UNIVERSIDADE FEDERAL DO PIAUÍ - UFPI
CURSO DE BACHARELADO EM SISTEMAS DE INFORMAÇÃO

IMPLEMENTAÇÃO DE MELHORIAS DE COMUNICAÇÃO E IHC NO SISTEMA DE CONSULTA E APOIO A DECISÃO – SCAD, USANDO MECANISMOS DE *REALTIME*.

ABIMAEL SANTIAGO VELOZO

PICOS – PIAUÍ
2015

ABIMAEL SANTIAGO VELOZO

IMPLEMENTAÇÃO DE MELHORIAS DE COMUNICAÇÃO E IHC NO SISTEMA DE CONSULTA E APOIO A DECISÃO – SCAD, USANDO MECANISMOS DE *REALTIME.*

Monografia submetida ao Curso de Bacharelado de Sistemas de Informação como requisito parcial para obtenção de grau de Bacharel em Sistemas de Informação.

Orientador: Prof. Ivenilton Alexandre de Souza Moura

PICOS – PIAUÍ
2015

FICHA CATALOGRÁFICA
Serviço de Processamento Técnico da Universidade Federal do Piauí
Biblioteca José Albano de Macêdo

V443i Velozo, Abimael Santiago
Implementação de melhorias de comunicação e IHC no sistema de consulta e apoio à decisão-SCAD, usando mecanismos de realtime / Abimael Santiago Velozo . – 2015.
CD-ROM : il.; 4 ¾ pol. (67 f.)

Monografia(Bacharelado em Sistemas de Informação) – Universidade Federal do Piauí, Picos, 2015.
Orientador(A): Prof°.Ivenilton Alexandre de Souza Moura
1. Processos de Software. 2. Sistemas Web. 3. Engenharia de requisitos 2. I. Título.

CDD 005

IMPLEMENTAÇÃO DE MELHORIAS DE COMUNICAÇÃO E IHC NO SISTEMA DE CONSULTA E APOIO À DECISÃO – SCAD, USANDO MECANISMOS DE *REAL TIME*

ABIMAEL SANTIAGO VELOZO

Monografia APROVADA como exigência parcial para obtenção do grau de Bacharel em Sistemas de Informação.

Data de Aprovação

Picos – PI, 26 de JUNHO de 2015

Prof. Esp. Ivenilton Alexandre de Souza Moura
Orientador

Prof. Esp. Leonardo Pereira de Sousa
Membro

Prof. Esp. Éder Jânio Francisco Gomes
Membro

DEDICATÓRIA

Aos meus pais Abraão José Velozo e Deusimar de Carvalho Santiago Velozo, por sempre me apoiarem e confiarem no meu potencial.

AGRADECIMENTOS

A Deus por ter me dado saúde e forças para superar os desafios.

A minha família, pelo apoio e incentivo. Em especial meus pais e irmãos, que apesar de todas as dificuldades, me apoiaram em todos os momentos e me fortaleceram para conseguir meus objetivos.

A meu orientador Ivenilton Alexandre, pela orientação e ensinamentos, me fazendo pensar de forma mais abrangente e detalhada.

A todos os professores do curso de sistemas de informação, que ajudaram na expansão dos meus conhecimentos.

Aos meus colegas e amigos de curso, que me ajudaram muito, principalmente no início, contribuindo com suas experiências e conhecimentos.

Obrigado a todos!

*“*O sucesso é ir de fracasso em fracasso sem perder o entusiasmo*.”*

Winston Churchill

RESUMO

O Sistema de Consulta e Apoio à Decisão (SCAD) foi desenvolvido por estudantes do curso de sistemas de informação da Universidade Federal do Piauí-Campus Picos, para resolver problemas na gestão de realização de reservas de espaços físicos, reserva de veículos e requerimentos de diárias. O presente trabalho tem como objetivo, realizar a implementação de um serviço de notificações em tempo real no SCAD. E o desenvolvimento de um chat que funcionará em tempo real, para comunicação entre os usuários do SCAD. O serviço de notificações realizará o envio de mensagens de aviso para os usuários, quando ocorrer novos cadastros ou atualizações nos dados do sistema. Os usuários do SCAD precisam atualizar as páginas para visualizar alterações realizadas nas informações. Com isso, pode acontecer de ocorrer mudanças nos dados e os usuários não perceber, podendo assim realizar operações com dados que foram alterados. O envio das notificações é importante para que os operadores do sistema estejam sempre atualizados sobre os acontecimentos com os dados do SCAD. O chat proporcionará comunicação em tempo real entre os utilizadores do sistema, permitindo assim a troca de informações sem a necessidade de locomoção dos utilitários do sistema.

Palavras-chave: Sistema *Web*, Tempo Real, *Chat*, Responsividade, Comunicação

ABSTRACT

The Consultation System and Decision Support (SCAD) was developed by students of the course of information from the University of Piauí-Campus Picos systems to solve problems in the management of carrying out physical spaces reservations, car reservations and daily requirements . This paper aims, carry out the implementation of a real-time notifications of service at SCAD. And the development of a chat that works in real time for communication between users of SCAD. The notification service will perform sending warning messages to users when there is new registrations or updates to system data. SCAD users need to update the pages to view changes made to the information. Thus, it can happen to be changes in the data and users do not realize and can thus perform operations with data that has changed. The sending of notifications is important for system operators are up to date on the events with the SCAD data. The chat will provide real-time communication between users of the system, thus allowing the exchange of information without the need for transportation of system utilities.

Keywords*: Web System, Real Time, chat, responsiveness, communication*

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

ASD	Desenvolvimento de *Software* Adaptativo
API	*Application Programming Interface*
CASE	*Computer-Aided Software Engineering*
CSS	*Cascading Style Sheets*
DNS	*Domain Name System*
FTP	*File Transfer Protocol*
HTML	*HyperText Markup Language*
HTTP	*Hypertext Transfer Protocol*
HTTPS	*Hyper Text Transfer Protocol Secure*
IHC	Interface Humano Computador
MVC	Model-view-controller
PSQL	PostgreSQL
SCAD	Sistema de Consulta e Apoio a Decisão
SCAEF	Sistema de Controle de Alocação de Espaços Físicos
SCAT	Sistema de controle e alocação de transportes
SCSD	Sistema de controle e solicitação de diárias
SQL	*Structured Query Language*
TCP	*Transmission Control Protocol*
UDP	*User Datagram Protocol*
UFPI	Universidade Federal do Piauí
XML	*e**X**tensible **M**arkup **L**anguage*
XMLHTTP	*Extensible Markup Language / Hypertext Transfer Protocol*
XMLHTTPREQUEST	*Extensible Markup Language / Hypertext Transfer Protocol*
XP	*Extreme programming*

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

A construção de uma aplicação *web* não é um trabalho unidimensional, desenvolvedores *web* modernos são obrigados a utilizarem uma série de tecnologias para construir aplicativos que atendam às necessidades dos usuários (LENGSTORF; LEGGETTER, 2013).

Os sistemas computacionais estão presentes em diversas áreas do nosso cotidiano e têm evoluído nos últimos anos. Essa evolução tem exigido novas soluções de *software* que possam interagir com o *hardware* da forma mais eficiente possível. Um dos motivos do aumento no uso de tecnologias se deve aos sistemas *web,* que possibilitam que trabalhos sejam realizados em diferentes áreas geográficas, precisando apenas de acesso à *internet*.

Dentre as várias atividades envolvidas na *internet*, o desenvolvimento de *sites* e sistemas *web* tem ganhado um grande desempenho, por ser uma atividade relativamente recente que desperta grande interesse e o desenvolvimento não apresenta um alto grau de complexidade.

Em muitas situações o desenvolvimento de aplicações *web* tem custo menor que de *desktops*. Podendo essas serem integradas com sistemas *desktops* já existentes, colocando eles para compartilhar seus dados com a *web*.

Aplicações *web* possuem muitas vantagens, umas das principais é permitir o acesso de qualquer lugar do planeta onde possa ter conexão com à *internet.* Com isso surge outras vantagens, pois torna possível o uso desses sistemas para realizar trabalhos sem a necessidade de ficar em um lugar fixo. Mas esses sistemas ainda possuem algumas características que podem se tornar problemas em algumas ocasiões de uso.

Uma característica dos sistemas *web* com potencial para ser um problema, é a necessidade de atualização das páginas onde os usuários visualizam alterações feitas nas suas informações. Essas atualizações não serão percebidas sem que realize atualização da página. Tal característica pode fazer com que operadores de sistemas utilizem dados que já foram atualizados ou removidos do sistema, causando assim problemas graves na realização das operações.

Com a evolução das tecnologias, surgiram novas ferramentas que permitem o desenvolvimento de rotinas computacionais dinâmicas. Exemplo disso são os mecanismos de *realtime* e as linguagens de programação dinâmicas, que possibilitam

o desenvolvimento de aplicações em tempo real. Com isso, é possível informar interativamente aos usuários o que está acontecendo no sistema, diminuindo assim a possibilidade de erros no uso desses dados.

1.1 Objetivos

Este trabalho tem como objetivo desenvolver um serviço de notificações em tempo real, que avisa aos usuários do sistema quando forem realizadas alterações nos dados e criados novos cadastros. E desenvolver também um *chat* que funciona em tempo real, para comunicação entre os usuários do SCAD.

1.2 Organização do Trabalho

Após a introdução que relatou sobre as vantagens e possíveis problemas dos sistemas *web* e o objetivo do projeto, serão apresentadas a seguir os próximos capítulos, que descrevem o desenvolvimento das funcionalidades do projeto e as tecnologias utilizadas.

- Capítulo 2 – Referencial teórico: fornece fundamento teórico para o trabalho. Neste capitulo são mostrados conceitos relacionados com Sistemas *web,* Sistemas *web* em tempo real e tecnologias utilizadas para desenvolvimento desse projeto.
- Capitulo 3 – Especificação do Sistema: Será mostrado o trabalho proposto, as funcionalidades desenvolvidas, como foi a utilização das ferramentas e os recursos que elas disponibilizaram.
- Capitulo 4 – Resultados e Discussões: Serão apresentadas as questões resolvidas com esse trabalho, o funcionamento dos serviços desenvolvidos e explicadas as funcionalidades com textos e mostrando imagens da execução do sistema.
- Capitulo 5 – Considerações Finais: Apresenta a conclusão do trabalho.

2 REFERENCIAL TEÓRICO

2.1 Engenharia de *Software*

Sommerville (2007) explica que praticamente todos os países, hoje em dia, dependem de sistemas complexos baseados em computadores. Portanto, construir e manter *softwares* dentro de custos adequados é essencial para o funcionamento da economia nacional e internacional. Para conseguir alcançar os padrões de desenvolvimento de *software*, com um custo aceitável e facilidades na manutenção, os desenvolvedores utilizam uma ciência chamada engenharia de *software*. A engenharia de *software* é um ramo da engenharia cujo foco é o desenvolvimento dentro de custos adequados, de sistemas de *software* de alta qualidade. Contudo, a falta de restrições naturais dos *softwares* significa que eles podem facilmente se tornar muito complexos e com isso muito difíceis de serem compreendidos.

De acordo com Pádua (2000), a engenharia de *software* se preocupa com o desenvolvimento de *software* como criação de produtos. Assim, estão fora de seu escopo, programas que são feitos unicamente para diversão do programador. E também pequenos programas descartáveis, feitos por alguém exclusivamente como meio para resolver um problema, e que não serão utilizados por outros.

Utilizando processos da engenharia de *software*, os desenvolvedores podem criar *softwares* com fases bem definidas e com documentação bem elaborada. Com isso há uma maior facilidade no entendimento do funcionamento do sistema.

Pressman (2011) afirma que quando um *software* é bem construído, atende as necessidades dos usuários, opera perfeitamente durante um longo período, é fácil de modificar e fazer manutenção e, mais fácil ainda de utilizar. Assim, ele é realmente capaz de mudar as coisas para melhor.

2.1.1 Processos de *Software*

Os processos de *softwares* procuram assegurar o desenvolvimento de *software* com elevada produtividade (de forma econômica), com definição de prazo e recursos e com qualidade assegurada.

Em engenharia de *software*, os processos podem ser definidos para atividades como: desenvolvimento, manutenção, aquisição e contratação de *software*. Pode-se

também definir subprocessos para cada um destes. O processo de desenvolvimento abrange subprocessos de determinação dos requisitos, análise, desenho, implementação e testes (FILHO, 2000, p.23).

Os *softwares* estão sendo utilizados em larga escala pelas mais diversas áreas de trabalho. Esses *softwares* desde os mais simples até os mais complexos passam por a fase do processo de desenvolvimento.

Pressman (2011) explica que, construir *software* é um processo de aprendizado social iterativo e o resultado, é a incorporação do conhecimento coletado, filtrado e organizado conforme se desenvolve o processo. Mas o que é o processo de *software*? É a metodologia para atividades, ações e tarefas necessárias para desenvolver um *software* de alta qualidade. Assim, o processo de *software* é diferente da engenharia de *software*, pois ele define a abordagem adotada conforme um *software* é elaborado pela engenharia. Já a engenharia de *software* engloba também, tecnologias que fazem parte do processo, como métodos técnicos e ferramentas automatizadas.

2.1.1.1 Modelos de processo

Um modelo de processo genérico para engenharia de *software* estabelece cinco atividades metodológicas, são elas: comunicação, planejamento, modelagem, construção e entrega. E um conjunto de atividades de apoio, que são aplicadas durante o processo, sendo elas: acompanhamento e controle do projeto, a administração de riscos, a garantia da qualidade, o gerenciamento das configurações, as revisões técnicas e outras.

Os modelos de processo prescritivos (ou tradicionais), foram propostos para proporcionar um padrão no desenvolvimento de *software*. Esses modelos recebem a denominação “prescritivos”, pois prescrevem um conjunto de elementos de processo, como atividades especificas do método, ações de engenharia de *software*, tarefas, produtos, garantia da qualidade, controles e mudanças para cada projeto.

A seguir será apresentado alguns dos modelos de processos prescritivos:

- Modelo Cascata (ou ciclo de vida clássico): propõe uma abordagem sequencial, pois executa as ações de desenvolvimento uma após a outra; e sistemática, pois mantém a mesma sequência de passos sempre que é aplicada para o desenvolvimento de *software*. Nesse modelo, inicialmente é realizado o levantamento de requisitos desejados pelo cliente, depois a fase de

planejamento, modelagem, construção, emprego e culminando na manutenção continua dos *softwares* desenvolvidos.

- Modelo Evolucionário: esse modelo baseia-se no desenvolvimento de uma versão inicial, levando o resultado para ser testado e comentando pelos usuários e, refinando esse resultado por meio da implementação de várias versões. Até o desenvolvimento de um sistema adequado para o cliente. Atividades como especificação, desenvolvimento e validação são interpostas, em vez de serem separadas, com *feedback* rápido que permeia as atividades.

Existem dois tipos importantes de desenvolvimento revolucionário, são eles:

1. Desenvolvimento exploratório: o objetivo é trabalhar com o cliente para explorar os requisitos e entregar uma versão final;
2. Prototipação *throwaway*: o objetivo é compreender as necessidades do cliente e, a partir disso, desenvolver a melhor definição de requisitos para o sistema.

Do ponto de vista da engenharia e do gerenciamento a abordagem evolucionaria possui dois problemas, são eles:

1. O processo não é visível: os gerentes precisam de acompanhamento regular para acompanhar o progresso;
2. Os sistemas são frequentemente mal estruturados: Com as mudanças frequentes, o *software* sofre com a desestruturação.

- Modelo incremental: Em um processo de desenvolvimento incremental, o cliente define os requisitos do sistema, de uma forma geral. Assim é identificado quais os serviços mais importantes e, então é definido a quantidade de incrementos do desenvolvimento, sendo que em cada incremento é entregue um subconjunto de funcionalidades do sistema.
- Modelo espiral: O processo é executado como uma espiral. Cada ciclo da espiral é considerado como uma fase do processo de desenvolvimento do *software*. Dessa forma, o ciclo mais interno pode estar relacionado à viabilidade do sistema, o próximo à definição dos requisitos, o próximo ao projeto do sistema e assim por diante. Cada *loop* na espiral está dividido em quatro áreas, são elas: definição de objetivos, avaliação e redução de riscos, desenvolvimento e validação e planejamento.

Os modelos prescritivos explicados anteriormente, se preocupam muito com o planejamento, projeto e documentação do *software*. Levando a um gasto considerável de tempo com essas ações.

Sommerville (2007) explica que os negócios atualmente operam em um ambiente global sujeito a rápidas mudanças. Eles têm de responder a novas oportunidades e mercados, mudanças na economia e ao surgimento de produtos e serviços concorrentes. Os sistemas computacionais (*softwares*) são partes de quase todas as operações de negócio e, assim, é essencial que um novo *software* seja desenvolvido rapidamente para aproveitar as novas oportunidades do mercado. Desenvolvimento e entregas rápidas são, portanto, muitas vezes o requisito mais importante para o desenvolvimento de sistemas de *software*. Mas na década de 1980 e início de 1990, havia uma visão geral de que a melhor forma de se obter o melhor *software* era por meio de um cuidadoso planejamento de projeto, garantia de qualidade formalizada, uso de métodos de análise e projetos apoiados por ferramentas CASE e controlados por um rigoroso processo de desenvolvimento de *software*. E o emprego dessa visão no desenvolvimento, faz com que este leve um tempo superior ao necessário no mercado moderno.

Para resolver essa questão, na década de 1990 desenvolvedores de *software* propuseram os métodos ágeis. Estes permitiam que a equipe de desenvolvimento se concentrasse no *software* somente, em vez da documentação e projeto do sistema.

Portanto, os métodos ágeis não devem ser utilizados para desenvolvimento de sistemas críticos nos quais é necessária uma análise detalhada de todos os requisitos e funcionalidades de segurança.

A seguir será apresentado alguns dos métodos ágeis:

- *XP* (*Extreme programming*): é uma metodologia de desenvolvimento de *software*, que proporciona a criação de sistemas de melhor qualidade, que são produzidos em menos tempo e de forma mais econômica que o habitual. Esses objetivos são alcançados através de um conjunto de valores, princípios e práticas, que diferem de forma substancial da maneira tradicional de desenvolver *softwares*. Nessa metodologia, todos os requisitos são expressos como cenários, que são implementados diretamente como uma série de tarefas. Os programadores trabalham em pares e antes da implementação do código, eles desenvolvem formas de testes para cada tarefa. A *XP* trabalha com uma abordagem extrema, para o desenvolvimento interativo. Novas versões do *software* podem ser compiladas várias vezes ao dia, sendo que aproximadamente a cada duas semanas é entregue novas implementações do *software*.

- Desenvolvimento de *Software* Adaptativo (ASD): É uma técnica para construção de *software* e sistemas complexos. A base filosófica do Desenvolvimento de *Software* Adaptativo se centraliza na colaboração humana e na auto-organização das equipes.
- *Scrum*: é uma metodologia de desenvolvimento ágil de *software*, usado para orientar atividades de desenvolvimento dentro de um processo que concentra as seguintes atividades: requisitos, análise, projeto, evolução e entrega.

2.1.2 Engenharia de Requisitos

Pressman (2011) define a engenharia de requisitos como sendo o amplo espectro de tarefas e técnicas que levam a um entendimento dos requisitos. Na perspectiva do processo de *software*, a engenharia de requisitos é uma ação de engenharia de *software* importante que se inicia durante a atividade de comunicação e continua na de modelagem. Ela deve ser adaptada às necessidades do processo, do projeto, do produto e das pessoas que estão realizando o trabalho.

O termo requisito pode ser usado em duas situações. Em alguns casos, um requisito é apenas uma declaração abstrata de alto nível de um serviço que o sistema deve oferecer a seus usuários. No outro extremo, é uma definição formal e detalhada de uma função do sistema. Alguns problemas que surgem durante o processo de engenharia de requisitos são resultantes da falta de separação entre esses diferentes níveis de descrição. Então para fazer a distinção dos requisitos, pode se usar requisitos de usuário para indicar requisitos abstratos e de alto nível e requisitos de sistema para descrição detalhada do que o sistema deve fazer (SOMMERVILLE, 2007).

Com a explicação sobre engenharia de *software* e seus métodos, podemos observar as possibilidades disponibilizadas para suporte no desenvolvimento de sistemas de diferentes tipos.

2.2 Sistemas *Web*

Durante alguns anos a *internet* foi usada tradicionalmente para apresentar conteúdos estáticos. Os *sites* eram basicamente uma estrutura de entidades estáticas,

pertencentes a uma única coleção de dados. Mas isso mudou nos últimos anos, com o avanço das tecnologias já é possível o desenvolvimento de sistemas *web* dinâmicos.

Segundo Pauli (2013), aplicação *web* é qualquer *software* baseado em *web* que realize ações de acordo com uma entrada de usuários e que normalmente interaja com sistemas de *backend*.

Os sistemas *web* são desenvolvidos para proporcionar agilidade e mobilidade na utilização dos recursos da *internet.* Eles são projetados para serem executados em um servidor *web* utilizando o protocolo *HTTP* (*Hypertext Transfer Protocol*), para realizar o envio de requisições entre cliente (navegador) e servidor (onde é executado as tarefas). Esses sistemas são apresentados através de um navegador *web,* podendo assim ser acessado a qualquer hora, de qualquer dispositivo computacional com recursos para esse tipo de acesso e em qualquer lugar do mundo onde se tenha uma conexão com a *internet*, permitindo que haja uma maior integração entre os usuários desses sistemas.

Os sistemas *web* são desenvolvidos dentro das seguintes etapas: especificação de requisitos, análise dos requisitos, implementação e definição das funções do sistema. Nessas fases é definida a forma de funcionamento do sistema (sistema estático ou dinâmico).

Sistemas *web* dinâmicos proporcionam a atualização de dados progressivamente, sem a necessidade de alterações no código e estrutura da aplicação. Eles são normalmente conduzidos por algum tipo de banco de dados. Isso proporciona, de fato, um alvo fácil para a administração do sistema e um lugar para armazenar dados.

De acordo com Lengstorf & Leggetter (2013), com o desenvolvimento de sistemas *web* dinâmicos o foco muda do servidor para o cliente, fazendo-o realizar mais trabalho, como recuperar e carregar conteúdos dinamicamente e mostrar os resultados em tempo real, com base em entradas de dados dos usuários do sistema.

2.3 Sistemas *Web* em Tempo Real

A utilização de um sistema *web* proporciona muitas vantagens, não é necessário a instalação de *softwares*, nem o armazenamento de dados na máquina do usuário, além da comodidade, já citada anteriormente, pois é possível o acesso a essas aplicações em qualquer lugar do planeta, bastando apenas ter acesso a

internet. Mas apesar de tantas vantagens, eles ainda apresentam algumas características quem podem ser entendidas como desvantagens em algumas ocasiões de trabalho.

Colemam & Greif (2014), explicam que se abrirmos dois navegadores *web,* executarmos uma página da mesma aplicação neles e, então realizar uma alteração nas informações da página de um dos navegadores, não será visível na do outro, a não ser que recarregue a página.

Nas aplicações *web* tradicionais quando a página é atualizada ou o usuário acessa o sistema e realiza algum trabalho, é enviado requisições ao servidor. Esse modo de trabalho pode causar problemas se o número de solicitações ao servidor aumentar significativamente e o sistema e seu ambiente de execução não estiverem preparados para esse aumento.

Rai (2013) informa que na *web* fomos habituados a *sites* e aplicativos que quando clicamos em um *link* a página é alterada, ou seja, carregada. E só assim novas informações cadastradas são buscadas para serem apresentadas aos usuários. Aplicações que disponibilizam informações em tempo real, sem a necessidade do envio de requisições ao servidor, têm sido tradicionalmente uma coisa difícil de alcançar e para chegar a isso muitos programadores tem usado de soluções não adequadas.

Os sistemas que funcionam em tempo real são diferentes dos outros sistemas de *software*. O funcionamento correto desses sistemas depende da capacidade de resposta deles a eventos, dentro de um curto intervalo de tempo.

Sommerville (2007) define sistemas em tempo real, como sendo *softwares* cujo funcionamento correto depende dos resultados produzidos pelo sistema e do tempo em que estes resultados são produzidos.

Os sistemas em tempo real normalmente utilizam processos assíncronos, ou seja, realizam a execução de tarefas simultaneamente, sem a necessidade de bloqueios de processos no servidor.

Pereira (2013), explica que sistemas *web* desenvolvidos sobre plataformas como: *.NET, Java, Python, PHP* ou *Ruby,* funcionam de forma síncrona, ou seja, as requisições são enviadas para uma fila de requisições no servidor por um sistema bloqueante e, todas as requisições são enfileiradas e esperam para serem processadas, sendo que o processamento é individual. Assim cada requisição em espera, fica um período de tempo ociosa. Com o aumento de acessos no sistema, à

frequência de gargalos em alguns casos serão mais frequentes, aumentando a necessidade de fazer *upgrade* nos *hardwares* dos servidores.

Com esses problemas e com o custo de manutenção e de atualização dos *hardwares* bem alto, aumentou a busca por soluções em tecnologias que façam o bom uso de *hardware*, que utilize o máximo de poder de processamento dos processadores, não o mantendo ocioso quando realiza operações do tipo bloqueante.

Com isso, surgiram as tecnologias de *realtime,* que possibilitam o desenvolvimento de aplicações interativas e em tempo real. Essas ferramentas permitem a implementação de rotinas em sistemas *web* que enviam informações em tempo real do servidor para o cliente. Algumas das ferramentas mais populares para conseguir tais objetivos são: a plataforma *node.js*, a linguagem de programação *JavaScript*, o *Ajax, framework Ruby On Rails* e a linguagem de programação *Ruby.*

2.3.1 Node.js

Pereira (2013) explica que o *Node.js* foi criado em 2009 por Ryan Dahl com a ajuda inicial de 14 colaboradores. Essa tecnologia foi criada para resolver o problema de tarefas do tipo bloqueantes, que paralisam um processo enquanto utilizam um I/O (entrada e saída) no servidor. Trata-se de uma plataforma altamente escalável e de baixo nível, pois a programação é realizada diretamente com diversos protocolos de rede e *internet* ou utilizando bibliotecas que acessam recursos do sistema operacional, principalmente recursos de sistemas baseados em *Unix*.

> O *Node.js* é uma plataforma construída sobre o *Runtime JavaScript* do *chrome* para construção de aplicações de rede, escaláveis e rápidas. O *Node.js* trabalha com o modelo não-bloqueante para I/O, que o torna leve e eficiente, perfeito para aplicações em tempo real de dados intensivos que são executados através de dispositivos distribuídos (CANTELON; HARTER; HOLOWAYCHUK; RAJLICH, 2014, p. 4, tradução nossa).

O *JavaScript* é a linguagem de programação do *Node.js* e isso foi possível graças à *engine JavaScript V8,* a mesma utilizada no navegador *Google Chrome*.

O *Node.js* fornece uma plataforma orientada a eventos e assíncrona para *server-side JavaScript*. Traz *JavaScript* para o servidor da mesma maneira que que um navegador traz *JavaScript* para o cliente. No *Node.js* a execução de tarefas de entrada e saída acontece de forma assíncrona e não bloqueante, pois trabalha com *event-loop*, que é um ciclo que busca os eventos vindos de interações no navegador

e os processa simultaneamente (CANTELON; HARTER; HOLOWAYCHUK; RAJLICH, 2014, tradução nossa).

Essa forma de trabalho citada acima permite que várias interações sejam realizadas no sistema sem que ocorra sobrecarga no servidor. Não há uma fila de espera para as requisições serem processadas individualmente, elas são processadas de forma assíncrona.

Segundo Pereira (2013), *Node.js* é orientado a eventos, ele segue a mesma filosofia de orientação a eventos do *JavaScript* cliente-*side*; a única diferença é que não existem eventos de componentes *HTML* (*HyperText Markup Language*), ele trabalha com eventos de I/O do servidor, como por exemplo: o evento *connect* de um banco de dados, um *open* de um arquivo, um *data* de um *streaming* de dados e muitos outros. O *event-loop* é o agente responsável por executar e emitir eventos no sistema. Na prática ele é um *loop* infinito que a cada interação verifica em sua fila de eventos se um determinado evento foi emitido. Quando ocorre, é emitido um evento. Ele o executa e envia para fila de executados. Quando um evento está sendo executado, pode-se programar qualquer lógica dentro dele, isso graças aos eventos do *JavaScript*.

Texeira (2013) explica que o *Javascript* contribuiu muito para o sucesso do *Node.js* e cita alguns motivos para o uso do mesmo:

- Facilidade no uso: O *Node.js* é fácil de instalar e configurar;
- *Javascript*: A linguagem *Javascript* é largamente utilizada em todo o mundo, o que possibilita uma melhor adaptação ao *Node*.
- O *Node* é simples: Os desenvolvedores do *Node* buscaram tornar o mais simples possível o desenvolvimento utilizando a plataforma.

Existem alguns *frameworks* que facilitam o desenvolvimento com *Node.js*, são eles:

- *Express JS:* Este é o *framework web* mais popular para *Node.js*. Usando um conjunto robusto de recursos, os desenvolvedores podem criar, *multi-page single*, e aplicações *web* híbridos.
- *SAILS JS*: é um excelente *framework* para desenvolvimento com *Node.js*. Ele facilita o desenvolvimento de aplicativos empresariais e outros aplicativos modernos.

- *TOTAL JS*: é outro grande *framework* para *Node.js* que ajuda a criar páginas *web* e aplicações *web*. Ele também suporta a arquitetura *MVC*. Este é um *framework* moderno de código aberto para a construção de *sites* utilizando *Node.js*, *HTML, JavaScript e CSS*.
- *PARTIAL JS*: é um *framework* para *Node.js, HTML, CSS e JavaScript*. Com ele, os desenvolvedores podem construir em larga escala aplicações *web* e *sites*. Suas características e arquitetura são os mesmos que *total.js*.
- *KOA JS*: Este *framework* de *Node.js* é projetado pela equipe do *Express*. *Koa.js* é uma ferramenta de nova geração, que tem como objetivo ser uma base mais expressiva e mais robusta para aplicações *web* e APIs.
- *LOCOMOTIVE JS*: *Locomotive* suporta padrão *MVC*, rotas *RESTful*, e baseia-se também no *Express*. Este é uma das *frameworks web* mais poderosas para *Node.js*. Suas características incluem:

1. Convenção sobre configuração;
2. Encaminhamento de *Helpers*;
3. Arquitetura *MVC*; e
4. Conexão com qualquer banco de dados.

O desenvolvimento de aplicações *realtime* utilizando o *Node.js* com esses *frameworks* é tecnicamente a criação de uma conexão bidirecional, que, na prática, é uma conexão que se mantém aberta para clientes e servidores interagirem em um único canal. A uma grande vantagem nessa forma de trabalho, pois a interação no sistema será em tempo real, trazendo uma experiência de usuário muito melhor que os sistemas *web* sem a aplicação de *realtime.*

É através do protocolo *WebSockets* que o *Node.js* mantém conexões abertas. Esse protocolo permite criar conexões persistentes tanto no cliente como no servidor. Pereira (2013) explica que o *Node.js* se tornou muito popular por disponibilizar além desse, outros protocolos que funcionam em bibliotecas de baixo nível. Alguns exemplos desses protocolos são: *Hypertext Transfer Protocol* (*HTTP*), *Hyper Text Transfer Protocol Secure (HTTPS), File Transfer Protocol (FTP)*, *Domain Name System (DNS)*, *Transmission Control Protocol (TCP), User Datagram Protocol (UDP)*.

O protocolo *WebSockets* não é compatível com todos os navegadores, isso torna inviável o desenvolvimento de aplicações *cross-browser* usando essa tecnologia.

2.3.1.1 *Socket.io*

Para resolver o problema de incompatibilidade do *WebSocket* com alguns navegadores, surgiu o *Socket.io*. Ele emula por exemplo, *ajax long-polling* ou outros transportes de comunicação em navegadores antigos que não possuem suporte para *WebSockets*.

Pereira (2013) explica que com o *Socket.io* é incluído um *script* no cliente que detecta informações sobre o navegador do usuário para definir qual será a melhor forma de comunicação com o servidor. Se o navegador utilizado possuir suporte para recursos de *WebSockets* ou *FlashSockets (*utilizando *Adobe Flash Player* do navegador), será realizada uma comunicação bidirecional, se não, é emulada uma conexão bidirecional que em curtos intervalos de tempo faz requisições ao servidor, usando a tecnologia *AJAX.*

Os *transports* de comunicação que o *Socket.io* executa são:

- *WebSocket;*
- *Adobe Flash Socket;*
- *Ajax long polling;*
- *AJAX multipart streaming;*
- *Forever iframe;*
- *JSONP Polling;*

O *Socket.io* facilita o trabalho dos programadores, pois toda decisão complexa é ele que faz. Isso torna seu uso muito simples.

O *Socket.io* fornece *APIs* para *broadcast,* envio de mensagens em tempo real e muito mais. Essas características fizeram dele muito popular para construção de jogos de navegador, aplicativos de bate-papo e aplicações de *streaming* (CANTELON; HARTER; HOLOWAYCHUK; RAJLICH, 2014, tradução nossa).

2.3.1.2 Execuções de rotinas assíncronas versus rotinas síncronas

Pereira (2013) apresenta um exemplo da execução de duas funções criadas usando o *Node.js.* Uma com execução síncrona e a outra assíncrona. A seguir será exibido o resultado da execução.

Foram criados dois arquivos, o primeiro com o código síncrono, que possui um *loop* de um até cinco e o segundo com o código assíncrono com um *loop* igual ao anterior.

Quando esses códigos foram executados os resultados foram os apresentados nas figuras 1 e 2:

1. O código síncrono demorou 1000 ms para executar o *loop*. Observando a figura 1, podemos perceber que cada ciclo foi executado em períodos diferentes, levando 200 ms para cada ciclo de execução.

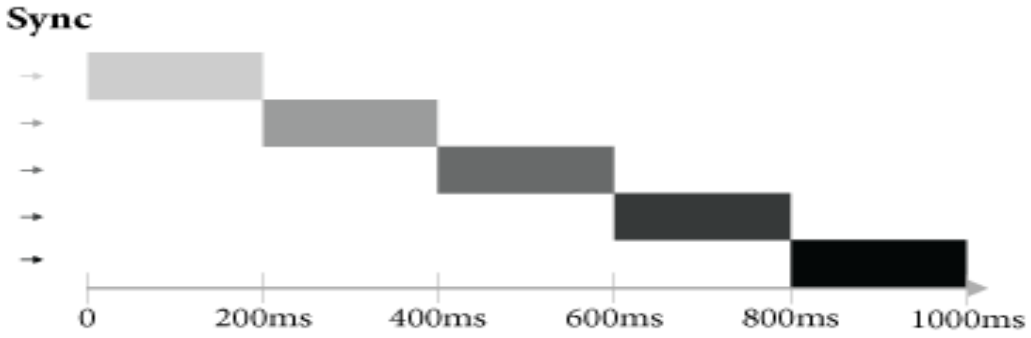


Figura 1- *Timeline* síncrona bloqueante

Fonte: Pereira (2013).

2. O código assíncrono demorou 200 ms para executar o *loop*. Verificando a figura 2, podemos perceber que a execução foi realizada de forma simultânea. Diminuindo assim em 800 ms o tempo de execução em relação ao primeiro código.

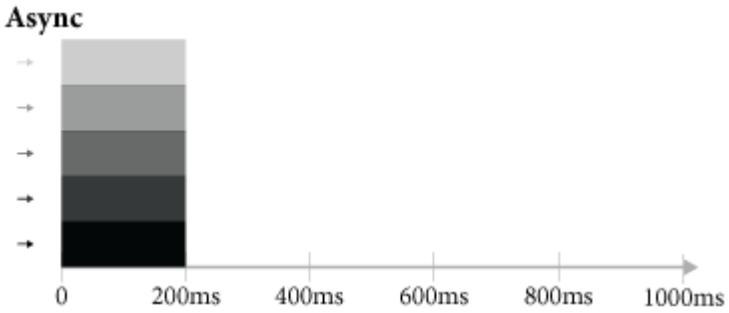


Figura 2- *Timeline* assíncrona não-bloqueante

Fonte: Pereira (2013).

Com o que foi apresentado, podemos perceber que com uso do *Node.js* é possível o desenvolvimento de um sistema *web* que atenda às necessidades de

qualidade, evitando a carência da utilização de *hardware* muito avançado, já que essa tecnologia busca tornar simples a execução das funções dos sistemas por ela gerado. O *Node.js* permite desenvolver sistemas que disponibilizam dados em tempo real para os usuários, diminuindo o tempo gasto em cada trabalho realizado e aumentando a simplicidade nas requisições realizadas em cada interação feita por usuários no navegador.

2.3.2 JavaScript

Silva (2010) explica que *JavaScript* foi criada pela *Netscape* em parceria com a *Sun Mycrosystems,* com a finalidade de fornecer um meio de adicionar interatividade as páginas *web.* A primeira versão, denominada *JavaScript* 1.0, foi lançada em 1995 e implementada em março de 1996 no navegador *Netscape Navigator* 2.0 quando o mercado era dominado pela *Netscape*.

JavaScript é uma linguagem de programação para *web.* Muitos *websites* modernos usam essa linguagem e todos os navegadores *web* modernos, *desktops, tablets* e telefones inteligentes incluem interpretadores *JavaScript*. Tornando o *JavaScript* a linguagem de programação mais onipresente na história. *JavaScript* é uma linguagem de alto nível, dinâmica, e possui tipagem dinâmica (Flanagan, 2011, tradução nossa).

Balduino (2013) informa que quando Brendan Eich desenvolveu a primeira versão do *JavaScript* para o navegador *Mozilla*, em 1995, a ideia era que a linguagem tivesse uma sintaxe parecida com *Java*, pegando emprestado até mesmo alguns objetos e métodos com nomes iguais. O próprio nome, *JavaScript*, foi uma jogada de *marketing* para que a linguagem pudesse pegar carona no sucesso do *Java*. Quebrando a resistência das pessoas de aprender uma nova linguagem. Mesmo assim, somado a isso uma série de escolhas ruins de *design*, praticamente dois terços da linguagem acabaram ignorados, seja pela resistência em aprender uma nova linguagem ou pelo fato de programação para *web* não ser popular na época.

Atualmente, com o crescimento da utilização de sistemas *web,* aumentou também a ocorrência do uso do *JavaScript* no desenvolvimento desses sistemas.

Com os aprimoramentos realizados na linguagem, os desenvolvedores passaram a utiliza-la para criação de páginas dinâmicas junto com *HTML* (*HyperText Markup Language*).

O *JavaScript* disponibiliza vários eventos que podem ser usados para executar ações de acordo com a manipulação da página pelo usuário. Alguns exemplos desses eventos são: eventos de clique, eventos para alteração de conteúdo de texto ou de imagem e erros de carregamento de páginas.

Os eventos mencionados anteriormente, são atividades realizadas pelo usuário, que são responsáveis por dar origem à uma interatividade na página. São os gerenciadores de eventos que interligam a ação do usuário com o evento a ser disparado. Desses eventos, o clique do *mouse* é o único gerenciado pelo *HTML.* Mas o *JavaScript* disponibiliza meios para combinação de funções e métodos a eventos, tornando-os possíveis de serem manipulados dentro do código *HTML.* Esses eventos e demais métodos do *JavaScript* são executados no lado da aplicação cliente.

Silva (2010) explica que o *JavaScript,* é uma linguagem desenvolvida para trabalhar no lado do cliente, isto é, a interpretação e o funcionamento da linguagem dependem de funcionalidades instaladas no lado do cliente (navegador).

Segundo David (2011), *JavaScript* é uma linguagem dinâmica, orientada a objeto, possui estilos de programação funcional e deriva da sintaxe da linguagem de programação *Java.* Além de ser uma linguagem de propósito geral, robusta e eficiente. *JavaScript* permite manipular e controlar o comportamento de páginas *web* utilizando manipuladores de eventos (um manipulador de evento é uma função do *JavaScript* que é invocada através de alguma ação do usuário ou quando o navegador é carregado).

Códigos *JavaScript* podem ser criados ou chamados dentro de páginas *HTML*, bastando apenas utilizar seus seletores, para selecionar ids (identificadores) e classes do *HTML*. Assim é possível indicar a parte de código que deve ser manipulada pelo *JavaScript*.

JavaScript é uma linguagem de *scripts* (*script* é uma sequência de instruções, que são interpretados ou executados por um outro programa e não pelo processador do computador), que possibilita adicionar novas funções às páginas *web.*

Com o uso de *JavaScript* é possível fazer com que muito do trabalho que o servidor realiza para os sistemas seja transferido para o navegador, pois essa linguagem manipula vários recursos no lado do cliente.

2.3.2.1 jQuery

A linguagem de programação *JavaScript* apresenta falta de compatibilidade de alguns de seus métodos com navegadores antigos. Para resolver esse problema foi desenvolvido uma biblioteca *JavaScript* chamada *jQquery*.

Balduino (2013) explica que resolver esse problema de incompatibilidade com os diversos navegadores não seria fácil, pois são muitos. Mas alguém já criou uma biblioteca cheia de comandos para resolver os casos mais comuns no desenvolvimento *web,* que é a biblioteca *jQuery. jQuery* simplifica a manipulação de documentos *HTML*, eventos, animações e interações com *AJAX (**A**synchronous **J**avascript **a**nd **X**ML)* para desenvolvimento rápido de aplicações *web.* Devido à sua simplicidade e flexibilidade, o *jQuery* acabou se tornando a biblioteca mais utilizada do *JavaScript.*

jQuery se tornou rapidamente uma habilidade fundamental para desenvolvedores de *interface*. Com ela é possível manipular elementos da página através de métodos chamados seletores, que identificam cada elemento desejado pelo programador no documento *HTML*.

A *jQuery* disponibiliza ainda métodos que permitem a redução e reutilização de códigos, validação e interação de conteúdos de páginas *HTML*, além de possibilitar o desenvolvimento *cross-browser*.

Com isso podemos entender que sua principal função da biblioteca *jQuery,* é facilitar a adição de comportamento dinâmico às páginas *web,* de forma relativamente simples*.*

2.3.3 AJAX

De acordo com Niederauer (2013), a palavra *Ajax* vem da expressão *Asynchronous JavaScript and XML*. O *Ajax* é o uso sistemático de *JavaScript* e *XML* (entre outras tecnologias) para tornar as páginas *web* mais interativas com o usuário, utilizando-se de requisições assíncronas de informações. Isso significa dizer que podemos fazer solicitações de informações ao servidor *web* sem que seja necessário recarregar as páginas que estamos acessando.

Balduino (2013) explica que antigamente, cada vez que o usuário quisesse atualizar alguma informação na sua página, por menor que fosse, era necessário

atualizar toda a página. Isso significa que cada clique fazia o navegador carregar uma nova página inteiramente, carregando cabeçalhos, rodapés, conteúdo em comum e as vezes até mesmo imagens, *javascripts,* e *css.* Um trabalho desgastante tanto para o cliente quando para o servidor. Tudo começou a mudar com o lançamento do *Internet explorer 5*, que implementava um objeto *ActiveX* chamado *XMLHTTP*. Apesar de não ter chamado muita atenção na época foi logo implementado em outros navegadores através do objeto *JavaScript XMLHttpRequest*. O objeto *XMLHttpRequest* permite que o documento *HTML* acesse páginas, arquivos e informações atualizadas sem a necessidade de carregar a página inteira.

Um sistema desenvolvido usando *AJAX*, explora muito bem os recursos que o navegador disponibiliza, já que o *JavaScript* é uma linguagem que funciona utilizando as funções do lado do cliente. Além disso, a busca de dados no servidor, sem a necessidade de recarregar páginas inteiras, possibilita maior rapidez e economia de banda de *internet* utilizada no servidor. Principalmente se o código da página for complexo.

2.3.4 Ruby On Rails

De acordo com Minetto (2007), *framework* de desenvolvimento é um suporte de onde se pode desenvolver algo maior e mais especifico. É uma coleção de códigos-fontes, classes, funções, técnicas e metodologias que facilitam o desenvolvimento de softwares.

O *framework Ruby On Rails* foi criado por David Heinemeier Hansson no ano de 2003. Começou a ganhar mais atenção em 2006, por parte da comunidade de desenvolvimento *web*. “O *rails* foi criado pensando na agilidade e praticidade que ele proporcionaria na hora de escrever aplicativos *web*” (CAELUM, 2014, 39).

De acordo com Fuentes (2012), o *Ruby On Rails* é um *framework* que usa a arquitetura *MVC* (*Model, View, Controller*), bastante conveniente para desenvolvimento de aplicativos *web.* O *Rails* já é usado por muitos desenvolvedores a bastante tempo, portanto já foi testado em diversas situações, como alta carga, ou grande número de usuários. É fácil começar com o *Rails*, pois ele faz muito trabalho para o desenvolvedor. Mais o aprofundamento dará muito mais poder de desenvolvimento ao programador.

> O *MVC* é um padrão arquitetural onde os limites entre seus modelos, suas lógicas e visualizações são bem definidas, sendo muito mais simples fazer um reparo, uma mudança ou uma manutenção, já que essas três partes se comunicam de maneira bem desacoplada (CAELUM, 2014, 39).

O *Rails* é um meta-*framework,* composto por vários *frameworks*. São eles:

- *Active Record;*
- *Action Pack;*
- *Action Mailer;*
- *Action Support;*
- *Action Web Services;*

São esses *frameworks* que tornam possível a realização de operações entre a aplicação e o banco de dados, a criação de códigos *HTML, XML, JavaScript* e o controle de fluxo de regras de negócios.

Assim, desenvolver aplicações com o *Ruby On Rails* é um pouco mais complicado do que simplesmente criar páginas *HTML* simples, ele é preparado para a criação de aplicações modernas e arrojadas. Ele não somente responde *HTML* aos usuários, mas também responde de maneira adequada para aplicações ricas em *cliente-side,* interagindo com vários *frameworks.*

O *Ruby On Rails* proporciona a alternativa de trabalhar juntamente com várias outras tecnologias. Um exemplo disso é a possibilidade de desenvolver aplicações em tempo real e assíncronas com o *Node.js*. Sendo que este é executado dentro do projeto *Rails.* Além do *Node.js,* O *Rails* permite o desenvolvimento com outras tecnologias, como *JavaScript, Ajax, HTML, CSS* e etc.

Assim, este *framework* diminui o tempo de criação de sistemas *web.* Com a disponibilização de códigos prontos, fácil integração com bancos de dados, possui servidor próprio para testes locais, dentre outras vantagens.

2.3.5 Ruby

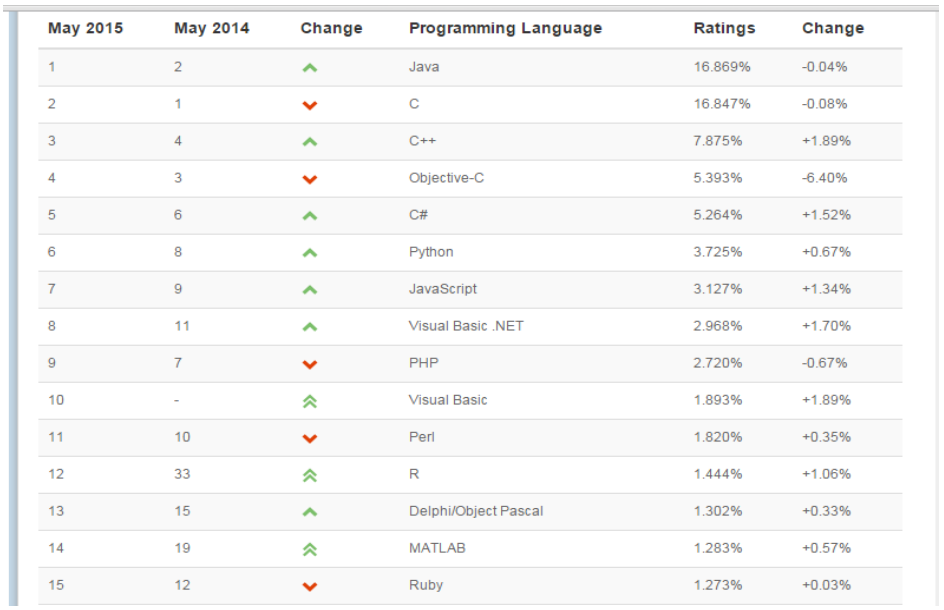
Souza (2012) explica que a linguagem de programação *Ruby* foi criada por Yukihiro Matsumoto, mas conhecido como Mats, no ano de 1995 no japão. O *Ruby* é uma linguagem de programação dinâmica, orientada a objeto e que possui algumas características funcionais. Ela foi criada para ser uma linguagem que juntasse

programação funcional e imperativa, mas acima de tudo que fosse uma linguagem legível. Esta é uma de suas grandes vantagens, ser extremamente legível.

> *Ruby* é uma linguagem interpretada, multiparadigma, o que permite ao programador lidar com vários estilos de paradigmas e possui tipagem dinâmica (CAELUM, 2014, 4).

De acordo com Souza (2012), a linguagem *Ruby* foi criada com inspiração em outras linguagens como *Perl, Smalltalk e Lisp*, e hoje está entre as linguagens mais usadas, muito em função da disseminação do seu principal *framework MVC*, o *Ruby on Rails.*

A figura 3 apresenta a tabela do *ranking* das linguagens de programação mais utilizadas atualmente. Nela podemos perceber a popularidade da linguagem *Ruby.*



May 2015	May 2014	Change	Programming Language	Ratings	Change
1	2	^	Java	16.869%	-0.04%
2	1	v	C	16.847%	-0.08%
3	4	^	C++	7.875%	+1.89%
4	3	v	Objective-C	5.393%	-6.40%
5	6	^	C#	5.264%	+1.52%
6	8	^	Python	3.725%	+0.67%
7	9	^	JavaScript	3.127%	+1.34%
8	11	^	Visual Basic .NET	2.968%	+1.70%
9	7	v	PHP	2.720%	-0.67%
10	-	^^	Visual Basic	1.893%	+1.89%
11	10	v	Perl	1.820%	+0.35%
12	33	^^	R	1.444%	+1.06%
13	15	^	Delphi/Object Pascal	1.302%	+0.33%
14	19	^^	MATLAB	1.283%	+0.57%
15	12	v	Ruby	1.273%	+0.03%

Figura 3 - *Ranking* de popularidade das linguagens de programação. Junho-2015

Fonte: http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html.

Com o que foi apresentado, podemos perceber que a linguagem *Ruby* juntamente com o *framework Ruby On Rails* possibilita o desenvolvimento de códigos legíveis e a separação de cada parte do código, deixando assim mais fácil a realização de manutenção, além de possibilitar um desenvolvimento ágil.

2.3.6 Frameworks de *Front-end Bootstrap*

Segundo Cochran (2012) o *Twitter Bootstrap* ou apenas *Bootstrap* foi criado em 2011 por Mark Otto e Jacob Thornton. Esse *framework* foi criado com o objetivo de facilitar a criação de *interfaces web*. Em 2012 o *Twitter Bootstrap* em sua versão 2.0

trouxe algumas melhorias, a principal delas foi a possibilidade da criação de *interfaces* responsivas, que adaptavam páginas para *desktops, tablets* e celulares. O *Twitter Bootstrap* rapidamente se tornou muito utilizado, com uma grande comunidade de usuários.

De acordo com Magno (2013), com o *Twitter Bootstrap* não é necessário o programador definir a partir do zero os elementos básicos, como botões e outros, na *interface* das páginas *web.* Ele disponibiliza vários elementos de *interface* de uso mais constante. Possui um *kit* de ferramentas, com convenções padrões de classes bem definidas, com uma limpa e prática documentação para os códigos que são disponibilizados prontos para serem usados e personalizados para as necessidades de cada usuário. Ele se adapta com muitos cenários de desenvolvimento, com isso os desenvolvedores estão aumentando seu uso junto com suas ferramentas.

O *Twitter Bootstrap* tem uma capacidade surpreendente, com sua grade de elementos para desenvolvimento responsivo, muitos componentes de *JavaScript* e um customizador de *interfaces web.* Ele é recomendado tanto para desenvolvedores experientes, como para iniciantes.

2.3.6.1 *Interfaces* responsivas

Magno (2013) explica que com o aumento de telefones inteligentes, há uma necessidade de desenvolver aplicações que disponibilizem conteúdo para essa crescente demanda, pois não é mais possível pensar em *desktop* como sendo a plataforma chave para ser utilizada como base para o desenvolvimento.

Um sistema *web* responsivo tem a capacidade de adaptar o conteúdo (textos, tabelas, gráficos) e o *layout* para ser exibido em diferentes tamanhos de telas de dispositivos. Essa característica, permite uma maior facilidade e mobilidade no uso do sistema, porque assim ele pode ser utilizado em qualquer equipamento computacional que o usuário tenha acesso, bastando apenas ter acesso a internet e a um navegador com compatibilidade aos recursos modernos da responsividade.

As primeiras abordagens para o uso da responsividade foi utilizando *plug-ins* que verificavam em qual tipo de dispositivo o sistema estava sendo executado. Assim, o sistema era redirecionado para páginas especificas para aquele tipo de aparelho, ou seja, uma versão do sistema desenvolvida com características adaptáveis para aquele tipo de máquina computacional especifica. Essa forma de desenvolvimento causava

muitos problemas, um exemplo, é a duplicação de conteúdo, quando ocorria falha na identificação do equipamento.

Mas com o desenvolvimento das tecnologias, se tornou possível o desenvolvimento de sistemas verdadeiramente responsivos. Eles se ajustam ao tamanho das telas onde são executados. Sendo que o conteúdo é sempre o mesmo para todos os dispositivos.

A seguir será apresentado algumas vantagens desses sistemas:

- Acessível para todo tipo de dispositivo;
- Um único sistema que se adapta automaticamente;
- Redimensionamento de textos, imagens, tabelas e outros componentes, para facilitar a visualização em tela pequenas;
- Evita conteúdo duplicado que prejudica os mecanismos de busca do sistema.

Como já foi mencionado anteriormente, o *Bootstrap* é uma tecnologia que permite o desenvolvimento responsivo. Ele disponibiliza métodos para tornar as páginas e seus componentes ajustáveis a todos os tipos e tamanhos de telas.

Com essas facilidades que o *Bootstrap* proporciona*,* o desenvolvimento de aplicações responsivas se tornou bem mais simples, com isso o aumento nesse tipo de desenvolvimento cresceu muito nos últimos tempos.

2.3.7 HyperText Markup Language

Tim Berners-Lee trabalhava na Seção de Computação da Organização Europeia de Pesquisa Nuclear (CERN), com sede em Genebra, na Suíça, quando iniciou pesquisas visando a descobrir um método que possibilitasse aos cientistas do mundo inteiro compartilhar eletronicamente seus textos e pesquisas (SILVA, 2011). O resultado dessas pesquisas foi o surgimento da *HyperText Markup Language.*

Silva (2011) explica que *HTML* é a sigla em inglês para *HyperText Markup Language,* que em português significa linguagem para marcação de hipertexto. Hipertexto é todo conteúdo inserido em um documento para a *web* e que tem como principal característica a possibilidade de se interligar a outros documentos da *web.* O que torna possível a construção de hipertextos são os *links*, presentes nas páginas dos *sites* que estamos acostumados a visitar quando navegamos na *internet*.

A *HTML* foi criada por Tim Berners-Lee e já evoluiu por sete versões, são elas:

- *HTML;*
- *HTML +;*
- *HTML 2.0;*
- *HTML 3.0;*
- *HTML 3.2;*
- *HTML 4.0;*
- *HTML 4.01;*
- *HTML5*

A evolução dessa linguagem trouxe alguns problemas, que tiveram que ser superadas com a introdução de outras tecnologias de suporte, capazes de melhorar o funcionamento de páginas *web.* Alguns exemplos dessas tecnologias, são: o *CSS (Cascading Style Sheets)* e *JavaScript.*

Silva (2011) explica que em 1995 assinalou o início de um desenvolvimento frenético de novas marcações para a linguagem *HTML,* com prioridade para criação de elementos e atributos de apresentação, discordando com o proposito inicial da linguagem, que era de ser uma ferramenta exclusivamente de marcação e estruturação de textos. Assim surgiu a versão 2.0 da linguagem *HTML.* Em março de 1995, Dave Raggett lançou sua proposta para a *HTML* 3.0, que vem com a primeira sugestão de uma marcação específica para estilização e apresentação, ao mesmo tempo que propõe a criação de novos atributos. Em 1997, o W3C endossou a *HTML* 3.2 como uma recomendação oficial. Nessa versão foi incorporado outros novos elementos, como elementos para marcar subscritos, sobrescritos e texto ao redor de imagens. Em julho de 1997, foi lançada a *HTML4* e*,* apenas em 2007 foram retornados os estudos para desenvolvimento de uma nova versão da linguagem, com isso foi criada a *HTML5.*

A *HTML* disponibiliza vários recursos para escrita de páginas para *web.* Através dela as páginas são apresentadas nos navegadores dos usuários e é possível criar e juntar textos, imagens e áudios em sistemas *web.*

2.3.8 *PostgreSQL*

De acordo com a Documentação do *PostgreSQL* 8.0.0 (2005), o *PostgreSql* é um sistema gerenciador de banco de dados objeto-relacional, baseado no

POSTGRES versão 4.2, desenvolvido pelo Departamento de Ciências da Computação da Universidade da Califórnia em Berkeley.

O *PostgreSQL* apresenta muitas funcionalidades modernas, como:

- Comandos complexos;
- Chaves estrangeiras;
- Gatilhos;
- Visões;
- Integridade transacional;
- Controle de simultaneidade multiversão.

O *PostgreSQL* é um banco de dados relacional, inteiramente caracterizado, *open source* e de uso livre. Ele pode ser usado praticamente com qualquer linguagem de programação moderna.

Ele é um banco de dados de grande porte, além de ser compatível com sistemas operacionais *windows, Linux* e outros. Possibilitando assim a sua utilização em projetos desenvolvidos para várias plataformas.

O *psql* é um dos principais recursos do *PostgreSQL.* Ele é um cliente no modo terminal, que permite digitar comandos interativamente, submetê-los para o *PostgreSQL* e ver os resultados da interação. Além disso disponibiliza vários comandos e muitas e variadas funcionalidades para facilitar a criação de *scripts* e automatizar a realização de muitas tarefas.

Com isso, podemos perceber que o *PostgreSQL* foi criado para oferecer vários recursos e assim proporcionar suporte a aplicações complexas e críticas, onde são necessários a criação de muitas tabelas e o armazenamento de grandes quantidades de dados importantes.

Com o uso das tecnologias esplanadas anteriormente, é possível o desenvolvimento de sistemas *web* que atendam às necessidades de qualidade satisfatórias para os usuários. Com a união dessas tecnologias é possível desenvolver aplicações que funcionam em tempo real, melhorando a forma como o usuário interage com o sistema e diminuindo as chances de erros na execução dos serviços. Elas dispõem de recursos para o desenvolvimento de sistemas modernos, de grande porte e com possibilidade para disponibilizar informações para muitos usuários simultaneamente. Sem que para isso ocorra problemas no funcionamento e desempenho da aplicação.

3 ESPECIFICAÇÃO DO SISTEMA

3.1 Método

Este capítulo apresenta as especificações da implementação de um serviço *web* de notificações em tempo real e o desenvolvimento de um *chat* utilizando ferramentas de *realtime,* no Sistema de Consulta e Apoio a Decisão-SCAD. Será mostrado os passos que foram executados para conclusão do desenvolvimento, levando em consideração cada tecnologia utilizada e os recursos que elas disponibilizaram.

O SCAD é formado pela integração de três sistemas, sendo eles: Sistema de Controle de Alocação de Espaços Físicos (SCAEF), que foi desenvolvido para melhorar a gestão de alocações de espaços físicos na Universidade Federal do Piauí - Campus Picos; Sistema de controle e alocação de transportes (SCAT), desenvolvido para melhorar a gestão das reservas de veículos no Campus; e o Sistema de controle e solicitação de diárias (SCSD), criado para ajudar na gestão de pedidos de diárias por profissionais do Campus.

Esses sistemas foram desenvolvidos por estudantes do curso de Sistemas de Informação da Universidade Federal do Piauí - Campus Picos. Para a implementação foram utilizados a linguagem de programação *Ruby* e o *framework Ruby On Rails*.

A versão inicial do SCAD atende as necessidades exigidas para a realização da gestão de atividades na UFPI. Mas foi verificado a possibilidade de implementação de novas funções que trariam melhorias significativas para o funcionamento do sistema. Esse projeto tem como objetivo desenvolver essas funções.

A princípio os responsáveis pelo desenvolvimento do sistema, pensavam na implementação de *realtime* nas páginas de apresentações de informações do SCAD. Com isso, quando atualizações fossem realizadas nos dados, esses dados modificados seriam apresentados em tempo real para os usuários. Mas depois de análise sobre como o sistema funcionaria com essa implementação, pode-se perceber que isso proporcionaria uma quebra no ritmo de trabalho dos usuários, pois a página poderia passar por transformações no meio de um trabalho, além de desviar a atenção dos mesmos com as mudanças.

Sendo assim, foi decidido que era mais produtivo desenvolver um serviço de notificações em tempo real, que avisa aos usuários do sistema quando for realizada

alterações nos dados e novos cadastros no sistema. Assim os operadores do sistema ficam atualizados sobre os dados que pertencem a cada um deles, sem que sejam atrapalhados na realização do trabalho no SCAD.

Para realização da atividade planejada, foram necessários estudos e pesquisas bibliográficas acerca das tecnologias fundamentais para o desenvolvimento do projeto:

3.1.1 Ruby On Rails

O *framework Ruby On Rails* possui recursos de integração com a plataforma *Node.js, a linguagem de programação JavaScript, o Ajax e o framework Bootstrap.* Assim, no desenvolvimento desse projeto, foram criadas funções utilizando essas tecnologias dentro do projeto já desenvolvido no SCAD. Sendo possível a utilização e integração de métodos novos com os já existentes.

3.1.2 Node.js

O *Node.js* disponibiliza uma grande quantidade de módulos que ajudam no desenvolvimento de rotinas de *realtime.* Além de possibilitar a comunicação do servidor com o cliente de forma rápida e assíncrona sem a necessidade de envio de requisições pelo cliente para o servidor*.* Assim, quando algum evento acontece, o servidor do *Node.js* envia informações em tempo real para o navegador, podendo assim atualizar os usuários sobre as ações realizadas no sistema. Essas características do *Node.js* foram utilizadas para criação do serviço de notificações. As mensagens são enviadas em tempo real para os clientes, quando um evento do servidor percebe alterações no banco de dados do SCAD.

3.1.2.1 *Socket.IO*

Para o envio em tempo real de dados entre o servidor e o cliente (navegador) é utilizado um módulo do *Node.js* chamado *Socket.io.* Esse módulo permitiu criar conexões para os clientes que realizam autenticação no SCAD. Persistindo essas conexões até o usuário fechar sua sessão de autenticação. Com isso é possível o envio de informações por meio da conexão aberta.

3.1.3 JavaScript

A plataforma *Node.js* utiliza a linguagem de programação *JavaScript,* mas diferentemente do princípio da linguagem (que é ser executada no lado do cliente), o *Node.js* a utiliza no lado do servidor*.* Essa linguagem disponibiliza vários métodos e eventos que são importantes para o trabalho em tempo real. Por esse motivo ela foi utilizada para desenvolvimento de rotinas no servidor *Node.js*.

3.1.3.1 jQuery

A biblioteca *jQuery* da linguagem *JavaScript* foi utilizada para manipular as informações disponibilizadas pelo servidor *Node.js.* Essa manipulação é realizada nas páginas do cliente.

3.1.4 Ajax

O *Ajax* foi utilizado para que partes de código de algumas páginas pudessem ser carregados, sem a necessidade de recarregar toda a página.

3.1.5 Linguagem de programação *Ruby*

A linguagem de programação *Ruby* foi utilizada para realizar cadastros de mensagens de notificações em tabelas do banco de dados, para essas mensagens serem enviadas para os clientes pelo servidor *Node.js.*

3.1.6 O Serviço de Notificações

O serviço *web* de envio de notificações em tempo real do servidor para o cliente, avisa aos usuários conectados no SCAD quando ocorre inserção, atualização, exclusão, aprovação e desaprovação de reservas de espaços físicos, alocação de veículos e pedidos de diárias por profissionais da UFPI, através de mensagens de notificação. Assim eles ficam atualizados sobre as informações presentes no sistema.

Essas notificações são enviadas por um canal que fica aberto entre o cliente e o servidor, sendo que esse canal é executado utilizando uma porta de conexão da

web. Através dessa porta é possível o servidor identificar o cliente e enviar dados por meio da conexão com ele.

As informações que são notificadas chegam até o canal de conexão através de eventos da linguagem *JavaScript,* que são disparados no momento exato que é observado mudanças em tabelas especificas do banco de dados. Com isso as informações são enviadas e apresentadas para os usuários em tempo real.

3.1.7 O *Chat*

O *chat* implementado tem o objetivo de possibilitar a comunicação entre os usuários do SCAD. Ele funciona em tempo real, permitindo que usuários mantenham contato sem sair do ambiente de trabalho, podendo assim trocar informações entre si e resolver problemas relacionados com as funções do sistema. Essa comunicação pode ser realizada sem que os usuários saiam da página onde estão executando suas tarefas, pois foi implementada uma função de receber e responder mensagens sem sair e sem recarregar a página atual. Assim não é necessário o usuário perder ou ter que parar de realizar o trabalho que está fazendo quando receber uma mensagem importante.

O *chat* foi desenvolvido pensando também em permitir que os usuários do sistema possam manter comunicação sem a necessidade de usar outros meios, como por exemplo, sistemas que possibilitam o envio de mensagens. Pois o uso de outra ferramenta causaria a interrupção da execução de suas tarefas.

3.1.8 Melhorias na IHC (Interface Humano Computador)

Foram realizadas algumas melhorias na IHC, com o objetivo de aperfeiçoar a apresentação das notificações para os usuários e também informações já existentes. Um exemplo dessas mudanças é a alteração realizada nas barras superiores do sistema, deixando-as fixas. É nessas barras onde ficam as opções de navegação pelo sistema para cada usuário e onde são apresentadas as notificações das informações do sistema e do chat. Antes as barras se moviam junto com a página, ficando escondidas na parte superior enquanto o usuário trabalhava na parte inferior da página, com isso as notificações de atualização do usuário e do chat chegavam e ele não perceberia, ficando desatualizado até o momento que voltasse para o topo da

página. Essa mudança foi realizada pensando em melhorias na Interação Humano Computador do SCAD.

Outra melhoria implementada, foi a responsividade nas páginas do SCAD. Adaptando assim o sistema para uso em qualquer tipo de dispositivo. Essa implementação foi realizada da forma que proporciona o melhor desempenho na execução das páginas, ou seja, utilizando tecnologias que adapta as páginas ao tamanho da tela, em vez de ocultar conteúdos de acordo com o aparelho computacional. Dessa maneira eram realizados os desenvolvimentos de sistemas responsivos até pouco tempo atrás, quando eram utilizadas tecnologias não apropriadas.

Magno (2013) explica que estamos vivendo numa época em que a *web* disponibiliza variados tipos de conteúdo. Assim, temos que encontrar a melhor maneira de carregar o conteúdo mais importante (texto, imagens importantes) de maneira ágil e em qualquer tipo de dispositivo.

3.1.9 Banco de Dados

O esquema completo do banco de dados do SCAD, pode ser encontrado nos Trabalhos de Conclusão de Curso de Marcelino Mendes da Silva Neto e José Denes Lima Araújo. Eles desenvolveram as versões iniciais dos sistemas que integram o SCAD.

Algumas alterações foram realizadas no banco de dados. Foram criadas novas tabelas para cadastro de notificações, cadastro de mensagens, armazenamento de clientes *online* no *chat* e cadastro de mensagens para usuários específicos.

As mensagens são cadastradas na entidade de notificações no banco de dados para serem buscadas e enviadas em tempo real para os clientes. Essa entidade tem relacionamento com todas as tabelas do banco de dados que armazenam informações que precisam ser notificados para os clientes quando são cadastrados, excluídos ou atualizados. Através do identificador de cada tabela é possível identificar de qual informação é a notificação e a qual cliente pertence.

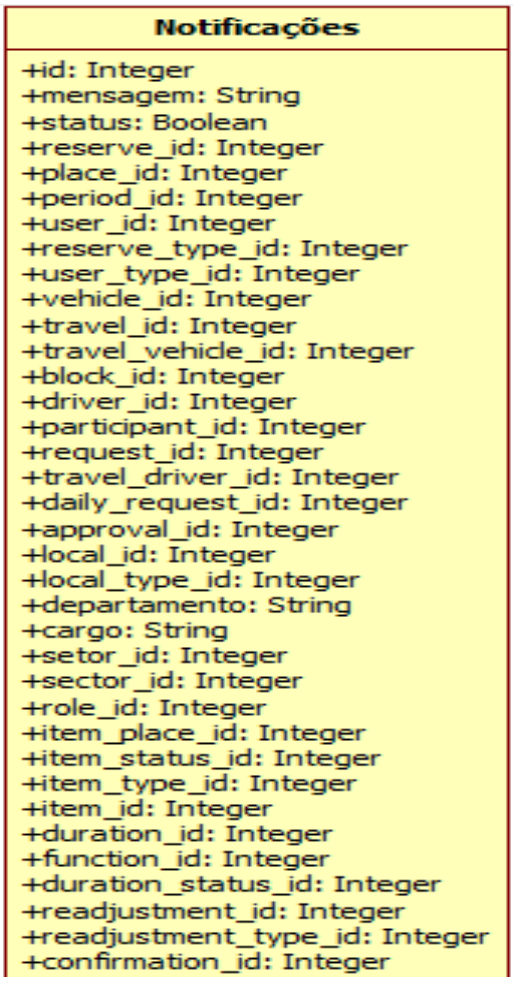


Figura 4 - Classe de cadastro de notificações

Fonte: Elaborada pelo autor (2015).

A entidade de clientes representa a tabela do banco de dados onde são armazenados os usuários que estão autenticados no SCAD. Através de buscas em seus dados é possível exibir, quem está disponível para receber mensagens em tempo real do *chat.* Ela possui relacionamento com a tabela de cadastro de usuários do SCAD. E armazena o identificador da conexão e o nome dos clientes.

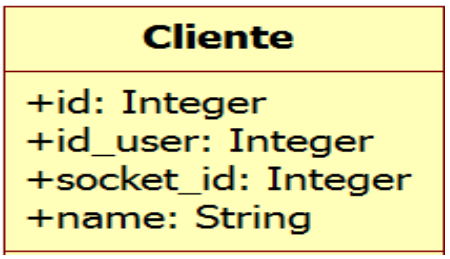


Figura 5 - Classe de clientes online

Fonte: Elaborada pelo autor (2015).

A entidade de mensagem do *chat* representa a tabela onde são cadastradas todas as mensagens enviadas. Essa tabela tem relacionamento com a tabela de usuários do SCAD, para que seja possível armazenar o identificador do usuário que enviou a mensagem e do que a recebeu. Assim, é possível manter um histórico de mensagens para cada usuário. Na tabela de mensagens é armazenado também um contador de mensagens não visualizadas e um campo para guardar notificações.

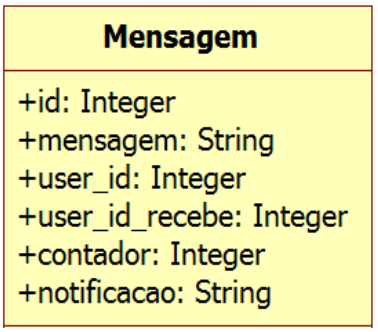


Figura 6 - Classe de cadastro de mensagens

Fonte: Elaborada pelo autor (2015).

A entidade de mensagens para usuários específicos, representa a tabela onde são armazenadas mensagens para os receptores que não estão autenticados no momento do envio da mensagem. Ela possui relacionamento com a tabela de usuários. Com isso é possível guardar o identificador do usuário receptor juntamente com a mensagem. Assim ele pode ser identificado, para que receba a mensagem na próxima vez que se autenticar no sistema.

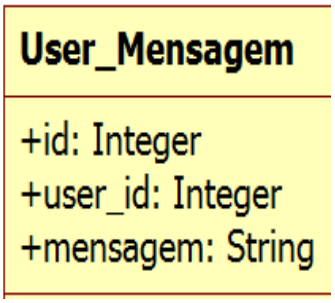


Figura 7 - Classe de cadastro de mensagens para usuários específicos

Fonte: Elaborada pelo autor (2015).

Essas tabelas foram criadas no banco de dados PostgreSQL. Esse banco de dados é utilizado pelo SCAD desde a sua implementação inicial. E foi mantido na realização dessa melhoria do sistema.

Nos estudos para desenvolvimento desse projeto, pode-se perceber as vantagens de bancos de dados não SQL (*Structured Query Language*), ou seja, banco de dados que não utilizam a linguagem SQL, para desenvolvimento de sistemas em tempo real. Pois alguns deles trabalham direto com objetos de dados, semelhantes a objetos trabalhos com a linguagem *JavaScript,* que é a linguagem do *Node.js.* Mas como explicado anteriormente, foi optado pela permanência do PostgreSQL, já que o SCAD funciona com esse banco de dados. E apresenta um desempenho considerado correto para o ambiente de execução do sistema.

3.1.10 *Interfaces* responsivas

Quando o SCAD é executado em dispositivos de telas pequenas, os dados, as tabelas que apresentam as informações e as barras de navegação, se tornam muito pequenas ou mudam de formação, causando dificuldade para seus usuários visualizarem e entenderem o conteúdo do sistema.

Esse funcionamento do SCAD se tornou insatisfatório para os padrões de tecnologias atuais. Muitos usuários usam variados tipos de aparelhos computacionais para executar suas funções de trabalho. Eles buscam utilizar dispositivos adaptados ao ambiente em que estejam.

Neste projeto foram utilizadas tecnologias modernas para implementar a responsividade no SCAD. Assim, os usuários poderão realizar atividades referentes a plataforma por meio de diferentes aparelhos computacionais (*smartphones, tablets*), bastando para isso que tenham acesso à *internet.*

A responsividade foi aplicada nas páginas do sistema utilizando o *framework bootstrap.* Com a utilização de seus métodos para aplicação de responsividade foi possível tornar as informações, tabelas de dados e barras de navegação, adaptáveis a diferentes tamanhos de telas de dispositivos. E assim, ampliar as formas de utilização do SCAD.

Figura 8 - Página do SCAD executada em telefone celular

Fonte: Elaborada pelo autor (2015).

Para funcionamento do SCAD com as novas funcionalidades é necessário a execução de dois servidores. Um servidor para executar a parte do sistema desenvolvida com o *Ruby On Rails* e o servidor do *Node.js.* Esses dois servidores são executados paralelamente, fazendo com que os métodos desenvolvidos utilizando a plataforma *Node.js* sejam executados integrados com as funções criadas com a linguagem de programação *Ruby*.

Com o desenvolvimento das duas novas funcionalidades no SCAD, ocorreu um grande aprimoramento na forma de trabalho na gestão das atividades da UFPI. O risco de erros na manipulação de informações diminui, pois os usuários estarão sempre atualizados sobre o que ocorre com os dados. Além disso, a possibilidade de comunicação usando o *chat* do SCAD, permite a resolução de problemas com a gestão das atividades sem a necessidade de parar o trabalho no sistema.

4 RESULTADOS E DISCUSSÕES

Esse capitulo explica o funcionamento dos serviços desenvolvidos neste projeto. Serão explicadas as principais melhorias no funcionamento do SCAD, ocorridas por causa das funcionalidades implementadas. E o funcionamento das novas funções e resultados alcançados.

4.1 Melhorias Alcançadas no Funcionamento do SCAD

Da forma como o Sistema de Consulta e Apoio a Decisão foi desenvolvido na sua primeira versão, quando o usuário está executando tarefas relacionadas com informações de uma página e outro usuário faz modificações nos dados dessa página, a modificação não é percebida pelo primeiro, sem que este atualize a página. Esse projeto resolveu os problemas causados por essa característica do SCAD, mantendo os usuários atualizados sobre a execução de tarefas relacionadas com seu trabalho.

O uso de informações desatualizadas pode levar a inconsistências de dados, causando erros graves no funcionamento das atividades gerenciadas pelo sistema. Esse trabalho resolve isso de uma maneira simples e produtiva, pois atualiza os usuários do SCAD sobre o que acontece no sistema, apenas enviando notificações em tempo real, sem atrapalhar o andamento da execução de suas tarefas no sistema.

4.2 Funcionamento dos Serviços Implementados

Para criar as mensagens que são enviadas nas notificações, foi preciso armazenar essas mensagens em uma tabela do banco de dados do SCAD. Elas são criadas e armazenadas no momento que os usuários executam ações no sistema. Para isso foi criado um cadastro de notificações nos métodos de cadastro, atualização e exclusão de todos os controladores do sistema, e também nos métodos de aprovação e desaprovação de reservas de espaços físicos, reservas de veículos e pedidos de diárias. Um servidor *Node.js* foi configurado para abrir uma conexão para cada cliente conectado no SCAD, com isso, dentro da conexão foi usado a linguagem *JavaScript* para verificar quando novos cadastros de notificações forem realizados. Quando é verificado uma nova inserção de notificação no banco, o servidor busca

essa notificação e a envia em tempo real para os clientes conectados, utilizando o módulo *socket.io* do *Node.js*.

Quando a notificação chega nas páginas do cliente, é verificado a qual usuário ela pertence, depois disso a notificação é exibida. Essas restrições foram criadas para que cada usuário do sistema receba apenas as mensagens de atualização das informações que pertencem a seu trabalho.

A seguir será apresentado o passo a passo do funcionamento de cada funcionalidade desenvolvida.

A Figura 9 mostra uma notificação da realização de uma reserva de espaço físico. A mensagem chegou para o chefe de setor de patrimônio (usuário fictício). Na imagem pode-se perceber a quantidade de notificações que chegou, sendo que no exemplo chegou uma. As notificações são exibidas na barra logo abaixo da barra principal.

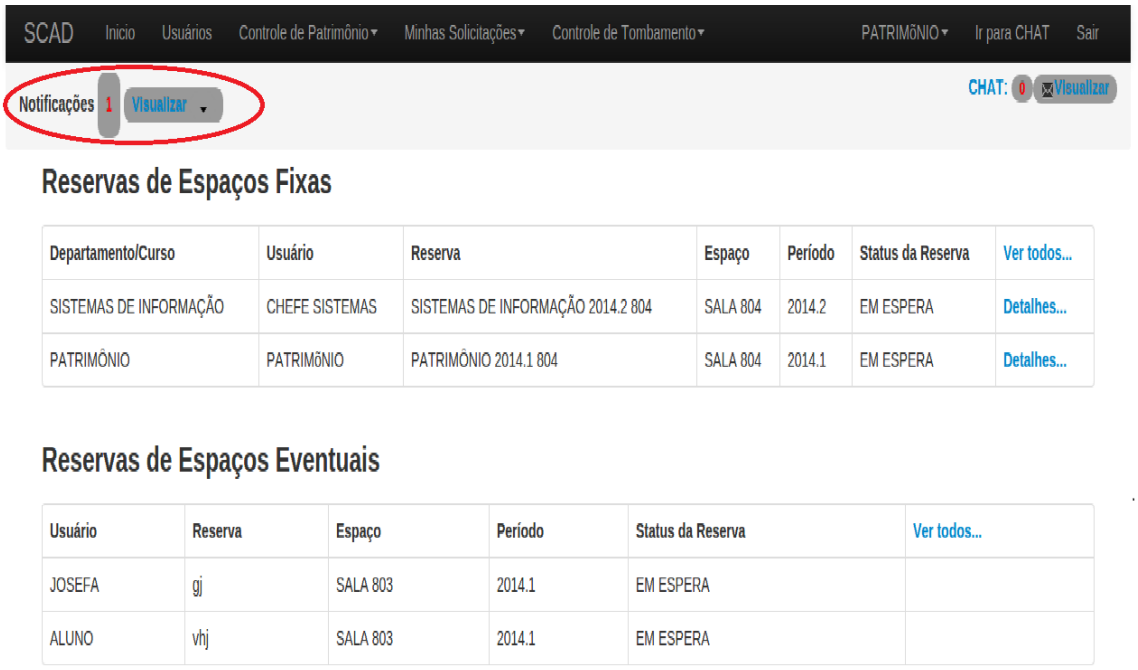


Figura 9 - Barra de notificações

Fonte: Elaborada pelo autor (2015).

A figura 10 mostra a mensagem sendo visualizada pelo chefe de setor de patrimônio. Ele tem a opção de marcar como visualizada, ou de deixa-la para ser visualizada depois, em outro momento do trabalho. Ele tem a opção também de ver todas as notificações que chegaram.

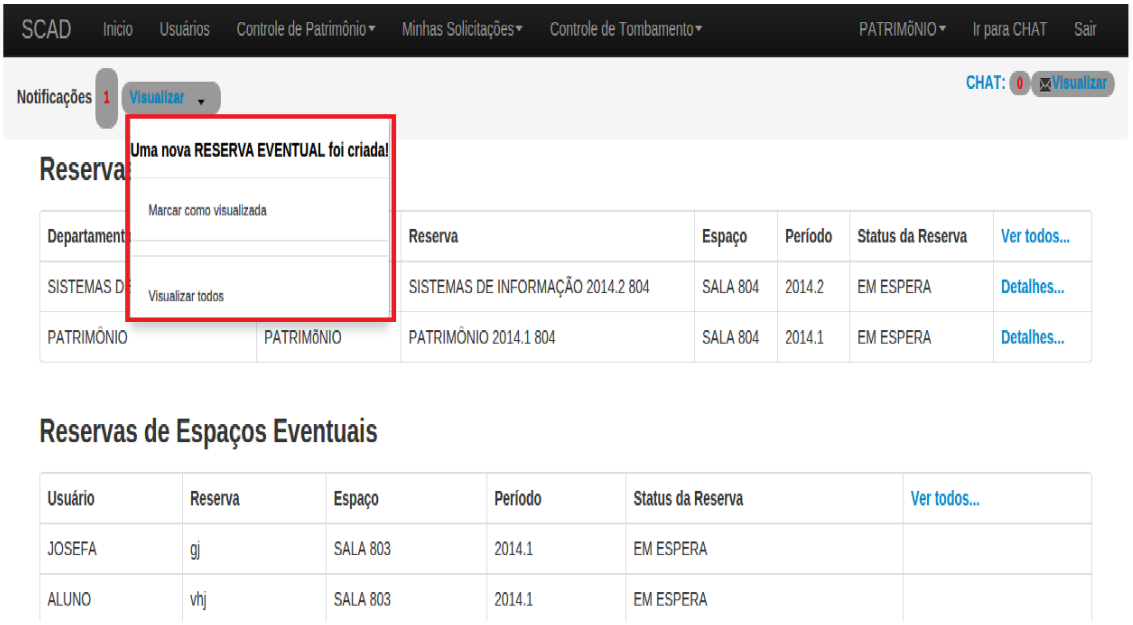


Figura 10 - Mensagem na barra de notificação

Fonte: Elaborada pelo autor (2015).

As figuras 11 e 12 mostram a notificação de aprovação da reserva eventual. Ela foi aprovada pelo chefe de setor de patrimônio. E a notificação foi enviada ao usuário que a solicitou.

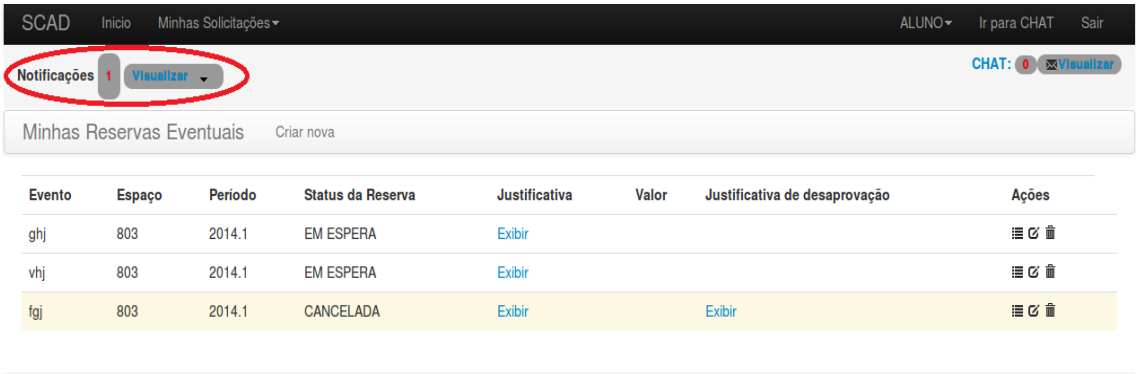


Figura 11 - Notificação de aprovação de reserva eventual

Fonte: Elaborada pelo autor (2015).

Figura 12 - Mensagem de aprovação de reserva eventual

Fonte: Elaborada pelo autor (2015).

A figura 13 exibe a página de listagem de todas as notificações. No exemplo da imagem, é mostrada a página referente ao chefe de setor de transporte. Como já foi explicado antes nesse trabalho, as notificações chegam para cada usuário de acordo com suas funções e autorizações.

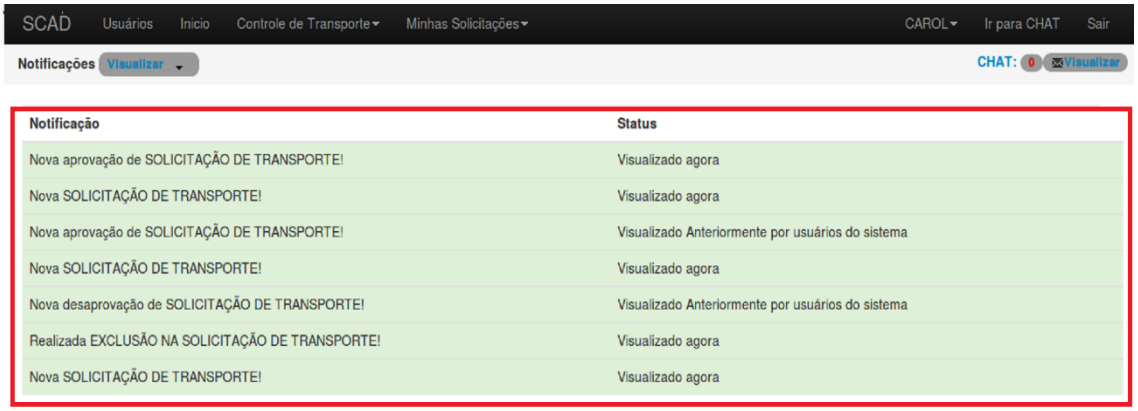


Figura 13 - Listagem de notificações

Fonte: Elaborada pelo autor (2015).

A seguir será explicado o funcionamento do *chat*. Serão explicadas imagens da página dos usuários online, da página de envio de mensagens e da forma como os usuários recebem mensagens em páginas de trabalho, com informações de gestão da UFPI.

A figura 14 é da página que apresenta a opção de seleção de usuários para enviar mensagem. Nessa página fica disponível também quais os usuários estão autenticados no SCAD no momento. Os usuários autenticados têm seus nomes destacados com a cor verde, para indicar que estão disponíveis e poderão receber as mensagens em tempo real.

Figura 14 - Página de usuários do chat

Fonte: Elaborada pelo autor (2015).

Quando o usuário seleciona outro na página da figura 14, ele é direcionado para a página mostrada na figura 15. Essa é a página de envio e recebimento de mensagens do *chat*. Nessa página é apresentada uma breve explicação sobre o propósito do *chat* e sua importância para os usuários do SCAD. Na figura 15 é apresentado o envio de uma mensagem por um aluno para o chefe de setor de patrimônio.

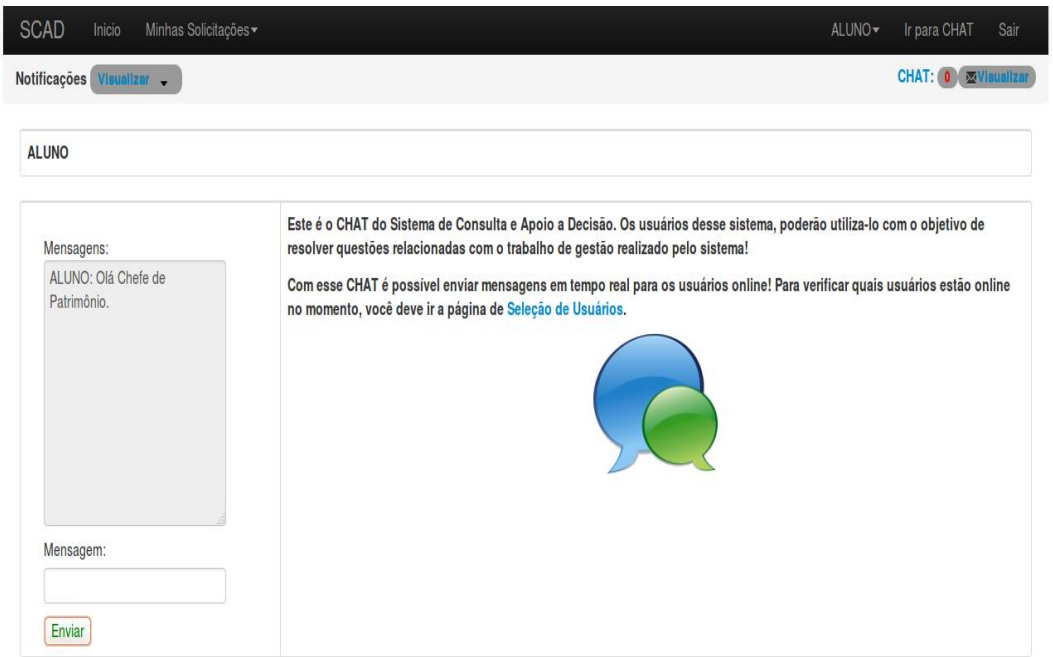


Figura 15 - Página de envio e recepção de mensagens

Fonte: Elaborada pelo autor (2015).

A figura 16 mostra o chefe de setor de patrimônio recebendo em tempo real a mensagem enviada pelo aluno e a respondendo.

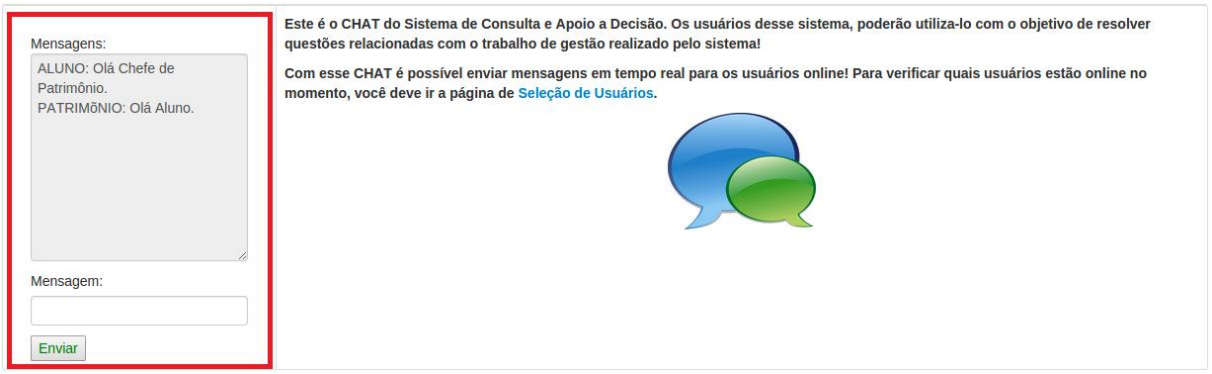


Figura 16 - Recepção de mensagem
Fonte: Elaborada pelo autor (2015).

As figuras 17, 18 e 19 mostram o chefe de setor de transporte recebendo mensagem do chefe de setor do curso de sistemas de informação. Mas dessa vez de forma diferente da apresentada anteriormente. Como já foi explicado nesse trabalho, foi verificado a importância de o usuário ter a opção de responder mensagens do *chat* sem a necessidade de sair da página em que está executando suas funções.

A figura 17 mostra a mensagem chegando na barra de notificações.

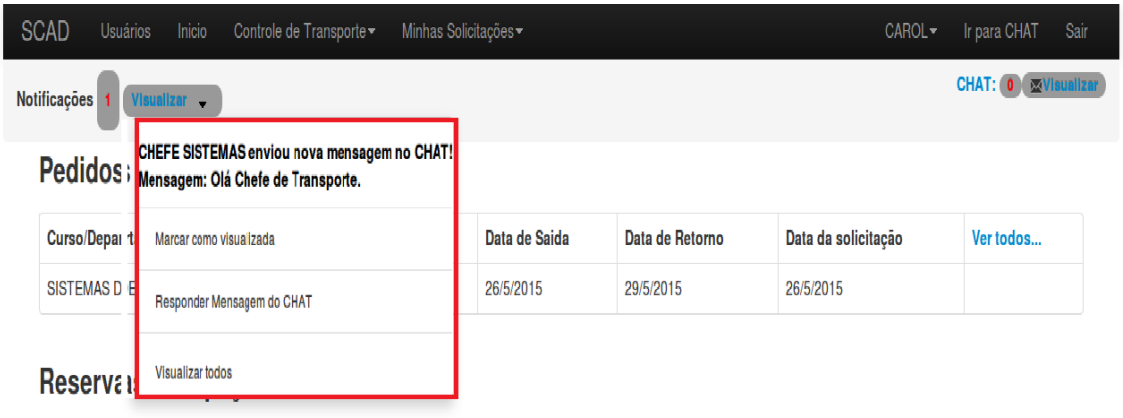


Figura 17 - Recepção de mensagem na barra de notificações
Fonte: Elaborada pelo autor (2015).

Na figura 18 é apresentada a opção de responder à mensagem sem sair da página e sem carrega-la.

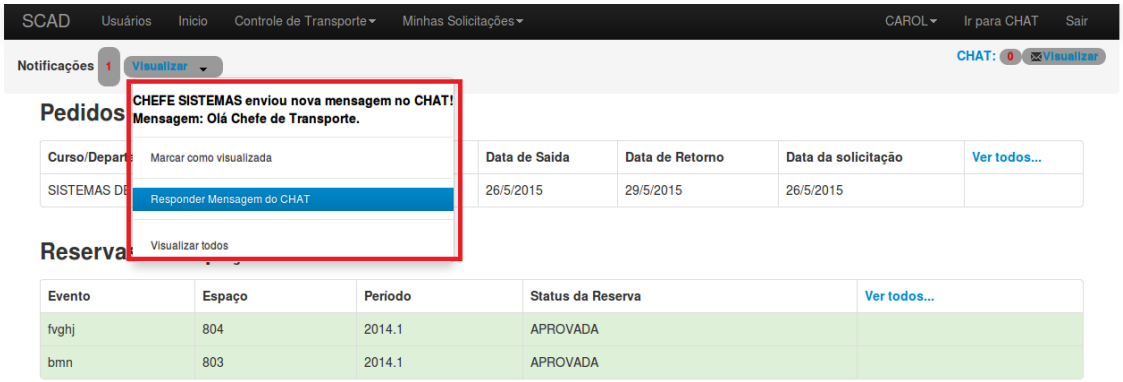


Figura 18 - Opção de responder mensagem

Fonte: Elaborada pelo autor (2015).

Depois de clicar na opção selecionada na figura 18, irá aparecer a opção de responder à mensagem, como é mostrado na figura 19. Essa opção fica disponível para o usuário na barra de notificações e sem a necessidade de carregar a página, assim não é perdido o trabalho que ele está realizando.

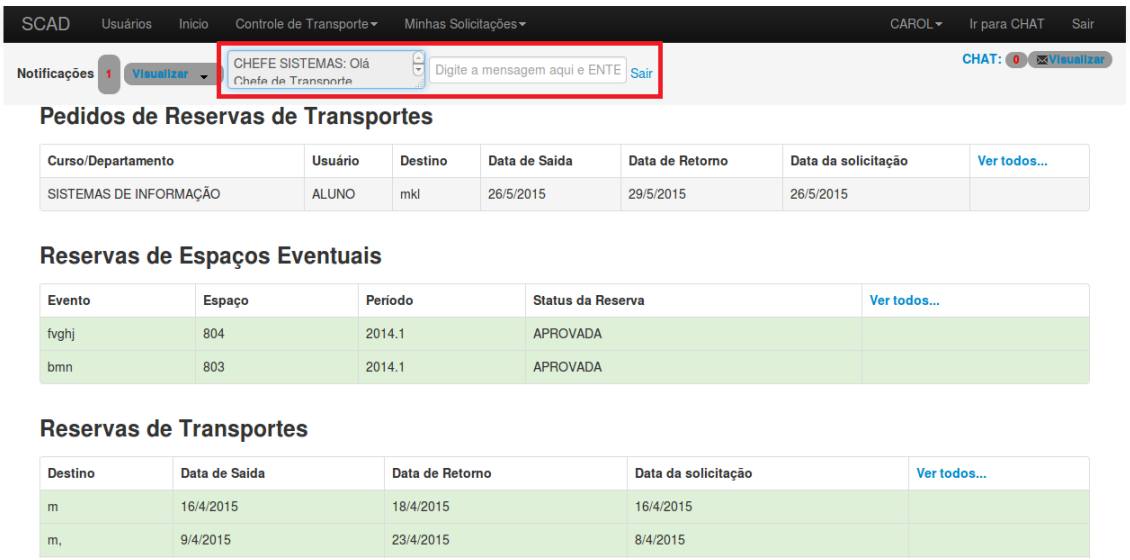


Figura 19 - Opção de responder mensagem do chat na barra de notificações

Fonte: Elaborada pelo autor (2015).

A figura 20 mostra o ícone de aviso de mensagens recebidas, enquanto o usuário não estava autenticado no SCAD. As mensagens são armazenadas e ele recebe uma notificação na próxima vez que se autenticar, avisando que ele tem

mensagens não visualizadas. Essa notificação aparece no final da barra de notificações, onde fica a opção chamada *CHAT*. Na figura 20 pode-se observar que o usuário tem uma mensagem pendente.

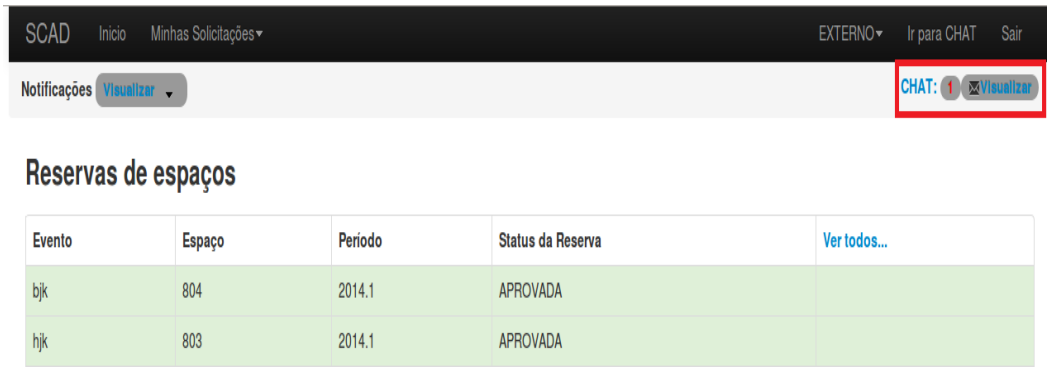


Figura 20 - Notificação de mensagens não visualizadas

Fonte: Elaborada pelo autor (2015)

Quando o usuário clica para visualizar a mensagem na opção apresentada na figura 20, ele é redirecionado para a página da figura 21, onde poderá visualizar as mensagens recebidas enquanto esteve ausente. Quando ele visualiza essa página, o contador de notificações do *chat* volta ao valor zero.

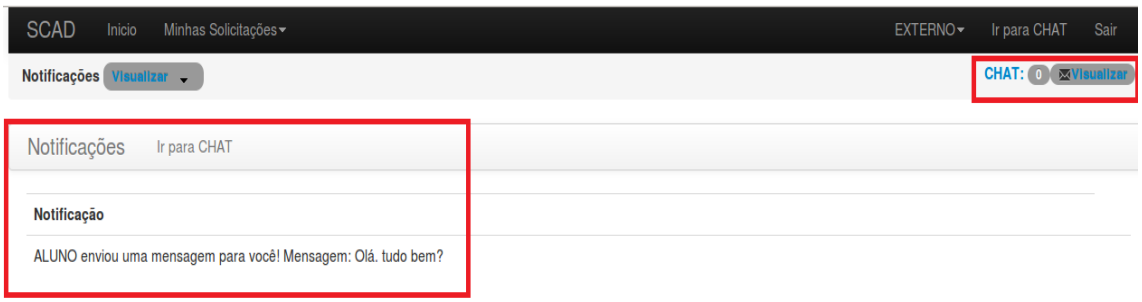


Figura 21 - página de visualização de mensagens

Fonte: Elaborada pelo autor (2015).

Para o usuário visualizar as mensagens de suas conversas, o *chat* disponibiliza a opção de seleção de conversa para ver o histórico de mensagens. Nesse histórico tem todas as mensagens enviadas e recebidas por ele e a opção de exclusão de mensagens e de toda a conversa. Para uma melhor compreensão a figura 22 exibe esse funcionamento.

Figura 22 - Histórico de mensagens do chat

Fonte: Elaborada pelo autor (2015).

Podemos verificar nas imagens acima que demonstram o *chat,* que este proporciona um grande dinamismo e opções diferentes e necessárias na forma como pode ser utilizado.

O *chat* funciona em tempo real, ou seja, as tecnologias utilizadas em seu desenvolvimento permitem a criação de métodos de envio e visualização de mensagens em tempo real, sendo que esse procedimento é realizado de forma assíncrona. Assim não ocorre problemas com congestionamentos no canal de transmissão quando a um aumento significativo no número de usuários utilizando o *chat.*

4.3 Resultados Alcançados

Foram acrescentadas algumas melhorias no Sistema de Consulta e Apoio a Decisão, após a implementação das novas funcionalidades, podemos citar como exemplo a diminuição da quantidade de requisições feitas ao servidor, pois antes os usuários precisavam atualizar as páginas dentro de períodos pequenos de tempo para procurar atualizações nas informações do sistema. O usuário tinha que fazer essa atividade com frequência para não utilizar informações desatualizadas, e com isso, muitas vezes eram realizadas atualizações nas páginas para que o servidor fosse requisitado mesmo sem ter nenhuma atualização nas informações, pois o usuário não teria como saber disso. Com a nova implementação, as páginas são recarregadas para busca de atualizações no servidor apenas quando os clientes são avisados por mensagens de notificações que informam as modificações ocorrentes nos dados do sistema.

Os principais objetivos do serviço de notificações em tempo real é poder evitar possíveis erros com o uso de informações desatualizadas, diminuir a sobrecarga desnecessária no servidor e manter os usuários atualizados sobre os trabalhos realizados no SCAD, sem que para isso ocorra uma quebra no ritmo de trabalho. Esse último foi o principal motivo pelo qual o serviço de notificações foi desenvolvido, em vez de um serviço que exibisse as informações em tempo real nas páginas, pois isso, como explicado anteriormente nesse trabalho, causaria mudanças na página de trabalho do usuário sempre que novas informações fossem cadastradas, atualizadas e excluídas. Em um dia de trabalho com grande movimentação de dados (inserção, atualização e exclusão) no sistema, muitas informações seriam apresentadas simultaneamente e de forma contínua nas páginas, podendo assim interromper a execução das tarefas dos usuários.

A eficiência e eficácia do serviço de mensagens de notificação podem ser observados no teste realizado com 40 (quarenta) usuários, através da aplicação de questionário avaliativo. Os resultados obtidos foram os seguintes:

Quadro 1 – Resultado do teste do funcionamento do serviço de notificações

QUANTIDADE DE USUÁRIOS	**RESULTADOS**
30 (trinta)	Responderam que as mensagens de notificações em tempo real são capazes de atualizar os usuários do sistema, de forma rápida e intuitiva. E que o tempo de resposta a partir da realização da tarefa (que deve ser notificada) pelo usuário, é considerado correto.
5 (cinco)	Responderam que seria melhor a implementação de um serviço que apresentasse as informações atualizadas em tempo real. Mas que o serviço de notificação é objetivo e eficiente.
5 (cinco)	Responderam que achariam melhor, que as páginas do sistema se atualizassem

	automaticamente quando as notificações fossem enviadas ao cliente. Mas que o serviço de notificação é eficiente.

Tendo como base os dados coletados no questionário avaliativo, podemos verificar o percentual da avaliação no seguinte gráfico:

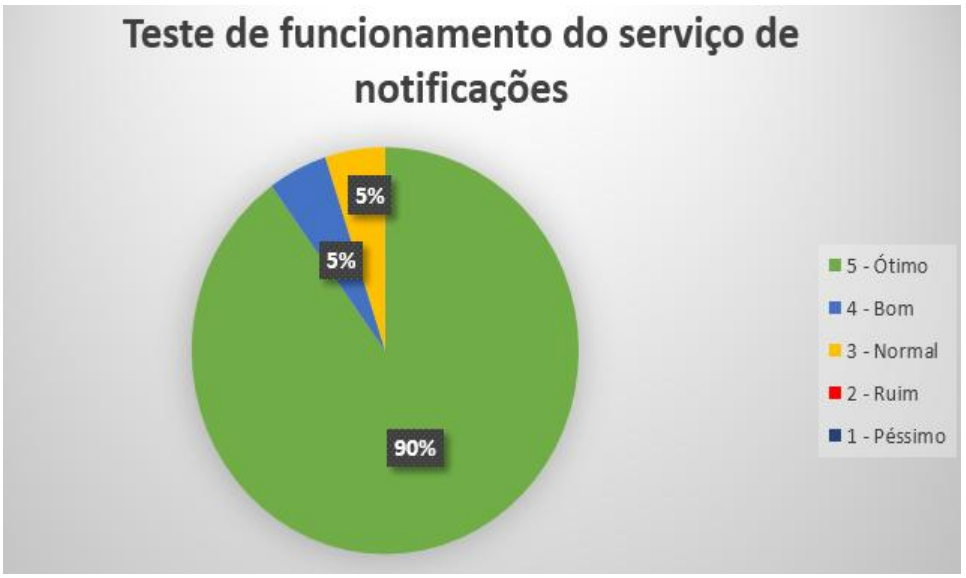


Figura 23 - Gráfico de percentual da avaliação do serviço de notificações pelos usuários

Fonte: Elaborada pelo autor (2015).

Podemos observar a eficiência do serviço de *chat*, no teste realizado com 40 (quarenta) usuários, mediante aplicação de questionário avaliativo, que obteve os seguintes resultados em relação a sua usabilidade:

Quadro 2 – Resultado do teste de usabilidade do *chat*

QUANTIDADE DE USUÁRIOS	**RESULTADOS**
21 (vinte e um)	Responderam não ter encontrado quase nenhuma dificuldade na utilização do *chat.*
10 (dez)	Responderam que não encontraram nenhuma dificuldade na utilização do *chat*.

8 (oito)	Responderam ter encontrado dificuldades na primeira utilização.
31 (trinta e um)	Responderam que a *interface* do *chat* é fácil de ser lembrada para um uso posterior.
9 (nove)	Responderam que encontraram pouca dificuldade de lembrar em um próximo uso.
18 (dezoito)	Responderam que a *interface* é totalmente intuitiva.
17 (dezessete)	Responderam que encontraram pouca dificuldade para entender a *interface*.
5 (cinco)	Responderam que tiveram algumas dificuldades no entendimento da *interface*.
32 (trinta e dois)	Responderam que as funções que o *chat* proporciona são simples e fáceis de serem executadas
8 (oito)	Responderam que encontraram algumas dificuldades na execução dessas funções.
25 (vinte e cinco)	Acharam o *chat* simples e objetivo para a troca de mensagens
15 (quinze)	Responderam que alguns ícones da *interface* torna o envio e recebimento das mensagens menos simples e objetivo.
38 (trinta e oito)	Sentiram-se seguros na execução das funções do *chat*, ou seja, não encontraram riscos (como por exemplo, botões de exclusão próximo a botões de enviar mensagens) no momento do envio e recebimento das mensagens.
40 (quarenta)	Responderam que a velocidade na troca de mensagens proporcionada pelo serviço de *chat* é ótima

Tendo como base a amostra dos dados coletados, os seguintes gráficos foram gerados em relação ao teste de usabilidade do serviço de *chat*:

A Figura 24 apresenta o gráfico com o percentual da avaliação realizada no teste de facilidade de utilização do *chat*.

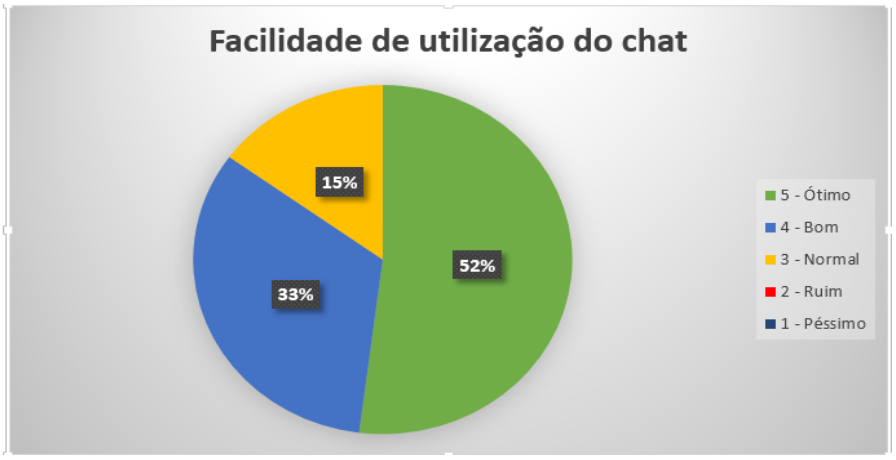


Figura 24 - Gráfico de percentual da avaliação de facilidade de uso do chat

Fonte: Elaborada pelo autor (2015).

A Figura 25 apresenta o gráfico com o percentual da avaliação realizada no teste de facilidade de aprendizagem das funções do *chat.*

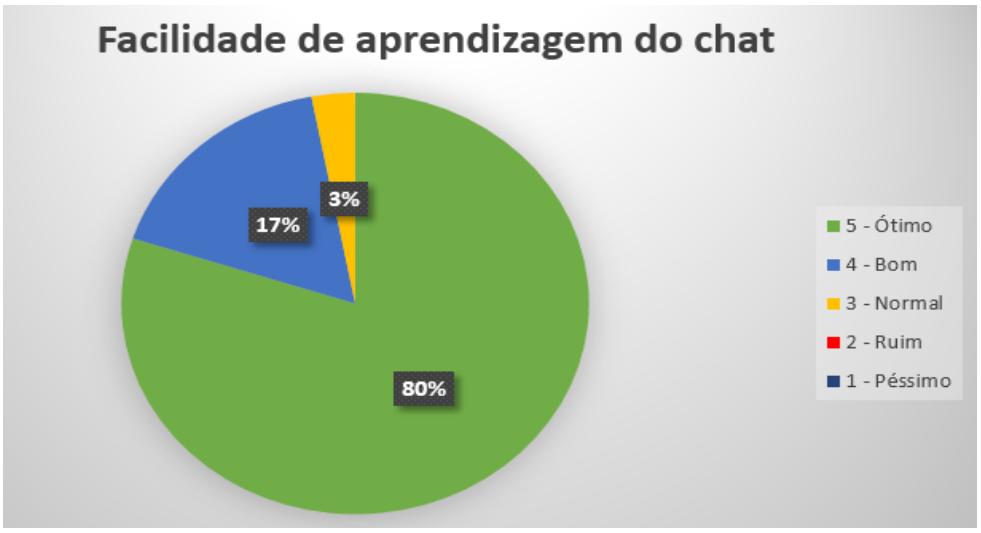


Figura 25 - Gráfico de percentual da avaliação de facilidade de aprendizado das funções do chat

Fonte: Elaborada pelo autor (2015).

A Figura 26 mostra o gráfico com o percentual da avaliação realizada no teste do nível de intuitividade da interface do *chat.*

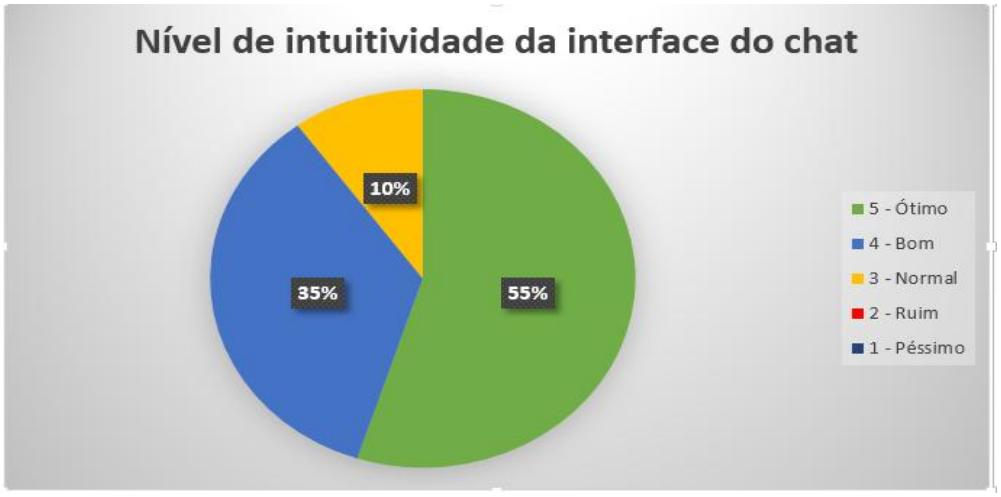


Figura 26 - Gráfico de percentual da avaliação do nível de intuitividade do chat

Fonte: Elaborada pelo autor (2015).

A Figura 27 apresenta o gráfico com o percentual da avaliação no teste de simplicidade de execução das funções do *chat.*

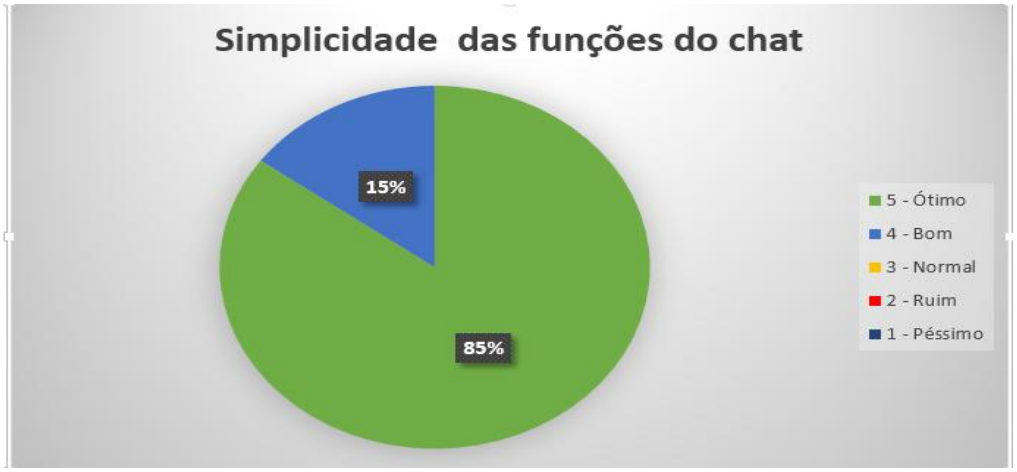


Figura 27 - Gráfico de percentual da avaliação de simplicidade das funções do chat

Fonte: Elaborada pelo autor (2015).

A Figura 28 apresenta o gráfico com o percentual da avaliação no teste de objetividade proporcionada pelo *chat,* na troca de mensagens.

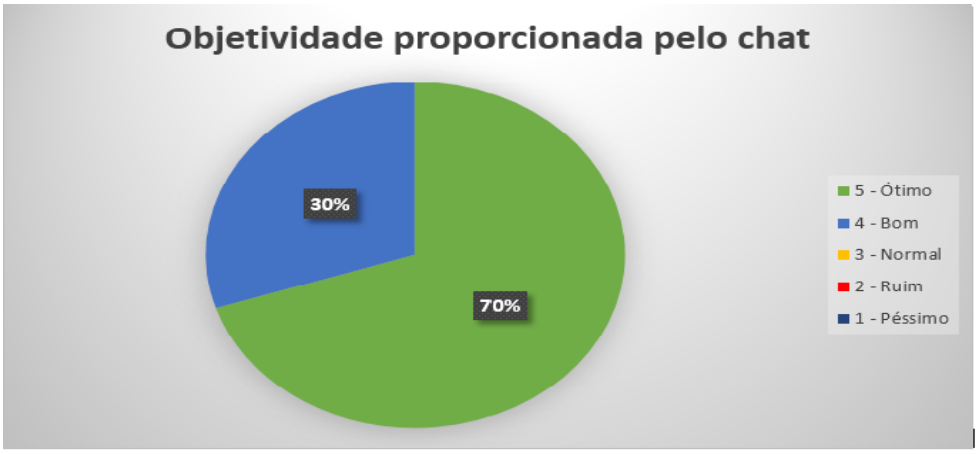


Figura 28 - Gráfico de percentual da avaliação da objetividade proporcionada pelo chat

Fonte: Elaborada pelo autor (2015).

A Figura 29 mostra o gráfico de percentual da avaliação no teste de segurança proporcionada pela interface do *chat.*

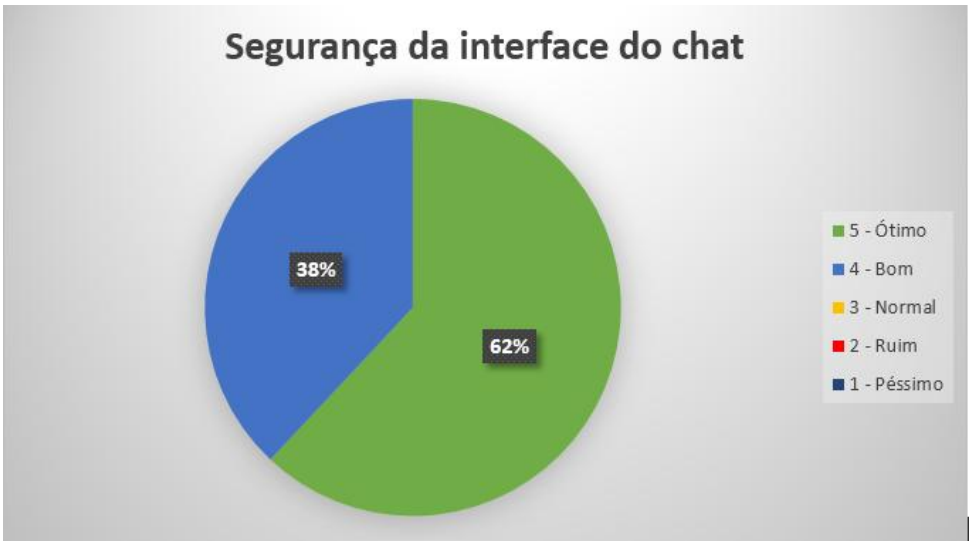


Figura 29 - Gráfico de percentual da avaliação de segurança proporcionada pela *interface* do chat

Fonte: Elaborada pelo autor (2015).

A Figura 30 mostra o gráfico de percentual da avaliação no teste de velocidade na troca de mensagens por meio do *chat.*

Figura 30 - Gráfico de percentual da avaliação da velocidade de troca de mensagens por meio do chat

Fonte: Elaborada pelo autor (2015).

O teste com o serviço de notificações apresentou resultados positivos. Grande maioria dos usuários aprovaram as melhorias que esse serviço proporcionaria para o funcionamento do SCAD. Mas uma pequena minoria opinou que achariam mais importante a implementação dessa funcionalidade com algumas mudanças em relação a que foi realizada.

As características (atualização automática das páginas e informações sendo mostradas em tempo real nas páginas) que os usuários do teste destacaram, para que na opinião deles, o serviço de notificações se tornasse mais eficiente, foram analisadas no momento de planejamento de como funcionaria o serviço. E foi verificado que essa forma de funcionamento causaria quebra na sequência de execuções de tarefas dos usuários do SCAD. Por exemplo, quando um usuário estivesse cadastrando informações no sistema e no momento de inserção de informações a página se atualizasse, as informações inseridas nos campos de texto seriam todas perdidas. Além de causar problemas na visualização das informações, pois se o sistema recebesse muitos cadastros, atualizações e exclusões de informações, as páginas se modificariam (informações aparecendo e se modificando) constantemente. Por essa razão, foi decidido a implementação de um serviço de envio de mensagens de notificações em tempo real, que mantêm os usuários sempre atualizados sobre as informações que pertencem ao trabalho deles.

O teste com o serviço de *chat* apresentou resultados satisfatórios. A maioria dos usuários encontraram pouca ou nenhuma dificuldade na execução das funções

de troca de mensagens. As principais características do *chat,* que os usuários do teste propuseram mudanças, foram na forma como os usuários são selecionados para o envio de mensagens e a pouca intuitividade dos ícones de navegação entre as funções do serviço. Levando em consideração as opiniões dos usuários, foi verificado que não seria possível alterações nas funcionalidades para que se adaptassem as características evidenciadas no teste, os principais motivos, são que essas funções foram desenvolvidas para se integrarem com o cadastro de usuários já existente no SCAD (assim, não é possível fazer modificações na seleção dos usuários, sem alterar o funcionamento padrão do sistema) e a forma como ocorre a navegação entre as funções do *chat* é a utilizada em todo o sistema, por isso foi optado pela manutenção do padrão.

Com tudo isso apresentado, podemos perceber as mudanças que irão acontecer no funcionamento do SCAD quando as novas funcionalidades forem colocadas em produção. O serviço de notificações e o *chat* proporcionará uma maior dinâmica e um melhoramento na forma como os usuários executaram suas tarefas, com menor risco de erros na execução da manipulação das informações e mais facilidades na comunicação entre usuários, que com o *chat* do SCAD utilizarão o próprio sistema para isso.

5 CONSIDERAÇÕES FINAIS

Esse trabalho foi desenvolvido com o objetivo de refinar o Sistema de Consulta e Apoio a Decisão-SCAD para Universidade Federal do Piauí – Campus Senador Helvídio Nunes de Barros. O SCAD foi desenvolvido para plataforma *web,* o que possibilitou o desenvolvimento de rotinas de *realtime,* utilizando ferramentas que trabalham de forma assíncrona, tornando essas rotinas eficientes para o trabalho com muitos usuários.

Esse projeto obteve resultados como desenvolvimento do serviço de envio de mensagens de notificações em tempo real e implementação de uma função de bate-papo *online* e em tempo real. Esses resultados estão em conformidade com os objetivos que foram propostos, levando em consideração a manutenção do princípio de funcionamento do SCAD.

As tecnologias utilizadas proporcionaram facilidades com a possibilidade de serem executadas de forma integradas. Sendo possível assim a criação de métodos com linguagens de programação e plataformas variadas, dentro do mesmo projeto. Com isso pode-se aproveitar os recursos disponibilizados por várias ferramentas na implementação do sistema.

Até o presente momento o SCAD está em fase de testes e treinamentos com os usuários. Então, os testes realizados para verificar a eficiência do sistema, foram realizados em ambiente de simulação, ou seja, em ambiente de desenvolvimento, pois o sistema ainda não está executando em modo de produção.

O sistema com essas novas funcionalidades permite maior confiabilidade na gestão das informações sobre as reservas de espaços físicos, alocações de veículos e pedidos de diárias (dado que estarão sempre atualizados sobre os dados que estão gerindo), e maior rapidez de comunicação e resolução de problemas via sistema.

Referências

ARAÚJO, José. **Sistema *Web* para Controle de Alocação de Espaços Físicos da Universidade Federal do Piauí – Campus Senador Helvídio Nunes de Barros.** TRABALHO DE CONCLUSÃO DE CURSO. Picos: Universidade Federal do Piauí, 2014.

BALDUINO, Plínio. **Dominando *JavaScript* com jQuery.** São Paulo: Casa do Código, 2013.

CAELUM. Disponível em www.caelum.com.br acessado em 14 de maio de 2015.

CANTELON, Mike; HARTER, Marc; HOLOWAYCHUK, T.J; RAJLICH, Nathan. ***Node.js in action.*** New York: Manning Publications Co, 2014.

COCHRAN, David. **Twitter Bootstrap Web Development How-To:** A hands-on introduction to building websites with Twitter Bootstrap's powerful front-end development framework. Birmingham: Packt Publishing, 2012.

COLEMAN, Tom; GREIF, Sacha. ***Discover METEOR Building Real-Time JavaScript Web Apps.*** Made in Osaka & Melbourne. © 2014

DOCUMENTAÇÃO DO POSTGRESQL 8.0.0. **The PostgreSQL Global Development Group.** California: Copyright © 1996-2005 The PostgreSQL Global Development Group.

FLANAGAN, David. ***JavaScript:*** *The Definitive Guide,* Sexta Edição, Reilly Media, Inc., 2011.

FILHO, Wilson de Pádua Paula, **Engenharia de Software: fundamentos, métodos e padrões.** Rio de Janeiro: LTC - LIVROS TECNICOS E CIENTIFICOS, 2000.

FUENTES, Vinícius. **Ruby On Rails:** Coloque sua aplicação *web* nos trilhos. São Paulo: Casa do Código, 2012.

LENGSTORF, Jason; LEGGETTER, Phil. **Realtime *Web* Apps:** *With HTML WebSocket, PHP, and jQuery. New York: Apress Berkely,* CA, USA, ©2013.

MAGNO, Alexandre. ***Mobile First Bootstrap: Develop advanced websites optimized for mobile devices using the Mobile First feature of Bootstrap***. Reino Unido: Packt Publishing Ltd, 2013.

MINETTO, Elton Luís. **Frameworks para Desenvolvimento em PHP.** São Paulo: Novatec, 2007.

NIEDERAUER, Juliano. ***Web* interativa com Ajax e PHP.** São Paulo: Novatec, 2013.

PAULI, Josh. **Introdução ao *web Hacking.*** São Paulo: Novatec, 2013.

PEREIRA, Caio Ribeiro. ***Node.js:*** Aplicação *web realtime* com *Node.js.* São Paulo: Casa do Código, 2013.

PRESSMAN, Roger. **Engenharia de Software: Uma abordagem Profissional.** 7ª ed. New York, NY, EUA: The McGraw-Hill Companies, Inc., 2011.

RAI, Rohit. ***Socket.IO Real-time Web Application Development: Build modern real-time web applications powered by Socket.IO.*** Birmingham: Packt Publishing, 2013.

SILVA, Marcelino. **SISTEMA WEB PARA CONTROLE DOS PROCESSOS DE ALOCAÇÃO DE VEÍCULOS DO SETOR DE TRANSPORTE DA UNIVERSIDADE FEDERAL DO PIAUÍ – CAMPUS SENADOR HELVÍDIO NUNES DE BARROS.** TRABALHO DE CONCLUSÃO DE CURSO. Picos: Universidade Federal do Piauí, 2014.

SILVA, Maurício Samy. ***JavaScript:*** Guia do Programador. São Paulo: Novatec, 2010.

SILVA, Maurício Samy. ***Html5*** **a Linguagem de Marcação que Revolucionou a *Web.*** São Paulo: Novatec, 2011.

SOMMERVILLE, Ian. **Engenharia de software**. 8ª ed. São Paulo: Pearson Addison-Wesley, 2007.

SOUZA, Lucas. ***Ruby:*** Aprenda a programar na linguagem mais divertida. São Paulo: Casa do Código, 2012.

TEIXEIRA, Pedro. ***Hands-on Node.js,*** 2013. 1p.

TIOBE.COM, **Índice TIOBE, para junho de 2015**. Disponível em <http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html>. Acessado em 13 de junho de 2015.

TERMO DE AUTORIZAÇÃO PARA PUBLICAÇÃO DIGITAL NA BIBLIOTECA "JOSÉ ALBANO DE MACEDO"

Identificação do Tipo de Documento

( ) Tese

( ) Dissertação

(X) Monografia

( ) Artigo

Eu, Abimael Santiago Velozo, autorizo com base na Lei Federal nº 9.610 de 19 de Fevereiro de 1998 e na Lei nº 10.973 de 02 de dezembro de 2004, a biblioteca da Universidade Federal do Piaui a divulgar, gratuitamente, sem ressarcimento de direitos autorais, o texto integral da publicação Implementação de Melhorias de Comunicação e IHC no Sistema de Consulta e Apoio a Decisão – SCAD, Usando Mecanismos de Realtime, de minha autoria, em formato PDF, para fins de leitura e/ou impressão, pela internet a título de divulgação da produção científica gerada pela Universidade.

Picos-PI 27 de Outubro de 2015.

Abimael Santiago Velozo
Assinatura

Abimael Santiago Velozo
Assinatura