MODELO DE SISTEMA DE COMUNICAÇÃO ENTRE NAVEGADOR E SISTEMA OPERACIONAL.

WILLIAM DOS SANTOS RUTZ[1]; KRISHNA FERREIRA XAVIER[2]; CALEBE DIAS BORGES[3]; JULIA GARCIA ALVES[4]; ROSA BEATRIZ SIMÕES SICA[5]; Profª. DrªADRIANE PIRES RODRIGUES RAMIRES [6]

[1]Instituto Federal Sul-rio-grandense – william.rutz@hotmail.com
[2]WeTech/IFSul – tsixav@gmail.com
[3]Instituto Federal Sul-rio-grandense– calebe97@gmail.com
[4]Instituto Federal Sul-rio-grandense – juliaga.004@gmail.com
[5]Instituto Federal Sul-rio-grandense – rbsica@gmail.com
[6]Instituto Federal Sul-rio-grandense – apires@pelotas.ifsul.edu.br

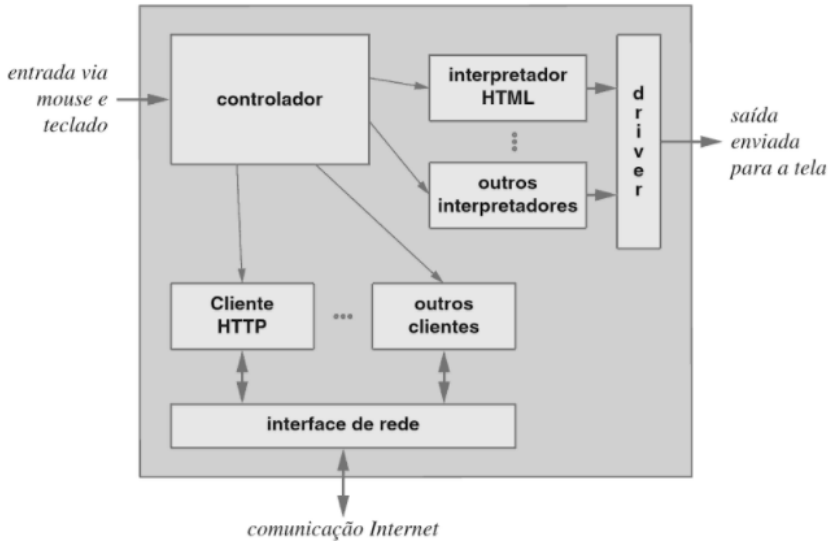
1. INTRODUÇÃO

Os Navegadores *Web* (NW) podem apresentar restrições de funcionalidades comparado às aplicações de *Software Desktop*, como, por exemplo, o controle do movimento do *mouse*. O modelo de comunicação utilizado pelos NW apresenta essas restrições em virtude da questão de segurança, a qual impede que *Web Sites* (WS) controlem ou executem comandos do SO.

Um NW, ou simplesmente Navegador (em inglês: *Web Browser*), é um programa que permite seus usuários interagirem com documentos *HTML* (*HyperText Markup Language*) hospedados em um Servidor da Rede. Tem alto destaque na era da *web* 2.0, uma vez que a maior parte das informações que necessita-se estão disponíveis *online*.

De acordo com COMER (2016), a arquitetura do NW é complexa, devido ao fornecimento de uma grande quantidade de serviços gerais, suportando uma interface gráfica complexa com capacidade de atender à demanda de serviços. Sabe-se que além de interpretar o *HTML*, o navegador deve fornecer suporte para outros protocolos e serviços. Na Figura 1 são ilustrados os componentes que compõem o navegador.

Figura 1: Arquitetura do Navegador Web



Fonte: COMER, p. 54

Para ocorrer a interação entre o navegador e o servidor, o NW utiliza o serviço de fluxo (que é uma sequência de *bytes* transmitidos de um programa de aplicação para outro) para se comunicar com o servidor, como exemplo o *download* de um arquivo, onde o NW envia uma requisição e o servidor responde com a página solicitada. Em seguida, a rede recebe os dados das duas aplicações e entrega-os para os respectivos destinos, segundo COMER (2016). Nesse contexto, este trabalho apresenta uma proposta de modelo de sistema capaz de realizar a comunicação entre uma página *Web* e o SO, para possibilitar um aumento nas funcionalidade de uma aplicação *Web.*

2. METODOLOGIA

O projeto em questão, tem como objetivo identificar uma solução para o problema abordado. O modelo proposto foca no uso dos movimentos do *mouse* como validação, que pode tornar mais acessível a utilização de computadores e da *internet* para usuários com algum tipo de deficiência físico-motora. Possibilitando através da correção e antecipação do movimento do *mouse,* e da adição de gravidade em elementos *HTML* como, por exemplo, um botão pequeno que atrai o ponteiro do *mouse* para o centro, facilitando assim o uso de uma interface *Web*.

Para definir uma alternativa que viabilize a realização desta proposta, foi elaborada uma pesquisa para obter quais as informações e as funcionalidades os NW tem acesso em relação ao SO, buscando assim dados úteis para a base de viabilidade do projeto.

Para indicar quais informações o NW tem acesso, foi utilizado um *site* que retorna os principais dados encontrados como, por exemplo, a localização do usuário que é identificada através da API do *Google Maps* (*Google Developers*, 2017), e o IP (*Internet Protocol*) do usuário. Conforme a Figura 2 (*Software*) pode-se observar informações identificadas sobre o *software* em si, como o nome SO do usuário e a versão do navegador.

Figura 2 - Informações do navegador sobre *Software* e *Hardware*.

Fonte: webkay.robinlinus.com

Por fim, foram identificadas as informações do *hardware* do computador do usuário, como a CPU, GPU e o nível de bateria. Nota-se que o navegador não tem acesso a dados sobre os dispositivos de entrada do usuário como o *mouse*.

Uma das tarefas mais difíceis durante a etapa de levantamento de requisitos, é compreender as necessidades do modelo. Para isso, foram aplicados conceitos definidos por PRESSMAN (2011), “O amplo espectro de tarefas e

técnicas que levam a um entendimento dos requisitos é denominado engenharia de requisitos" (PRESSMAN, 2011).

Através destes requisitos, foi definido que o modelo deve ser capaz de realizar um diálogo entre uma PW e um segundo sistema (que faz parte do modelo), através de requisições *HTTP* (*Hypertext Transfer Protocol*), que por sua vez deve mandar comandos para o SO. Diante destas instruções, foi preparado um estudo de viabilidade de tecnologias e bibliotecas aptas para atender esses requisitos, que serão apresentadas no tópico 3 deste trabalho.

3. RESULTADOS E DISCUSSÃO

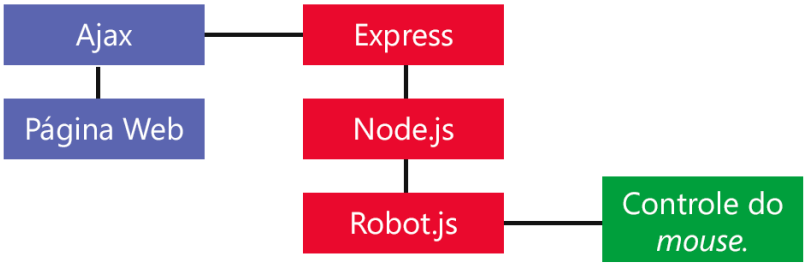
O modelo do sistema proposto consiste em validar a comunicação entre uma PW e o SO para ampliar as funcionalidade de uma Aplicação *Web*, para ajustar ou melhorar o controle do *mouse*, utilizados para tecnologias de acessibilidade, como o VisiUMouse e IOM (Interface Óculos Mouse).

A principal tecnologia utilizada é o *Node.js*, que é um interpretador de código *JavaScript*, que funciona do lado do servidor, TILKOV e VINOSKI (2010). Apresenta vários módulos, os quais são subprogramas que executam tarefas definidas. O módulo utilizado foi o *RobotJS*, com ele tornou-se possível a simulação das principais ações do usuário, como a digitação de teclas, cliques e movimentação do *mouse*. Em conjunto, foi utilizado o *Express*, que é um *Framework* para aplicativos da *Web* do *Node.js*. As ferramentas citadas são executadas no computador do usuário como um *Software* habitual.

Parte do modelo precisa ser executado dentro do NW. Para isso, foi utilizado a *Ajax* que é o uso metodológico de tecnologias como *Javascript* e *XML (eXtensible Markup Language)*, providas por NW, para tornar PW mais interativas com o usuário, utilizando-se de solicitações assíncronas de informações.

A Figura 3 apresenta o diagrama do modelo proposto, em azul (lado esquerdo) são as partes que estão dentro do NW, que por sua vez se comunica com o *Express* por meio de requisições *HTTP* através do *Ajax*. Em vermelho é representado o *Express*: o qual recebe os dados e comandos da PW; o *Node.js* que é responsável pela lógica e controle de todos dados e comandos do modelo; o *RobotJS* permite simular as principais funcionalidade do *mouse*, como a movimentação. Em verde (lado direito) é ilustrado o controle do *mouse* em si.

Figura 3: Diagrama do modelo de sistema



Fonte: Autor

4. CONCLUSÕES

À partir da problemática abordada no artigo, que consiste na limitação da comunicação entre o SO e o NW, foram realizados estudos para elaboração de testes de tecnologias que pudessem sanar essa falha. Para isso foram utilizadas as tecnologias: *Node.js*, *Express*, *RobotJS* e *Ajax*. Com base no teste de viabilidade constatou-se que é possível criar aplicações híbridas, isto é, aplicações que podem apresentar funcionalidades *Web* e de *Softwares Desktop*.

Tendo em vista os resultados alcançados, o modelo de sistema apresentado pode servir como base para estudos, pesquisas ou outros trabalhos, que tenham interesse em desenvolver aplicações similares.

O modelo de sistema apresentado neste trabalho será utilizado como base para a desenvolvimento de um *Framework* de acessibilidade, denominado até o momento de *Gravel*, o qual tem como objetivo melhorar a interação do usuário de Tecnologia Assistiva, com foco no VisiUMouse e no IOM, com interfaces *Web*. Entre suas principais funcionalidades estarão a possibilidade de adicionar uma "força gravitacional" em elementos *HTML*, como botões e entrada de dados; fazer a correção da movimentação do cursor do *mouse*; permitir *feedback* visual para as ações do usuário como, por exemplo, efeito de som ao clicar e também tornar possível a parametrização dos dados do uso do *mouse*.

5. REFERÊNCIAS BIBLIOGRÁFICAS

COMER, Douglas E. **Redes de Computadores e Internet-6**. Bookman Editora, 2016.

GOOGLE DEVELOPERS. **The Google Maps Geolocation API, Google Maps Geolocation API, Google Developers**. Disponivel em: <https://developers.google.com/maps/documentation/geolocation/intro>. Acesso em 3 Out. 2017.

KE, Chang-zheng; HUANG, Hou-kuan. **Principle and application of Ajax technology [J]**. Railway Computer Application, v. 1, p. 011, 2007.

LINUS, R. **What every Browser knows about you**. Disponivel em: <http://webkay.robinlinus.com>. Acesso em 5 Out. 2017.

PRESSMAN, R.S. **Engenharia de Software**. Porto Alegre: AMGH Editora, 2011.

TILKOV, Stefan; VINOSKI, Steve. **Node.js:** Using JavaScript to build high-performance network programs. IEEE Internet Computing, v. 14, n. 6, p. 80-83, 2010.