SITE DE COMUNICAÇÃO PARA CLÍNICA VETERINÁRIA

WEBSITE FOR A VETERINARY CLINIC

Kleber José da Silva[1], Vivian Toledo Santos Gambarato[2]

[1]Aluno do Curso de Tecnologia em Análise e Desenvolvimento de Sistemas, Faculdade de Tecnologia de Botucatu, Av. José Ítalo Bacchi, sn, Botucatu, SP, kb-max@hotmail.com
[2] Profª Ma da Faculdade de Tecnologia de Botucatu. Av. José Ítalo Bacchi, sn, Botucatu, SP, vivian.gambarato@fatec.sp.gov.br

RESUMO

Atualmente, as tecnologias relacionadas à difusão de informação e conteúdo se espalham de maneira a cobrir o maior número de campos e aumentar o conhecimento de diversas maneiras. Todos se beneficiam da velocidade e da alta disponibilidade com que grandes quantidades de informação são distribuídas. Essa mesma grande quantidade de informações pode gerar dificuldades, tanto no consumo, bem como na interpretação das mesmas. Como no exemplo de uma clínica veterinária, os clientes podem ter interpretações diferentes de um especialista, em relação ao mesmo assunto. Assim, se faz necessária a criação de ferramentas que auxiliem na fácil assimilação e propagação de tais informações, usando um sistema que utilize as tecnologias mais modernas e que não criem barreiras na sua utilização. Para facilitar esse acesso, foi escolhida a criação de um sistema que utiliza a arquitetura de uma *API REST,* que possibilita sua integração com diversas linguagens e dispositivos. Esse sistema disponibiliza ferramentas para a criação de um canal de comunicação entre clientes e especialistas de uma clínica veterinária, atendendo de maneira satisfatória e ajudando a quebrar barreiras na comunicação entre ambos.

Palavras-chave: API. Arquitetura. Disponibilidade. *REST*. Sistema. *Web*.

ABSTRACT

Today, technologies related to the diffusion of information and content spread to cover as many fields as possible and increase knowledge in a variety of ways. Everyone benefits from the speed and high availability with which large amounts of information are distributed. This same great amount of information can generate difficulties, both in the consumption, as well as in the interpretation of the same ones. As in the example of a veterinary clinic, where clients may have different interpretations of a specialist, in relation to the same subject. Thus, it is necessary to create tools that help in the easy assimilation and propagation of such information, using a system that uses the most modern technologies and that do not create barriers in its use. To facilitate this access, the creation of a system that uses the architecture of a REST API was chosen, which allows its integration with several languages and devices. This system provide tools for creating a channel of communication between clients and specialists of a veterinary clinic, attending in a satisfactory manner and helping to break down barriers in communication between both.

Key words: API. Architeture. Availabillity. REST. System. Web.

1. INTRODUÇÃO

Com os desafios do mundo moderno, o uso das tecnologias em diversas áreas tem facilitado a compreensão e ajudado a resolver problemas que antes eram muito difíceis de serem solucionados. Na área da saúde, a tecnologia tem expandido fronteiras e traz consigo muitas inovações, coisas que antes eram impossíveis, hoje são possíveis. É cada vez mais comum a utilização de serviços e aplicações que utilizam a Internet para diversos fins, como a comunicação, o entretenimento e a pesquisa.

Com o uso de sistemas, pode-se ganhar agilidade e eficiência nos pré atendimentos e dinamizar consultas a banco de dados, além de facilitar o acesso a informação, que é muito importante nos dias atuais. Porém, com tamanho volume de informação não é muito fácil ter uma interpretação razoável de uma informação e aproveitar ao máximo seu conteúdo. Usando como exemplo os clientes de uma clínica veterinária, muitos não conseguem obter informações a respeito das melhores práticas em relação ao cuidado com o seu animal de estimação. Concentrar e disponibilizar tais informações em um canal de fácil acesso, possibilita sua difusão de maneira mais eficiente e evita a tomada de decisões de maneira equivocada.

Utilizando linguagens de programação, como o *Java* e *Javascript*, pode-se construir uma aplicação que visa simplificar a comunicação entre clientes e médicos veterinários, com informação clara e precisa, com alta disponibilidade e fácil acesso com diversos dispositivos, diminuindo a barreira da comunicação entre ambos.

Com a necessidade de aumentar o desempenho de aplicações que processam dados via *web* de maneira remota, foi utilizada a arquitetura *REST* (*Representacional State Transfer*), que descreve o comportamento para aplicativos de serviços em rede. O padrão *REST* é muito utilizado na integração entre serviço *web*, onde a troca de dados deve ser ágil e de fácil acesso (FIELDING, 2000). A arquitetura *REST* não deve ser considerada um modelo a ser seguido, mas um conjunto de práticas com o objetivo de beneficiar o serviço, tornar mais eficiente, coeso e integrado com as próprias tecnologias da *web*.

São usados métodos *HTTP* (*HyperText Markup Language*) para a manipulação dos recursos, através da *URL*, que identifica o método a ser utilizado, cada qual com sua finalidade. Atualmente a representação mais comum dos dados é o formato *JSON* (*Javascript Object Notation*), sendo que o *XML* (*Extensible Markup Language*) e o *CSV* ainda são muito utilizados (DEWAILLY, 2015).

Os recursos são um conceito muito importante, sem que qualquer informação, dados que tenham a possibilidade de ser armazenados podem ser considerados um recurso. Todas as

requisições feitas à um serviço *REST* devem conter as informações necessárias para que o mesmo possa receber, processar e devolver tais requisições de maneira adequada. Todas as informações sobre o estado do usuário devem ser mantidas no lado do cliente, com a finalidade de reduzir grandes quantidades de recursos, como processamento e armazenamento. O cliente pode enviar mensagens, que podem ser visualizadas por um especialista usuário da aplicação, para que possam ser sanadas as dúvidas e enviar uma resposta adequada.

O principal objetivo foi desenvolver um site com o objetivo de facilitar a interação entre clientes e especialistas de uma clínica veterinária, utilizando padrões e linguagens que estão entre as mais usadas do mercado para fornecer uma experiência de interação de maneira satisfatória.

2. MATERIAL E MÉTODOS

Para o desenvolvimento do site, algumas tecnologias foram utilizadas e serão descritas a seguir.

2.1 *API*

Uma *API* (*Application Programming Interface* ou Interface de programação de aplicativos) é uma interface que permite a integração com outros sistemas, as vezes de outras empresas. Assim, uma aplicação de uma determinada empresa pode ser utilizada para complementar o serviço de outra aplicação. Possui um conjunto de rotinas e padrões que são utilizados para diversos fins, como criação de interfaces gráficas, cálculos, renderização de objetos 3d, comunicação e padronização entre sistemas (DEWAILLY, 2015).

Normalmente uma *API* é utilizada quando há um objetivo que outros desenvolvedores e sistemas utilizam sua aplicação, sendo que há várias *APIs* que disponibilizam uma extensa documentação e recursos que facilitam sua comunicação. Existem diversos tipos de *APIs*, com se pode citar o Google, que possui vários serviços, como o *Google Maps, Youtube API, AdSense, Google+,* dentre outros. O próprio *Google* fornece material para suas *APIs* possam ser utilizadas e até integradas a outros projetos (GOOGLE, 2018).

2.2 Arquitetura *REST*

A arquitetura *REST*, citada anteriormente nesse artigo, refere-se a uma sigla que significa *Representacional State Transfer* (Transferência de Estado Representacional), onde cada requisição *web* representa um estado, que pode ser ou não alterado. Não é um padrão ou

uma tecnologia propriamente dita, mas um modelo que pode ser utilizado para planejar e construir sistemas cujo objetivo seja a comunicação em rede (FIELDING, 2000).

Desenvolvido por Roy Thomas Fielding em sua tese de mestrado, que também é um dos criadores do protocolo HTTP, foi amplamente utilizado. O próprio protocolo HTTP utiliza os princípios da arquitetura *REST* como modelo, sendo que sua idealização de um modelo de objeto HTTP foi usada nos padrões *HTTP* 1.1, que visa a construção de uma aplicação *web* bem definida e eficiente, onde o usuário avança entre os recursos por meio de ligações que usam as principais operações (*GET*, *DELETE*, *POST*, *PUT*) que o levam para o próximo recurso, que será transferido para a utilização do usuário. A própria arquitetura ignora detalhes de implementação e especificações do protocolo, focando em oferecer restrições e conjuntos de práticas que visam estabelecer o melhor aproveitamento dos recursos durante a comunicação (FIELDING, 2000).

O padrão *REST* conta com princípios que regem sua filosofia, sendo que os principais são (DEWAILLY, 2015):

- *Client-server* (cliente-servidor): os clientes e servidores são separados e não há implicação em como as informações são usadas e armazenadas. O servidor fica com a missão de processar as requisições e o cliente se responsabiliza em fornecer uma interface de acesso ao usuário;
- *Stateless*: as requisições *HTTP* devem carregar todas as informações necessárias, fazendo com que nem o servidor e nem o cliente precisem registrar o estado das comunicações, utilizando de recursos para manter a sessão entre ambos;
- Deve possuir uma série de ações que se aplicam a todos os recursos da informação;
- O uso de formatos de representação padrão;
- Recursos: são os elementos das informações, essenciais para a manipulação de requisições;
- *Cache*: usado para melhorar o desempenho da aplicação, onde parte das requisições são armazenadas no cliente, diminuindo o número de acessos e aumentando a eficiência e estabilidade da aplicação;
- *Layers* (camadas): permite o particionamento da aplicação, encapsular serviços, impedindo que se acesse camadas restritas;

- Representação dos recursos: toda requisição *HTTP* para um serviço *REST* deve enviar uma resposta em um formato padronizado. Os mais comuns são o *JSON*, o *XML*, o *HTML* e o *YAML* (*YAML Ain't Markup Language*).

2.3 *Java*

A linguagem de programação *Java*, criada por James Gosling e sua equipe na *Sun Microsystens* em 1995. Possui como característica o fato de seu código não ser compilado diretamente para a linguagem de máquina nativa de um sistema operacional, mas sim interpretada por uma máquina virtual, que a codifica de acordo com cada sistema. É uma linguagem que possui características de orientação a objetos e integra funções como alta portabilidade (capacidade de ser executada nos mais diversos dispositivos e sistemas), recursos de rede e segurança, além de alta estabilidade e garantir compatibilidade com versões anteriores (DEITEL; DEITEL, 2017).

Hoje, o *Java* é uma plataforma de desenvolvimento robusta, sendo que a linguagem de programação padrão é a que carrega seu nome. Dispõe de diversas tecnologias e padrões, como o *Java SE* e o *Java EE*.

Atualmente, o *Java* passou por mudanças sendo agora atualizado e a cada ciclo de seis meses. E cada três anos é lançada uma versão LTS (*Long- Term Support*), que traz melhorias e estabilidade a plataforma. Aversão atual é a 11 LTS (DIAS, 2018).

2.4 *Spring Framework*

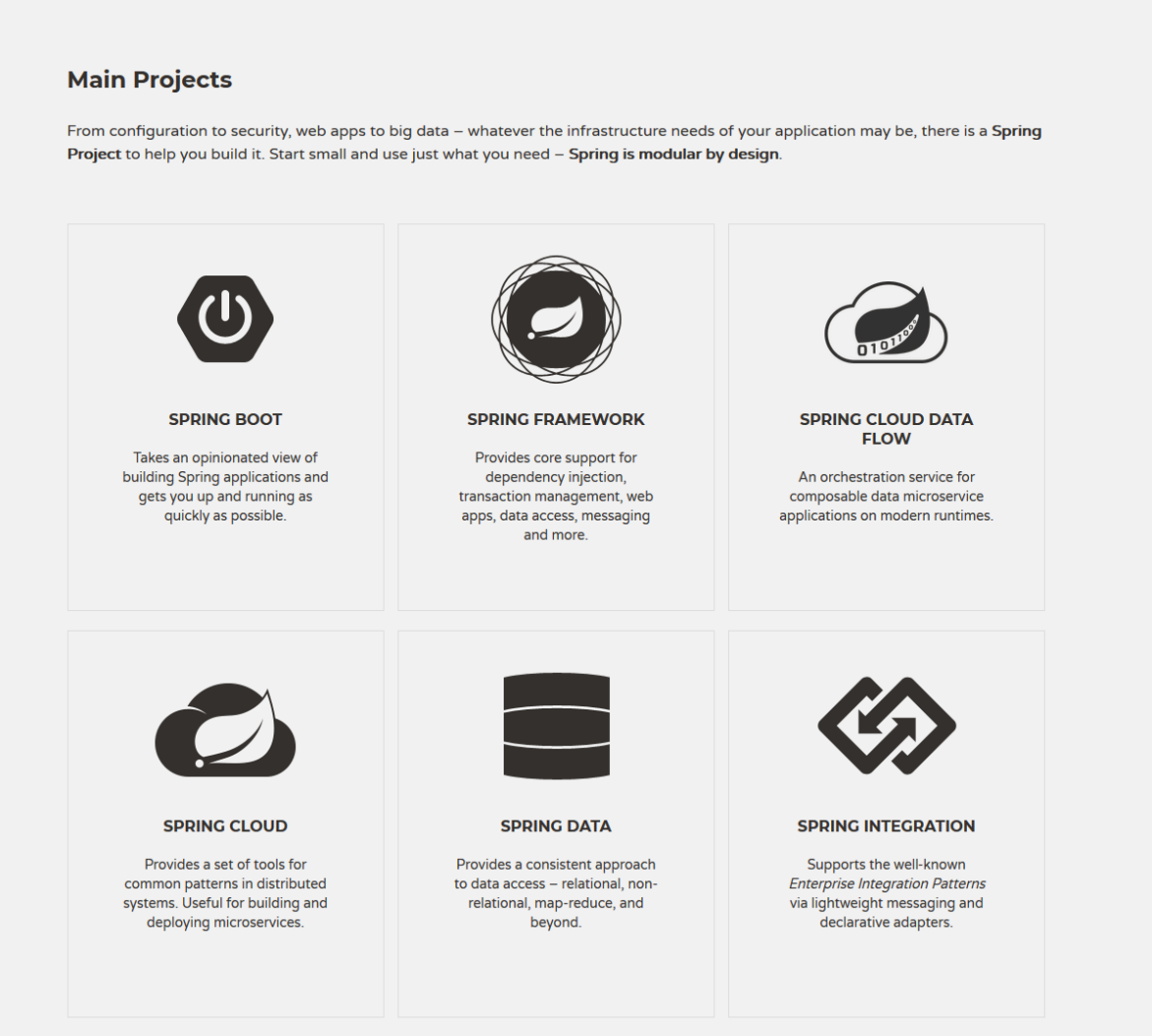
O *Spring* é um dos *frameworks open source Java* mais utilizados. Criado por Rod Johnson em 2002, foi desenvolvido utilizando ideias e conceitos dos padrões de projeto IoC (Inversão de controle) e injeção de dependência. Foi desenvolvido para facilitar a programação em Java, permitindo a criação de sistemas com recursos e de maneira rápida. Hoje é mantido pela *Pivotal Software Inc*, com sede em Palo Alto e São Francisco, Califórnia (Spring Framework Overview, 2018).

O *framework Spring* dispõe de diversas ferramentas e tecnologias que simplificam a implementação e desenvolvimento de código de infraestrutura e sistemas, reduz o acoplamento dentro do sistema e modulariza seus componentes. Além de estar em constante evolução, sempre adicionando novas funcionalidades de maneira rápida. Ao contrário do *Java* EE, onde há uma demora na implementação de novidades, já que existe uma necessidade maior de tornar a nova funcionalidade compatível com versões anteriores e evitar que aplicações já em funcionamento “quebrem”. O *Spring* busca novidades que visam tornar o desenvolvimento

mais eficiente, trazendo novos recursos de acordo com as novidades do mercado e as tendências apresentadas em congressos e pesquisas (BOAGLIO, 2017).

O *Spring* possui diversos componentes que visam modularizar e tornar necessário apenas o que o projeto precisar durante o desenvolvimento e seu futuro *deploy*. Como exemplo, existem *o Spring Data, Spring Secutity, Spring MVC, Spring Cloud*, o que facilita muito o desenvolvimento de aplicações, já que possuem uma forte integração entre si, bastando adiciona-las no projeto e com o mínimo ou as vezes sem necessidade de configurações extras já está pronto para colocar em funcionamento (BOAGLIO, 2017). A Figura 1 mostra alguns módulos do *Spring Framework*

Figura 1 – Exemplo de projetos modulares do *Spring Framework*



Fonte: (Spring Framework Overview, 2018)

2.5 *Maven*

É uma ferramenta responsável pelo gerenciamento de dependências e automação de compilação, sendo mantido pela *Apache Foundation*, é baseado no modelo objeto, que utiliza um arquivo (*pom.xml*) para fazer o *download* dos pacotes e bibliotecas que o projeto necessita, além de contar com diversos modelos semi-prontos, que trazem um conjunto de especificações para agilizar o início de um projeto (Welcome to Apache Maven, 2018)

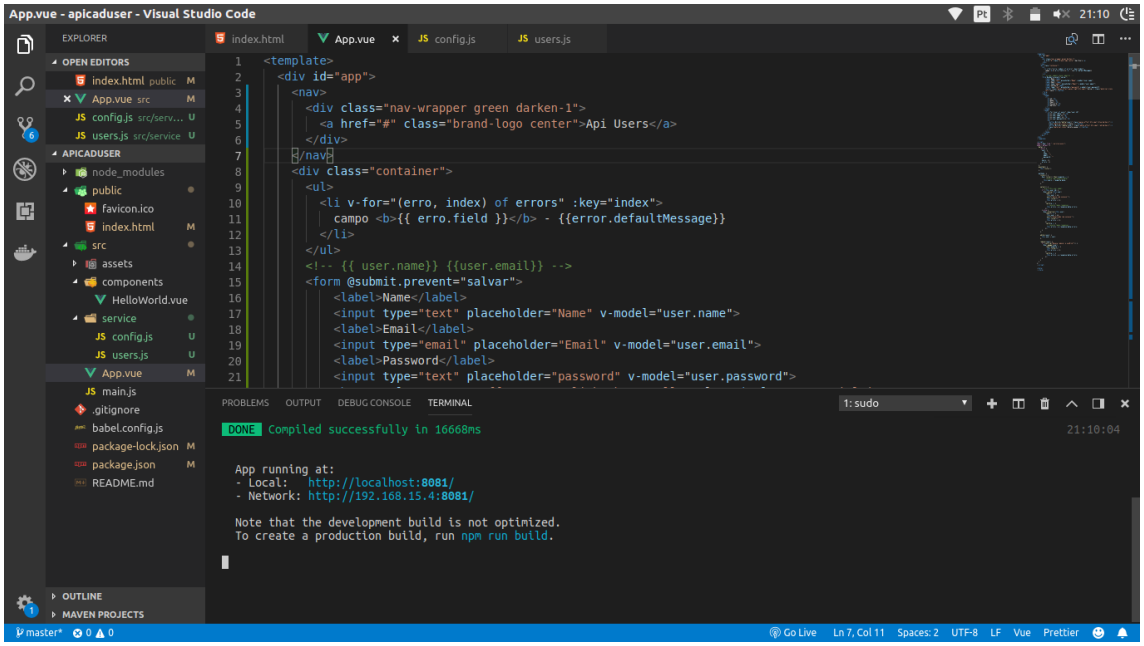
Ainda segundo o autor, permite também que o desenvolvedor ou a equipe foque no modelo de negócios, deixando o *Maven* encarregado de decidir quais bibliotecas são necessárias para compilar e realizar a execução do código. Usando um comando é possível validar, gerar código extra necessário, além de realizar os testes e finalizar o projeto, já com o pacote com o código finalizado. Uma das vantagens de sua utilização é fazer com que o projeto siga uma padronização, como exemplo a estrutura de pastas, que são definidas pela ferramenta. Isso torna todo o código mais organizado e focado nos objetivos necessários para a construção da aplicação (OTTERO, 2012).

2.6 *Vue.JS*

O *Vue.Js* é um dos muitos *frameworks open source Javascript* existentes no mercado. Criado por Evan You, em um dos seus projetos no *Google Creative Labs*, para atender suas necessidades na criação da interface do protótipo de um projeto seu, com finalidade de reduzir a escrita de código *HTML* e tempo. O *Vue.Js* permite que se vincule os modelos de dados à camada de apresentação de maneira rápida e elegante, permitindo a reutilização de componentes por toda a aplicação, além de permitir a criação de um projeto simples e como é um *framework* progressivo, a aplicação pode escalar até se tornar uma aplicação de uso em larga escala (YE, 2018).

Possui uma usabilidade e curva de aprendizado simples, bastando adicionar o arquivo vue.js no projeto. A Figura 2 mostra o código *Javascript* do *Vue.JS* usado para criar a página.

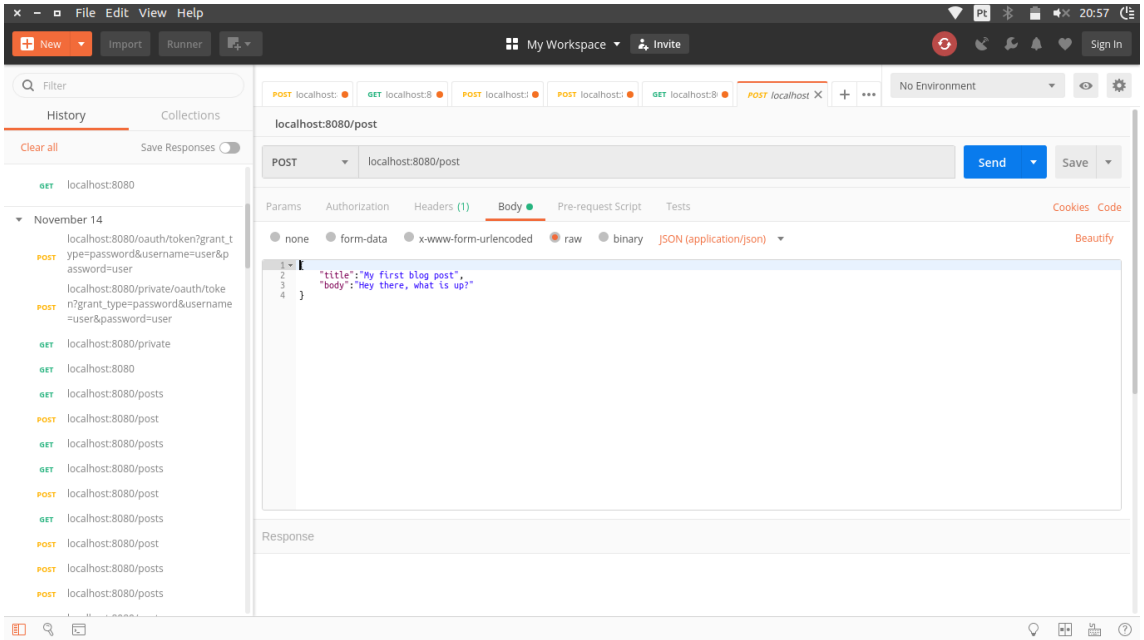
Figura 3 – Projeto de *front-end* criado usando o *framework Javascript Vue.JS*



2.7 *Postman*

É uma ferramenta (também disponível como extensão para o *browser Google Chrome*) que permite realizar requisições e testes *HTTP* utilizando uma interface gráfica mais amigável ao usuário. Tem como o objetivo de extrair o máximo de informações e depurar serviços *REST*. Possui um ambiente para a documentação, execução e teste de APIs (DEWAILLY, 2015). A Figura 3 exibe uma tela do aplicativo usado para testar as requisições.

Figura 3 – Interface para teste de requisições do *Postman*



2.8 Descrição do Hardware Utilizado

Foi utilizado um *laptop* com processador Intel Core I5 da marca Acer, com 12Gb de memória RAM, 1 terabyte de armazenamento e sistema operacional GNU/Linux Ubuntu 16.04 LTS, *Java* versão 8, *Spring Framework* versão 5, *Spring Boot* versão 2.1.0.

3. RESULTADOS E DISCUSSÃO

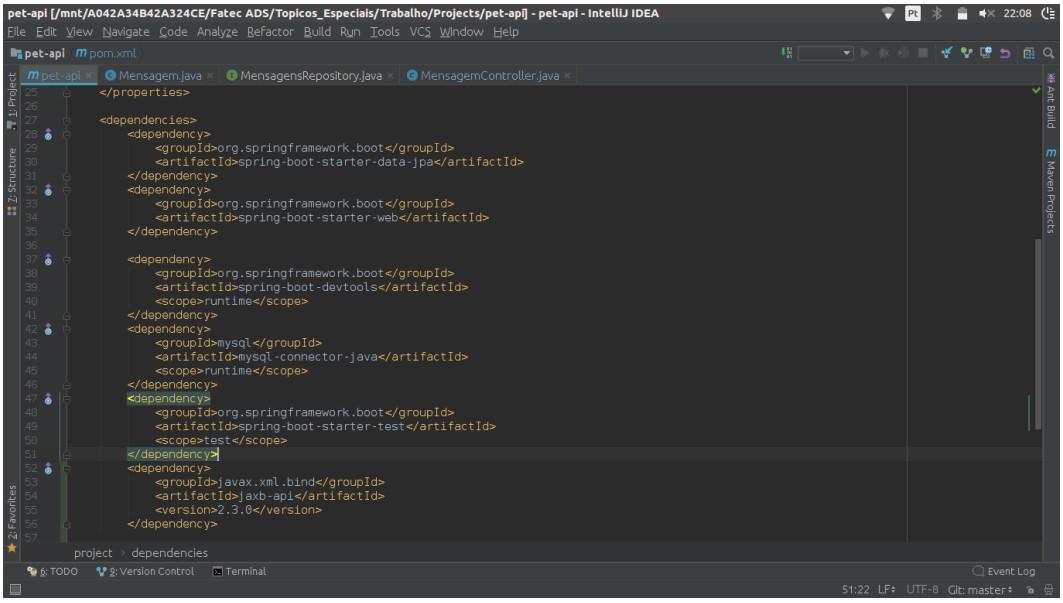
A partir do levantamento de requisitos para o desenvolvimento do sistema, onde as necessidades foram levantadas, tendo uma primeira imagem do projeto. Uma estratégia foi traçada para que se tivesse um planejamento que guiasse as etapas de concepção do projeto. Nessa fase foram definidos os recursos utilizados, como linguagem de programação, sistema de banco de dados, padrões de interface e usabilidade.

Com o uso da linguagem de programação Java, que possui uma documentação extensa e uma grande coleção de funcionalidades e bibliotecas que visam atender um grande número de projetos diferentes, em ambientes diferentes. Além de contar um modelo de maturidade e alta compatibilidade com versões anteriores, como descrito.

Além do Java, também foi utilizado o *framework Spring*, que possui como objetivo simplificar a construção de uma infraestrutura que suporte e execute todo o projeto, evitando perda tempo com a escolha do servidor que rodará a aplicação, *pool* de conexões com o banco de dados, entre outros, dando um maior enfoque na regra de negócio do projeto.

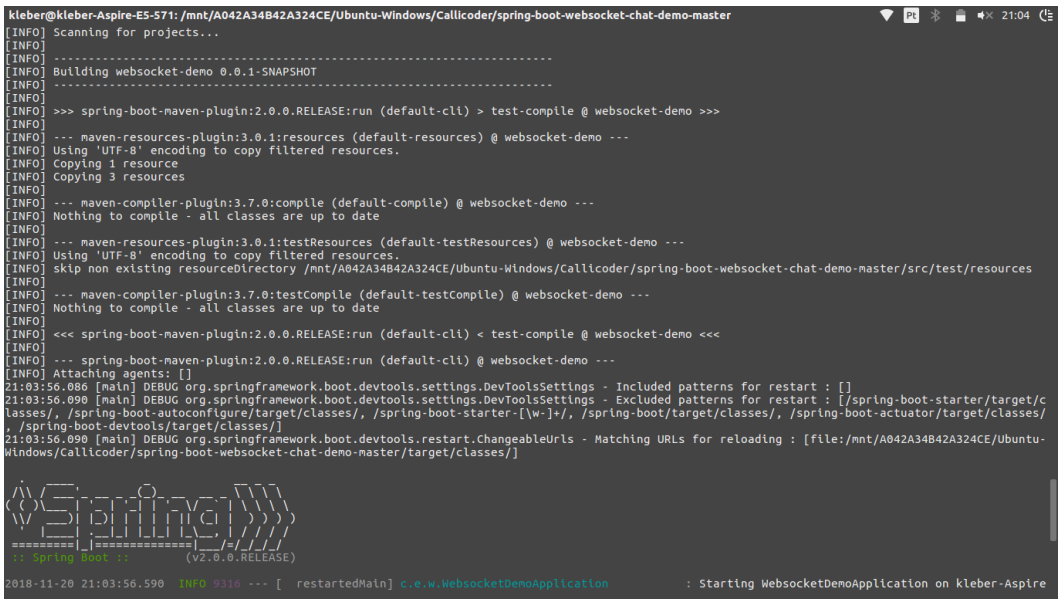
O processo construção da *API REST* usando o *Spring framework,* foi bastante simplificado, pois o básico desse *framework* já oferece recursos para a construção de uma aplicação robusta. Com a utilização de alguns de seus módulos, como o *Spring Boot*, o grau de automatização do *deploy* da aplicação foi alcançado de maneira satisfatória. O próprio *Spring Boot* se encarrega, em conjunto do gerenciador de pacotes *Maven*, de adicionar as bibliotecas necessárias ao projeto. Algumas delas, como a dependência *Web*, já incorpora ferramentas para a implementação de um projeto que utiliza recursos da *Web*, além de contar com um servidor integrado. A dependência *JPA* é a responsável pelo gerenciamento das ações referentes ao banco de dados do projeto, estabelecendo uma padronização para a persistência de dados. A dependência *Devtools* se encarrega de verificar se houve alteração no código da aplicação e caso haja, refaça a execução do sistema de maneira automática. Na Figura 4 pode ser visto o arquivo de configuração das dependências exigidas pelo projeto.

Figura 4 – Arquivo de configuração *pom.xml* (*Project Object Model* ou Modelo de Objeto do Projeto)



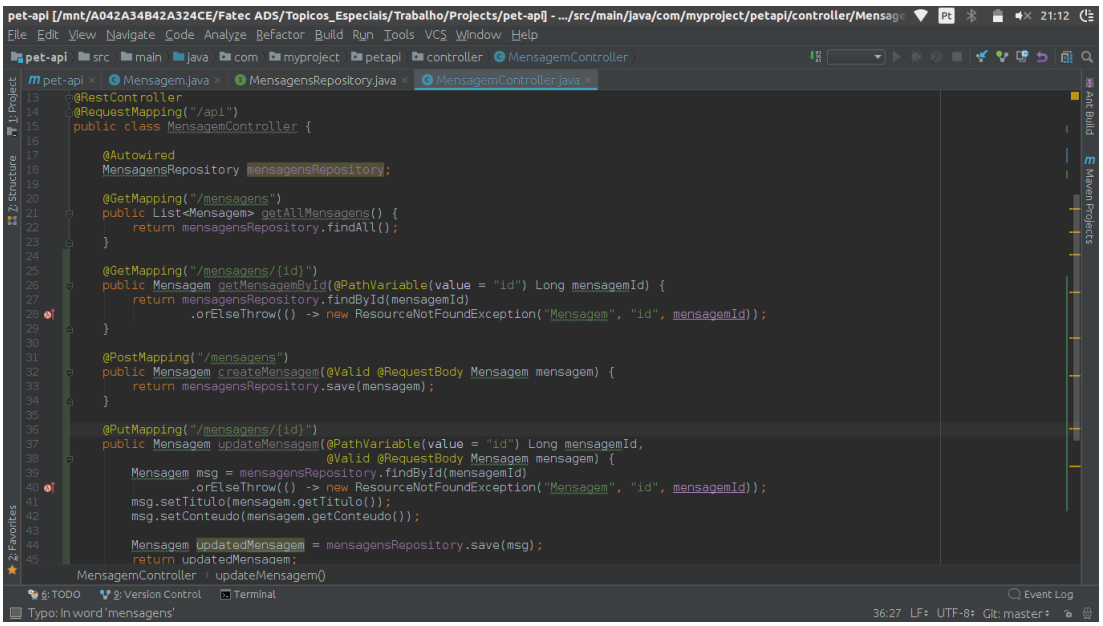
O *Spring* permite a utilização da ferramenta *Maven*, que é responsável por baixar as bibliotecas necessárias para o projeto, cuidando para que as versões especificadas no arquivo de configuração pom.xml sejam adicionadas. É possível atualizar todas as versões das bibliotecas com poucos ajustes, pois o *Maven* em conjunto com o *Spring* automatizam essa tarefa, economizando tempo na escolha de dependências para o projeto. A Figura 5 mostra parte do carregamento das dependências e execução de projeto

Figura 5 – Gerenciamento do projeto com o *Maven* e o *Spring Boot*



Ao utilizar a arquitetura *REST* como modelo, a aplicação passa a receber informações por meio de requisições *HTTP*, sendo que as mesmas podem alterar o estado dos dados enviados ao sistema. Cada ação é mapeada de acordo com o protocolo *HTTP*, permitindo que se consiga interagir com o sistema. As ações mais comuns são o *GET*, o *POST*, o *PUT* e o *DELETE*, cada qual possui seu conjunto de regras de utilização. Cada requisição é enviada para um *endpoint*, que tratara de receber e processá-las de maneira adequada. A Figura 6 exibe um exemplo de uma classe controladora, com os respectivos mapeamentos de cada ação.

Figura 6 – Classe *Java* com o *Controller* que usa *HTTP* para tratar as requisições.

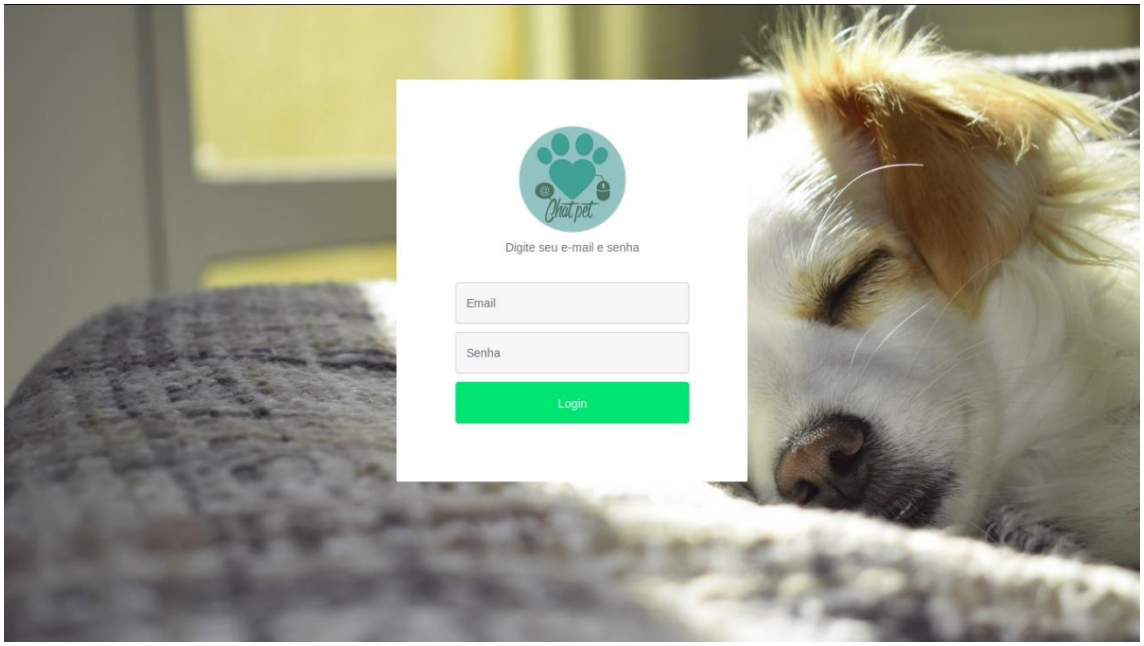


O conteúdo e é enviado no formato *JSON*, mas também existem os formatos *XML* e *YAML*.

O sistema recebe as informações vindas de uma página *web*, através do corpo de uma requisição *HTTP*, e é recebido pelo sistema, que a identifica e processa suas requisições.

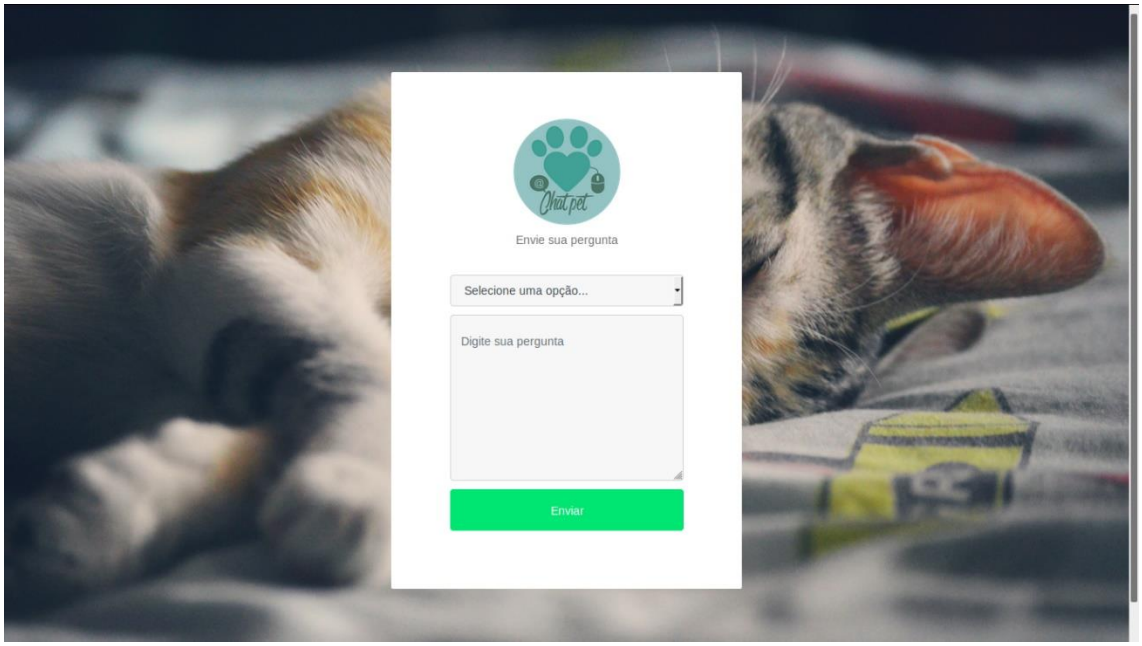
O *Vue.Js* fica responsável pela interface de *front-end* do sistema, pela qual o usuário fará a interação com a aplicação. Por ser um *framework* leve e rápido, o *Vue*, como também é chamado, permite que as páginas sejam carregadas de maneira rápida, além de permitir a criação de páginas dinâmicas, que sejam reativas. Página reativas alteram seu comportamento à medida que o usuário ou sistema interage com os seu conteúdo e recursos. Como exemplo, a lista de mensagens, ao receber ou remover dados, é atualizada instantaneamente, sem que haja a necessidade do carregamento da página inteira. Na Figura 7 é exibida a tela de login para acesso ao site.

Figura 7 – Tela de login do site



O usuário consegue efetuar o envio de mensagens através de uma página própria, que contém um formulário renderizado pelo navegador. Há vários mecanismos de validação incluídos que exibem mensagens alertando se há campos sem preenchimento, evitando que a mensagem seja enviada de maneira incompleta, tornando complicado o envio da mensagem e sua futura resposta. A Figura 8 mostra a tela de envio de mensagens.

Figura 10 – Tela para envio de mensagens



A página de visualização permite que o usuário da aplicação possa ver se há novas mensagens disponíveis e ao clicar em uma delas é redirecionado para a página de visualização de mensagem. Aqui é possível conferir a mensagem em sua totalidade e adicionar novas respostas a mesma.

Como consequência da criação do sistema, pode-se oferecer um melhor atendimento ou até mesmo antecipar-se a certos imprevistos, sendo mais proativo no atendimento. Tais experiências positivas ajudam a fidelizar o cliente, possibilitando a comunicação rápida e eficiente, além trazer mais clientes e aumentar a taxa de fidelização e satisfação dos donos de animais de estimação.

4. CONCLUSÕES

A aplicação construída seguindo o modelo *REST* permite que a mesma possa ser utilizada por qualquer dispositivo ligado à rede. O site permite que o cliente possa se *logar*, enviar e checar se suas mensagens foram recebidas.

O processo de envio de mensagens fica facilitado, através de uma interface limpa, de fácil utilização. Através da aplicação, fica estabelecida uma linha de comunicação rápida e confiável entre cliente e um especialista (veterinário), com o objetivo de tornar mais claras as informações, de maneira relevante para ambas as partes.

A clínica que utiliza o site consegue atrair mais clientes, pois o atendimento de qualidade sempre é a maior propaganda. Agilidade na comunicação, em conjunto com a comunicação rápida e efetiva ajudam a fidelizar o cliente e facilitam o atendimento, gerando benefícios para a clínica, sendo mais um beneficio que pode colocar a mesma a frente de seus concorrentes.

REFERÊNCIAS

BOAGLIO, Fernando. **Spring Boot:** Acelere o desenvolvimento de microsserviços. São Paulo: Casa do Código, 2017.

DEITEL, P.; DEITEL, H. **Java Como Programar.** 10. ed. São Paulo: Pearson, 2017

DEWAILLY, Ludovic. **Building a RESTful Web Service with Spring.** Birmingham: Packt, 2015.

FIELDING, R. T. **Representational State Transfer (REST).** 2000a. Disponível em: <https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm>. Acesso em: 14 nov. 2018.

DIAS, Estevão. **As boas partes do Java 11.** 2018. Disponível em: <https://www.devmedia.com.br/as-boas-partes-do-java-11/40193>. Acesso em: 22 nov. 2018.

GOOGLE Developers - **Product Index.** 2018. Disponível em: <https://developers.google.com/products/>. Acesso em: 09 nov. 2018.

OTTERO, R. **Artigo Introdução ao Maven**. 2012. Disponível em: <https://www.devmedia.com.br/introducao-ao-maven/25128>. Acesso em: 10 Nov. 2018.

Spring Framework Overview. 2018. Disponível em: <https://docs.spring.io/spring/docs/current/spring-framework-reference/overview.html>. Acesso em: 01 set. 2018.

Welcome to Apache Maven. 2018. Disponível em: <https://maven.apache.org/>. Acesso em: 10 nov. 2018.

YE, J. J. ***Building Applications with Spring 5 and Vue.js 2*.** Birmingham: Packt Publishing, 2018.

DIRETRIZES PARA AUTORES
REVISTA TEKHNÉ E LOGOS (FATEC BOTUCATU)

1. SUBMISSÃO DOS TRABALHOS

Deverá ser encaminhada uma declaração de anuência, com nome completo, endereços institucionais e e-mails e as assinaturas de todos os autores, bem como o nome do autor indicado para correspondência, a qual será anexada em "documentos suplementares" no portal da Revista Tekhne e Logos.

O trabalho deve ser acompanhado, se for o caso, de uma declaração de conflito de interesses na qual conste o tipo de conflito.

Todas as instituições patrocinadoras da pesquisa devem ser mencionadas no trabalho.

Toda pesquisa envolvendo seres humanos ou animais deve ter aprovação prévia do Comitê de Ética da instituição de origem. Nesses casos, o número do protocolo no Comitê de Ética deve ser mencionado no trabalho.

As normas da Revista Tekhne e Logos podem sofrer alterações, portanto não deixe de consultá-las antes de fazer a submissão de um artigo. Elas são válidas para todos os trabalhos submetidos neste periódico.

Lembre-se que SE as normas da revista não forem seguidas rigorosamente, seu trabalho não irá tramitar

2. FORMA E PREPARAÇÃO DOS MANUSCRITOS

Na primeira versão do artigo submetido, os nomes dos autores e a nota de rodapé deverão ser omitidos. Somente na versão final o artigo deverá conter o nome de todos os autores com identificação em nota de rodapé

O manuscrito submetido para publicação deverá digitado em processador de texto em formato DOCX, encaminhado via eletrônica (http://www.fatecbt.edu.br/seer) obedecendo as especificações a seguir:

Papel: Formato A4

Espaçamento do texto: em coluna simples, com espaço entre linhas de 1,5

Margens: 3,0 cm de margens esquerda e superior e margens direita e inferior com 2,0 cm, orientação retrato

Fonte: Times New Roman, tamanho 12.

Parágrafos: 1,25 cm.

Número de páginas: até 15 (quinze) páginas, numeradas consecutivamente, incluindo as ilustrações.

Tabelas: devem fazer parte do corpo do artigo e ser apresentadas no módulo tabela do Word. Essas devem ser elaboradas apenas com linhas horizontais de separação no cabeçalho e ao final das mesmas, evitando o uso de palavras em negrito e coloridas, as quais devem ser ajustadas automaticamente à janela. O título deve ficar acima e centralizado. Se o trabalho for redigido em inglês ou espanhol, deve vir também redigido em português. Exemplo de citações no texto: Tabela 1. Exemplos de citações no título: Tabela 1. Investimento econômico-financeiro (sem ponto no final após o texto). O título deve ficar acima e centralizado, redigido na fonte Times New Roman, tamanho 12. Em tabelas que apresentam a comparação de médias, segundo análise estatística, deverá haver um espaço entre o valor numérico (média) e a letra. As unidades deverão estar entre parêntesis.

Gráficos, Figuras e Fotografias: devem ser apresentados em preto e branco ou em cores (se necessário), nítidos e com contraste, inseridos no texto após a citação dos mesmos, com resolução de 300 dpi. Se o trabalho for redigido em inglês ou espanhol, deve vir também redigido em português. Exemplo de citações no texto: Figura 1. Exemplos de citações no título: Figura 1. Investimento econômico-financeiro (sem ponto no final após o texto). O título deve ficar acima e centralizado, redigido na fonte Times New Roman, tamanho 12(doze).

Fórmulas: deverão ser feitas em processador que possibilite a formatação para o programa Microsoft Word, sem perda de suas formas originais e devem ser alinhadas à esquerda e numeradas sequencialmente à direita

Nomes científicos: devem ser escritos por extenso e em itálico.

3. ESTRUTURA E ORGANIZAÇÃO

3.1 ARTIGO ORIGINAL

O artigo deve ser apresentado na seguinte sequência:

Título: no idioma português com no máximo, 15 (quinze) palavras em letras maiúsculas e em negrito

Título: no idioma inglês com, no máximo, 15 (quinze) palavras em letras maiúsculas e em negrito.

Autores: até 5 (cinco), por extenso, posicionados logo abaixo do título em inglês ou em português (a depender do idioma do trabalho), com chamada para nota de rodapé da primeira página, com as seguintes informações: formação, titulação e instituição a que o autor está filiado, seguido do endereço, CEP, cidade, estado e endereço de e-mail, sem nenhuma sigla.

Resumo: apresentando em folha à parte, deve condensar, em um único parágrafo, o conteúdo, expondo objetivos, materiais e métodos, os principais resultados e conclusões em não mais do que 250 palavras. A palavra RESUMO devem ser redigida em letras maiúsculas e centralizada.
Palavras-chave: no mínimo de 3 (três) e no máximo de 5 (cinco) termos. Não devem repetir os termos que se acham no título, podem ser constituídas de expressões curtas e não só de palavras e devem ser separadas por ponto em ordem alfabética.
Abstract: além de seguir as recomendações do resumo, não ultrapassando 250 palavras, deve ser uma tradução próxima do resumo. A palavra ABSTRACT devem ser redigida em letras maiúsculas e centralizada.
Key words: representam a tradução das palavras-chave para a língua inglesa.
Introdução: Deve ocupar, preferencialmente, no máximo duas páginas, apresentando o problema científico a ser solucionado e sua importância (justificativa para a realização do trabalho), e estabelecer sua relação com resultados de trabalhos publicados sobre o assunto a ser pesquisado. O último parágrafo deve expressar o objetivo, de forma coerente com o constante no Resumo. Esta seção não pode ser dividida em subtítulos.
Material e Métodos: Esta seção pode ser dividida em subtítulos, indicados em negrito. Deve ser redigida com detalhes para que o trabalho possa ser repetido por outros pesquisadores, evidenciando e referenciando a metodologia empregada para a realização da pesquisa e da informação sobre os métodos estatísticos e as transformações de dados.
Resultados e Discussão: Podem ser divididas em subseções, com subtítulos concisos e descritivos. O texto dos Resultados e discussões devem ser discutidos e interpretados à luz da literatura, não apresentando os mesmos resultados das tabelas e figuras.
Conclusões: não devem ser vastas e discursivas, sendo necessário apresentá-las com coerência aos objetivos propostos. Deve ser capaz de evidenciar a solução de seu problema por meio dos resultados obtidos.

3.2 ARTIGOS DE REVISÃO

Os artigos de revisão bibliográfica deverão conter: Título (português e inglês), resumo com palavras-chave e abstract com keywords. Introdução; Desenvolvimento do assunto com discussão que deverão ser apresentados em tópicos; Considerações finais e Referências. Deverão conter no máximo 15 páginas.

As demais normas são as mesmas utilizadas para artigos originais.

Agradecimentos: facultativo.

4. CITAÇÕES NO TEXTO

As citações de autores no texto são conforme os seguintes exemplos:

a) Joaquim (2005) ou (JOAQUIM, 2005)

b) Joaquim e Silva (2010) ou (JOAQUIM; SILVA, 2010)

c) Havendo mais de três autores, é citado apenas o sobrenome do primeiro, seguido de et al. (não itálico): Rossi et al. (2008) ou (ROSSI et al., 2008).

5. REFERÊNCIAS

No artigo deve existir no mínimo dez (10) referências

Devem seguir a NBR 6022, 6021, 6023, 10520, 6028, 6024 da ABNT. Recomenda-se que 70% das referências tenham sido publicadas nos últimos 5 anos e também que 50% sejam de periódicos científicos, apresentadas da seguinte maneira:

a) Artigo de periódico: SIMÕES, D.; SILVA, R. B. G.; SILVA, M. R. Composição do substrato sobre o desenvolvimento, qualidade e custo de produção de mudas de *Eucalyptus grandis* Hill ex Maiden × *Eucalyptus urophylla* S. T. Blake. **Ciência Florestal**, Santa Maria, v. 22, n. 1, p. 91-100, jan./mar. 2012. Disponível em: <http://dx.doi.org/10.5902/198050985082>. Acesso: 21 jan. 2014.

b) Livro: MACHADO, C. C.; LOPES, E. S.; BIRRO, M. H. B. **Elementos básicos do transporte florestal rodoviário**. Viçosa: UFV, 2005. 167p.

c) Capítulo de livro: NOGUEIRA, E. Análise de investimentos. In: BATALHA, M. O. (Org.) **Gestão Agroindustrial**. 5. ed. São Paulo, SP. Atlas, 2009. p. 205-266.

d) Dissertação e Tese: MACHADO, R. R. **Avaliação do desempenho logístico do transporte rodoviário de madeira utilizando Rede de Petri**.75f. Dissertação (Mestrado em Ciência Florestal) apresentada a Universidade Federal de Viçosa/ MG. 2006. Disponível em: <http://www.tede.ufv.br/tedesimplificado/tde_arquivos/4/TDE-2006-11-06T144815Z-43/Publico/texto%20completo.pdf>. Acesso em: 21 ago. 2013.

e) Trabalhos de congressos: SILVA, R. M.; BELDERRAIN, M. C. N. Considerações sobre diagrama tornado em análise de sensibilidade. In: ENCONTRO LATINO AMERICANO DE INICIAÇÃO CIENTÍFICA, 8., 2004, São José dos Campos. **Anais...** São José dos Campos, SP: UNIVAP, 2004. p. 8-11.

f) Trabalhos de conclusão de curso ou monografias: não aceitos.

Condições para submissão

Como parte do processo de submissão, os autores são obrigados a verificar a conformidade da submissão em relação a todos os itens listados a seguir. As submissões que não estiverem de acordo com as normas serão devolvidas aos autores.

1. A contribuição é original e inédita, e não está sendo avaliada para publicação por outra revista; caso contrário, deve-se justificar em "Comentários ao editor".
2. O arquivo da submissão está em formato Microsoft Word ou OpenOffice ambos com extensão DOCX.
3. O manuscrito está editado em coluna simples, com espaço entre linhas de 1,5, fonte Times New Roman, tamanho 12, tabulação de 1,25 cm, formato A4, com 3,0 cm de margens esquerda e superior e margens direita e inferior com 2,0 cm, orientação retrato e máximo de 15 páginas.
4. Existe documento suplementar que comprove a anuência dos coautores para a publicação do artigo.
5. Caso a pesquisa envolva seres humanos ou animais, a mesma tem aprovação prévia do Comitê de Ética da instituição de origem e esse documento será submetido como documento suplementar.
6. URLs para as referências foram informadas quando possível.
7. O texto segue os padrões de estilo e requisitos bibliográficos descritos em Diretrizes para Autores, na página Sobre a Revista.

Política de Privacidade

Os nomes e endereços informados nesta revista serão usados exclusivamente para os serviços prestados por esta publicação, não sendo disponibilizados para outras finalidades ou a terceiros.