexão for estabelecida. Seguindo este mesmo princípio, a tela de login também armazenará as informações do usuário no dispositivo, permitindo que ele faça login sem estar conectado.

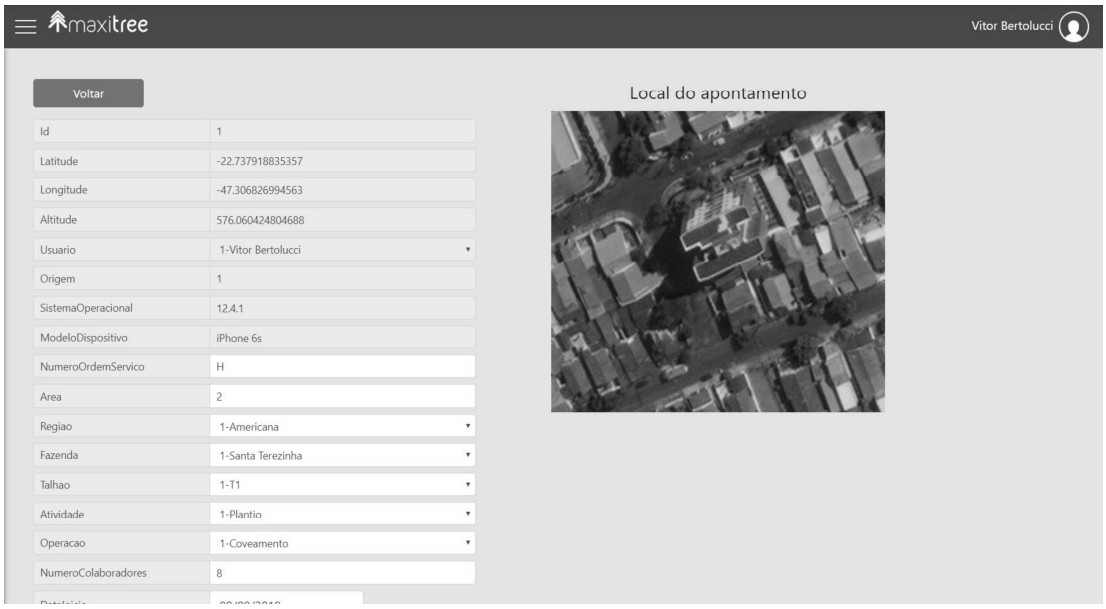
Quando o envio de um formulário é concluído, seus dados ficarão disponíveis na tela de aprovação de apontamentos da versão *web* (FIGURA 12), onde poderão ser conferidos, sofrer algumas correções e serem aprovados ou reprovados. Se um apontamento é aprovado, seus dados serão então registrados no sistema, contabilizando os custos e consumos que ele acarretou. Esta etapa de aprovação é importante para garantir que informações incorretas não sejam contabilizadas e decidiu-se por mantê-la na versão *web* para permitir que os trabalhadores em campo consigam realizar seus apontamentos sem interferências.

FIGURA 12 - TELA DE APROVAÇÃO DE APONTAMENTOS



Fonte: o autor

FIGURA 13 - TELA DE DETALHAMENTO DE APONTAMENTOS

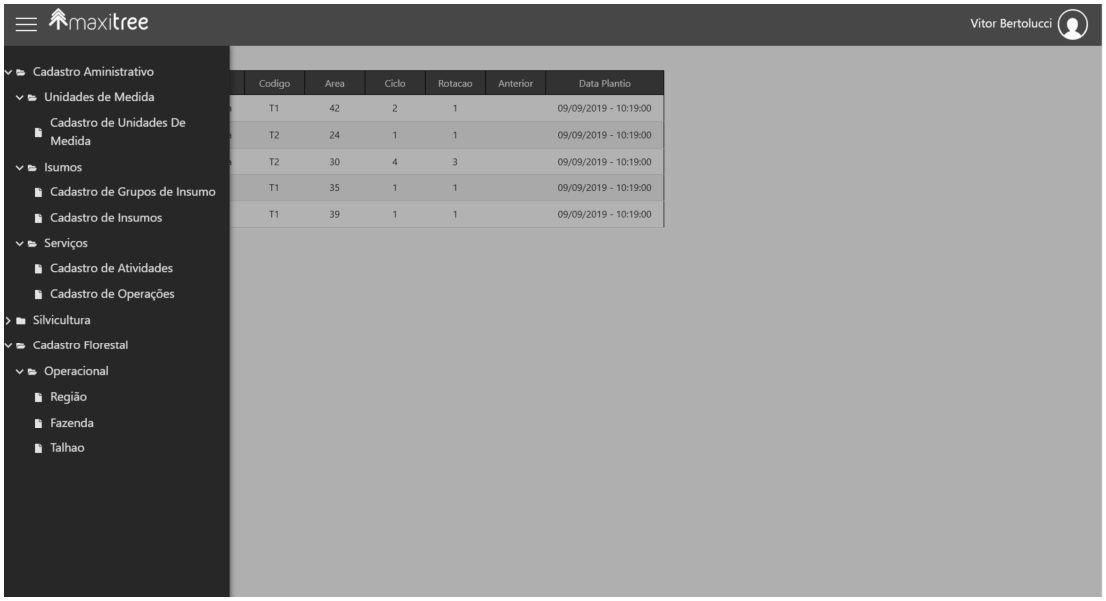


Fonte: o autor

Tendo finalizado o funcionamento dos apontamentos, seguiu-se para as telas de cadastro de informações, que serão responsáveis por exibir os dados cadastrados em forma de tabela, permitir a edição destes dados e por permitir o

registro de novos dados. Com base na modelagem de dados realizada, as telas necessárias serão: cadastro de unidades de medida, cadastro de grupos de insumo, cadastro de insumos, cadastro de atividades, cadastro de operações, região, fazenda e talhão (FIGURA 13).

FIGURA 14 - LISTA DE TELAS NO MENU LATERAL



Fonte: o autor

6 CONCLUSÃO E TRABALHOS FUTUROS

Neste trabalho foram apresentados a arquitetura e o desenvolvimento de um sistema de gestão florestal simples, composto por um servidor *back end* implementado em Node.js, uma versão *web* implementada com ReactJS e uma versão *mobile* implementada com React Native.

O sistema está em pleno funcionamento e atende a necessidade de pequenas empresas florestais, garantindo-as assertividade e agilidade nos apontamentos realizados em campo e as auxiliando no controle e gestão das atividades através de uma interface padronizada para exibição e registro de dados.

Tendo concluído esta primeira versão do sistema, abre-se um horizonte de possibilidades para aprimoramento e adição de novas funcionalidades e módulos. Nos próximos trabalhos serão desenvolvidos módulos para os processos de colheita e inventário florestal, que trarão as funcionalidades de apontamento, registro e visualização de dados no mesmo formato dos módulos atuais. Além destes, pretende-se desenvolver um módulo de controle de estoque para gerenciar os estoques de insumos.

Ainda, em trabalhos futuros, após o desenvolvimento de todos os módulos, pretende-se implementar integrações entre eles, de forma que os dados do cadastro florestal sejam e mantidos atualizados automaticamente a partir dos apontamentos de silvicultura e colheita, e possibilitando usar o sistema para gerar o planejamento anual da silvicultura.

7 REFERÊNCIAS

BATISTA, R. M. **Proposta de desenvolvimento de software para gestão de viagens da DEAD/UFVJM - SIGEV.** 2015. 71p. Dissertação (Bacharel em Sistemas de Informação) – Universidade Federal dos Vales do Jequitinhonha e Mucuri, Diamantina, 2015.

CALLADO, A. A. C. **Custos: um desafio para a gestão no agronegócio.** In: VI Congresso Brasileiro de Custos, 6, 1999, São Paulo - SP. CBC, 1999. P. 3.

CARVALHO, A. A. P. **Arquitetura e desenvolvimento de uma aplicação web distribuída baseada em JavaScript.** 2014. 38p. Dissertação (Bacharel em Sistemas de Informação) – Universidade Federal do Estado do Rio de Janeiro, Rio de Janeiro, 2014.

JavaScript. Mozilla Developer Network. Disponível em <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript>**.** Acesso em 19 ago. 2019.

Modelo de Objeto de Documento (DOM). Mozilla Developer Network. Disponível em <https://developer.mozilla.org/pt-BR/docs/DOM/Referencia_do_DOM>**.** Acesso em 20 ago. 2019.

O que é Node.js. NodeBR. Disponível em <http://nodebr.com/o-que-e-node-js/>**.** Acesso em 23 ago. 2019.

RAQUEL, L. G. **Dita ofertas: uma aplicação para reproduzir ofertas de produtos no rádio do carro.** 2017. 60p. Dissertação (Graduação em Ciência da Computação) – Universidade Regional de Blumenau, Blumenau, 2017.

Uma visão geral do HTTP. Mozilla Developer Network. Disponível em <https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Overview>**.** Acesso em 16 ago. 2019.