Estudo de Caso: Utilizando Raspberry Pi para Acesso Remoto de Estações de Trabalho

Bruno Rodrigues de Almeida[1], Mirkos Ortiz Martins[1]

[1]Sistemas de Informação – Centro Universitário Franciscano
97010-032– Santa Maria – RS – Brazil

brunorodrigues@outlook.com, mirkos@gmail.com

Abstract. *Faced with traditional models of access to computing resources, a study is conducted to find other solutions with a more attractive cost / benefit for companies and educational institutions. In order to achieve the objectives, the FDD agile methodology was used in the development of the application, which will be responsible for configuring the remote desktop access environment and, thus, virtualising the computational resources. In order to achieve this, Raspberry Pi hardware was used instead of microcomputers, in order to achieve low energy consumption, lower heat generation and better utilization of physical space.*

Resumo. *Diante dos modelos tradicionais de acesso a recursos computacionais, é realizado um estudo para encontrar outras soluções com um custo/benefício mais atraente para empresas e instituições de ensino. Para alcançar os objetivos, foi utilizada a metodologia ágil FDD no desenvolvimento da aplicação, que será responsável por configurar o ambiente de acesso à área de trabalho remota e, assim, virtualizar os recursos computacionais. Para tanto, foi utilizado o hardware Raspberry Pi como equipamento no lugar dos microcomputadores, com a finalidade de alcançar um baixo consumo de energia, uma menor geração de calor e um melhor aproveitamento do espaço físico.*

1. Introdução

O acesso a recursos computacionais está presente no cotidiano empresarial e educacional, seja para tarefas simples, como edição de texto, impressão de documento e correio eletrônico, seja para tarefas mais avançadas, como a manipulação de sistemas corporativos informatizados ou ambiente virtual de aprendizagem [CETIC.BR 2017; CETIC.BR 2016]. Esse acesso é realizado por meio de microcomputadores, como computadores pessoais (*PC*), *laptops* e *All in one* com seus respectivos sistemas operacionais.

A forma tradicional de acesso, utilizando microcomputadores, muitas vezes ocupa um espaço físico valioso, além da constante manutenção com *hardware* e *software* pelo profissional de Tecnologia da Informação (TI). Há também a geração de calor em consequência ao seu consumo de energia elétrica, o que é outro ponto negativo. O consumo elétrico, em computadores convencionais, tem uma significativa diferença (aproximadamente 80 %) quando comparado a dispositivos que têm como função básica acessar servidores remotos [Da Silva *et al* 2012].

Embora computadores convencionais venham sendo a escolha mais comum em redes corporativas e laboratórios de ensino, o conceito de área de trabalho remoto,

associado à Computação em Nuvem, oferece benefícios que demonstram ser importantes para o bom funcionamento da TI, como a disponibilidade, o gerenciamento simples e a escalabilidade. Pesquisadores destacaram que, se utilizassem a Nuvem em vez de adquirir computadores convencionais, as universidades teriam uma redução estimada de 42 % nos custos de operação [Chandra e Borah 2012].

Para um trabalho eficaz, seu desenvolvimento deve ser economicamente viável, de fácil implementação e manutenção para acesso a recursos de um sistema operacional (SO), como suas ferramentas e aplicações, proporcionando ao setor de TI maior controle das atividades realizadas pelos usuários, sendo, assim, uma solução voltada para empresas de pequeno e médio porte e instituições de ensino. A proposta visa que a solução – a substituição de *hardware* – seja de baixo consumo de energia, possibilitando, assim, outros diversos benefícios, como a utilização de rede elétrica estabilizada com redundância à falha e menos custos com infraestrutura elétrica.

1.1. Justificativa

Os recursos, ao longo do tempo, têm ficado cada vez mais escassos e caros, como é o caso do espaço físico, da energia elétrica e da mão de obra capacitada. A Tecnologia da Informação tem a importante tarefa de criar mecanismos e soluções para auxiliar gestores a encontrarem melhores formas de otimizar esses recursos.

1.2. Objetivo geral

Realizar um estudo técnico sobre a aplicação de tecnologias alternativas às opções tradicionais ofertadas pelo mercado, em relação ao acesso à estação de trabalho ou de ensino, demonstrando que, em um universo de possibilidades existentes, há soluções que sejam passíveis de desenvolvimento com um baixo custo monetário e um menor impacto ambiental.

1.3. Objetivos específicos

Os objetivos específicos deste trabalho são:

- Realizar um estudo sobre tecnologias referente à virtualização de estações de trabalho e seus benefícios;
- Realizar um levantamento de requisitos para viabilidade do projeto;
- Realizar testes individuais de todos os componentes que compõem o projeto;
- Utilizar um *hardware* de baixo custo monetário e elétrico a fim de substituir o microcomputador;
- Desenvolver um *software* para configuração da aplicação, utilizando a linguagem de programação PHP, juntamente com o *framework* Bootstrap e comandos Linux;
- Aplicar o *software* desenvolvido em ambiente de produção, com a finalidade de validação e obtenção de resultados;
- Apresentar os resultados obtidos.

2. Referencial teórico

Nas próximas seções, serão enfatizados os conceitos essenciais e as ferramentas utilizadas para o desenvolvimento deste trabalho. Serão abordados assuntos como Raspberry Pi, Área de Trabalho Remota, rDesktop, PHP, APACHE, entre outros.

2.1. Raspberry Pi

O *hardware* denominado Raspberry Pi 3 Model B, que foi utilizado como uma solução de baixo custo, é um computador de tamanho reduzido que tem, aproximadamente. Composto por um processador Quad Core 1.2GHz e 1GB de memória RAM, tal equipamento demonstra-se eficiente para executar diversas atividades [Raspberry 2017].

O Raspberry Pi 3 Model B utiliza um cartão SD como armazenamento interno para instalar seu SO, e as entradas e saídas de dados são diversificadas, contando com uma controladora *Ethernet* (RJ-45), *Wireless LAN, Bluetooth* 4.1 e quatro portas USB [Raspberry 2017].

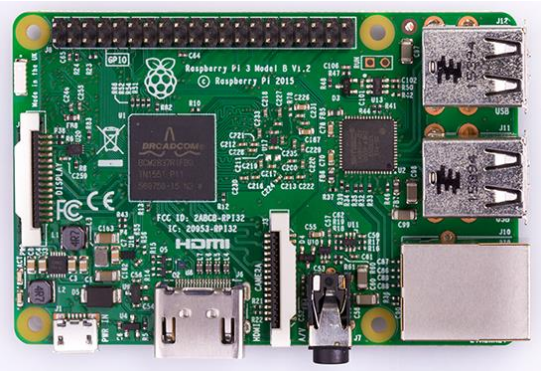

Figura 1. O Raspberry Pi 3 Model B

2.1.1 Raspbian

Raspbian OS é uma distribuição livre, baseada em Linux, desenvolvida e otimizada especialmente para o Raspberry Pi e derivada do Debian, considerado o SO padrão da Raspberry Foundation [Raspbian 2017].

2.2. *Remote Desktop Protocol*

O *Remote Desktop Protocol* (RDP), se utilizado com conexões de rede para acesso a aplicativos baseados no Windows, fornece recursos de exibição por meio de um servidor. Ele é um protocolo com uma capacidade de múltiplos canais, que permite canais virtuais separados para transportar dados de comunicação e apresentação do dispositivo a partir do servidor, assim como dados de teclado e *mouse* do cliente [Microsoft 2017].

2.2.1. rDesktop

O rDesktop é um cliente *UNIX* de código aberto que possibilita a conexão com o serviço de Área de Trabalho Remota do Windows, capaz de comunicar-se nativamente com o RDP para apresentar a área de trabalho do Windows ao usuário [Rdesktop 2017].

2.3. Tecnologias *Web*

O Apache *HTTP Server Project* é um trabalho colaborativo destinado ao desenvolvimento de um *software* robusto que possa ser usado comercialmente, disponibilizando livremente um servidor HTTP (*Web*) [Apache 2017].

Hypertext Markup Language versão 5 (HTML5) é uma linguagem de marcação responsável pelo conteúdo apresentado nas páginas *World Wide Web* (WWW), desta

forma sendo possível a representação gráfica do painel de configuração contido na Raspberry Pi [W3C 2017].

Bootstrap é um *framework* HTML, *Cascading Style Sheets* (CSS) e JavaScript, muito usado nos dias de hoje, pois a sua estrutura facilita o desenvolvimento *Web* por meio de componentes prontos e personalizáveis. Trata-se de uma ferramenta *open-source*, ou seja, gratuita, com atualizações frequentes e que é, assim, aprimorada constantemente [Bootstrap 2017].

JQuery é um *framework* desenvolvido em JavaScript e utilizado para facilitar a manipulação de elementos em HTML, a manipulação de eventos, o controle e criação de animações e as requisições assíncronas em Ajax [Jquery 2017].

O PHP (*Hypertext Preprocessor*) é uma linguagem de programação gratuita e fortemente difundida na internet, em aplicações de diversos segmentos. A linguagem é interpretada ao lado do servidor, ou seja, processa no *back-end*, enquanto o HTML, que é responsável pela apresentação da página web, é processado do lado do cliente, no *front-end* [PHP 2017].

3. Trabalhos relacionados

Os trabalhos e os estudos que serão apresentados a seguir demonstram, em sua parte ou essência, pontos relevantes ao estudo de caso que será empregado neste trabalho. Dessa forma, seus resultados servirão de ajuda ou de inspiração.

3.1. Raspberry Pi: uma solução para monitoramento e controle do tráfego de dados em micro e pequenas empresas

O projeto realizado por [Ribeiro e Pereira 2015], utilizando Raspberry Pi como *hardware* de servidor para monitoramento e controle do tráfego de dados, visa tornar-se um grande recurso tecnológico de baixo custo monetário e de baixa energia, aplicado a empresas que tenham deficiência no controle e na segurança de dados.

A proposta do projeto é utilizar tecnologias livres e estáveis, como o SO baseado em Linux, gerenciador de páginas da web e linguagem de interpretação como PHP, para que seu desenvolvimento continue de baixo custo, porém usando *softwares* de qualidade.

Em sua conclusão, o projeto foi uma solução eficaz, eficiente e com baixo investimento e consumo de energia, e, dessa forma, qualquer micro ou pequena empresa terá um grande recurso ao utilizar essa solução para segurança de dados. Destacou-se que, mesmo o Raspberry Pi sendo um equipamento com configurações limitadas, ele atendeu de forma esperada quando bem trabalhado com o SO adequado e otimizado para suas funcionalidades propostas.

3.2. Virtualização de estações de trabalho: estudo de caso da Batavo Cooperativa Agroindustrial

No trabalho de [Sviercoski e Ranthum 2013], que foi realizado na Batavo Cooperativa Agroindustrial, o departamento de tecnologia recebeu tarefa de implementar uma solução para otimização do tempo de atendimento aos seus usuários internos e de diminuir a necessidade de constantes atualizações de *hardware* e de *software*. A solução encontrada pela equipe técnica foi virtualizar as estações de trabalho utilizando

tecnologias presentes no mercado, como Microsoft Hyper-V, Citrix XenServer e XenApp.

O objetivo principal do trabalho era realizar um estudo das principais soluções de virtualização, dividindo-o em várias etapas, como levantamento de requisitos e testes para validação junto aos usuários e administradores dos sistemas. Ao final do trabalho, o objetivo era reduzir custos sem perder a qualidade do serviço já ofertado.

Os resultados encontrados foram bastante satisfatórios, como o tempo de resposta para resolver algum problema com os equipamentos dos usuários, a vida útil do equipamento, podendo ser triplicada, a heterogeneidade dos sistemas operacionais e os aplicativos, como o suíte Office. Segundo o relato dos usuários que trabalharam nos testes, a virtualização trouxe maior facilidade e estabilidade para seus trabalhos, sendo até melhor que a utilização dos computadores físicos.

3.3. Considerações

Este trabalho pretende realizar um aproveitamento teórico e técnico dos dois trabalhos apresentados. [Ribeiro e Pereira 2015] demonstraram que a utilização do Raspberry Pi em sua aplicação, a qual necessitava de um alto controle de tráfego de dados, é uma solução eficiente de *hardware*. Já o trabalho de [Sviercoski e Ranthum 2013], por meio de um estudo de caso, demonstrou que a virtualização de *desktop* é benéfica em diversos pontos, como a redução de custos, por exemplo.

4. Metodologia

A escolha do Raspberry Pi para desenvolvimento foi baseada em características predominantes do equipamento, como seu tamanho reduzido, o baixo consumo de energia elétrica e as *interfaces Ethernet,* USB e HDMI. Para uma melhor acomodação desse equipamento, será utilizada uma *case* com furação VESA de 100 mm para ser parafusada na traseira do monitor. Assim, será proporcionada uma melhor experiência ao usuário e, também, evitará possíveis problemas de manuseio incorreto, pois se trata de uma placa muito pequena.

O Apache será responsável por disponibilizar o acesso ao painel de configuração do Raspberry Pi através de requisições HTTP oriundas de um navegador que poderá ser off-line diretamente no Raspberry Pi ou remotamente de qualquer local de rede com acesso a mesma *subnet*.

As codificações em PHP contidas no painel de configuração do Raspberry Pi serão responsáveis por todas as interações com o SO, assim, tornando possível todas as funcionalidades necessárias para a solução proposta nesse trabalho.

Como a solução do problema proposto nesse trabalho envolveu *hardware* e *software*, foi necessário dividir o projeto em duas fases. A primeira fase foi a construção do *hardware*, que envolveu colocar o Raspberry Pi em um *case* com seu próprio sistema de resfriamento, cabear o sistema em uma rede local, anexar o *hardware* na seção traseira de um monitor, conforme visto na Figura 2, interligar o cabo HDMI do monitor ao Raspberry Pi e ligar o conjunto na energia elétrica.

Figura 2. Raspberry Pi fixado ao monitor dentro de um *case* vermelho.

Após a montagem do sistema, foi necessário instalar o Raspbian no Raspberry Pi e configurá-lo para reconhecimento dos comandos básicos através da utilização de scripts executados no shell do SO.

Com o SO executando em um estágio de instalação padrão do Raspberry Pi, então foram instalados e configurados os pacotes necessários para a execução particular dessa solução, como o PHP, o Apache e o pacote rDesktop. Além desses pacotes, outros auxiliares para o desenvolvimento foram instalados, como VIM, Webmin, SMB e HTOP.

Após o sistema configurado, seguindo para a segunda fase foi desenvolvido o *software* utilizando a metodologia de desenvolvimento descrita nos itens a seguir.

4.1. *Feature Driven Development* (FDD)

Metodologias ágeis são adaptativas e trabalham com constantes *feedbacks*, ajustando-se a mudanças dos requisitos e, assim, agregando um maior valor ao produto final, contando com a satisfação do cliente e a organização do projeto, que se baseia em prioridades nas funcionalidades do *software* [Barbosa e Libardi 2010].

Dentre os destaques da metodologia ágil, tem-se *Extreme Programming* (XP), SCRUM e *Feature Driven Development* (FDD), sendo este o escolhido para o desenvolvimento deste trabalho, pois ele atende às necessidades surgidas.

A *Feature Driven Develoment* (FDD) é conhecida por absorver práticas de métodos ágeis para a Engenharia de *Softwares* orientada a objeto, sendo uma ferramenta criada em 1997 e publicada para utilização em 1999 [FDD 2017].

FDD é uma metodologia incremental e iterativa; cada iteração é a implementação completa de todas as funcionalidades que formam seu conjunto e que devem ser apresentadas ao cliente, e, assim, este perceberá a evolução conforme avança.

As iterações não devem passar de duas semanas. Caso necessário, a iteração deve ser dividida em mais partes, para que seja respeitada a recomendação [Palmer e Felsing 2002].

Segundo [Palmer e Felsing 2002], a metodologia FDD resume-se na execução dos seguintes processos:

- Desenvolver um modelo inicial: são definidos os requisitos, o domínio do sistema para sua construção e a documentação da especificação das funcionalidades;
- Criar lista de funcionalidades: na lista de funcionalidades, cada funcionalidade é revisada pelo cliente;
- Planejar por funcionalidade: criar funções sequenciais que são executadas de acordo com sua prioridade;
- Arquitetar por funcionalidade e construir por funcionalidade: implementação de algumas funcionalidades selecionadas pelo grupo de desenvolvedores de acordo com sua prioridade; não ultrapassar duas semanas.

4.1.1. Desenvolver modelo inicial

Sendo o primeiro processo do ciclo de vida da metodologia, desenvolver modelo inicial tem como resultado uma arquitetura inicial denominada *object model* (Modelo de Dados Abrangente). Estudos sobre o escopo do projeto e seu contexto são realizados a fim de modelar cada área, tendo-se um domínio do negócio. Para ajudar a entender o processo descritivo, pode-se utilizar o Diagrama de Domínio, o qual ilustra a estrutura física [FDD 2017].

O Diagrama de Domínio, apresentado na Figura 3, tem como finalidade demonstrar, na visão de alto nível, o domínio do sistema, identificando a relação entre todas as entidades e suas dependências.

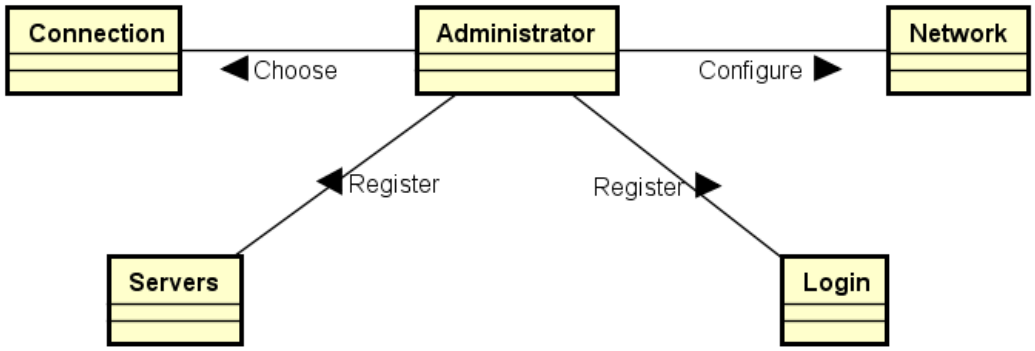

Figura 3. Diagrama de Domínio

4.1.2. Construir lista de funcionalidades

Construir lista de funcionalidades é uma atividade inicial que percorre todo o projeto, tendo como fundamento identificar todas as funcionalidades que satisfaçam os requisitos. O resultado que deve ser obtido é uma lista de Requisitos Funcionais (RF) e Requisitos Não Funcionais (RNF), objetivando todas as necessidades reais do projeto, do ponto de vista do cliente [FDD 2017].

A seguir, são listadas as funcionalidades previstas para este projeto. O documento de requisitos descreve os RNF e os RF. As funcionalidades são relevantes e devem estar de acordo com as necessidades, além de estarem bem estruturadas, com o objetivo de evitar custos maiores com tempo e dinheiro.

RF01. Configurar conexão de rede

RF02. Cadastrar login automático

RF02.1. Definir usuário e senha padrão

RF03. Escolher resolução de vídeo

RF04. Cadastrar lista de servidores de conexão

RF05. Definir servidor padrão de conexão automática

RF05.1. Permitir conexão manual

RF06. Exibir informações gerais das configurações

RNF01. Exigir usuário e senha para configurações

RNF02. Utilizar a linguagem de programação PHP >=5.4

RNF03. Utilizar *framework* Bootstrap

RNF04. Utilizar o mínimo de recursos de processamento

RNF05. Utilizar Raspberry Pi como *hardware*

RNF06. Permitir configuração remota

4.1.3. Planejar por funcionalidade

Assim como construir lista de funcionalidades, planejar por funcionalidade é uma atividade que abrange todo o projeto e tem por objetivo criar um plano de negócio, por meio de um bom estudo e planejamento. Esse trabalho deve levar em consideração a complexidade e a ordem das funcionalidades que serão implementadas com suas dependências. A saída desse trabalho deve ser uma tabela descritiva das funcionalidades e suas estimativas de tempo [FDD 2017].

A Tabela 1 demonstra o planejamento por funcionalidade, contendo a ordem de desenvolvimento e o custo de tempo em dias.

Tabela 1. Planejamento por funcionalidade

Ordem	Funcionalidade	Custo de tempo em dias
1	RF01. Configurar conexão de rede	5 dias
2	RF02. Cadastrar login automático	3 dias
3	RF02.1. Definir usuário e senha padrão	1 dia
4	RF03. Escolher resolução de vídeo	3 dias
5	RF04. Cadastrar lista de servidores de conexão	3 dias
6	RF05. Definir servidor padrão de conexão automática	5 dias
7	RF05.1. Permitir conexão manual	1 dia
8	RF06. Exibir informações gerais das configurações	3 dias

4.1.4. Arquitetar por funcionalidade

A atividade arquitetar por funcionalidade é executada uma vez para cada funcionalidade, com o objetivo de produzir o Pacote de Arquitetura por funcionalidade. São criados, nessa etapa, o Diagrama de Atividade e o Diagrama de Classe [FDD 2017].

A Figura 4 apresenta o Diagrama de Atividade, demonstrando a relação do administrador com o sistema na configuração do dispositivo para conexão com o servidor. O administrador acessa o painel de configuração utilizando seu usuário e senha válidos, faz a configuração da conexão com a rede local, escolhe se a conexão com o servidor será com um usuário e senha pré-definidos, define a resolução de vídeo mais adequada para o monitor que estiver usando e, por fim, cadastra o servidor ou os servidores, que estarão presentes para seleção manual, ou um principal, que terá conexão automática quando inicia o Raspberry Pi.

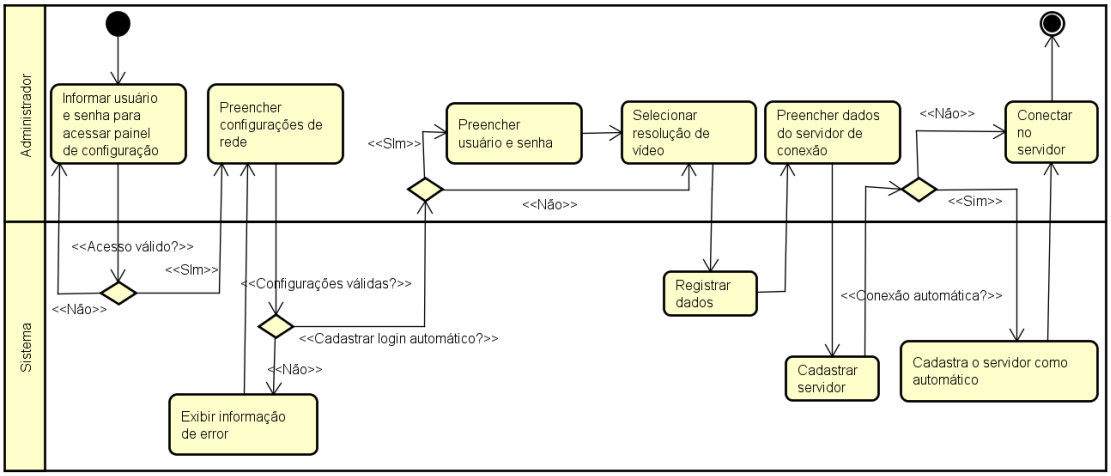

Figura 4. Diagrama de Atividade

A Figura 5 apresenta o Diagrama de Classes do sistema.

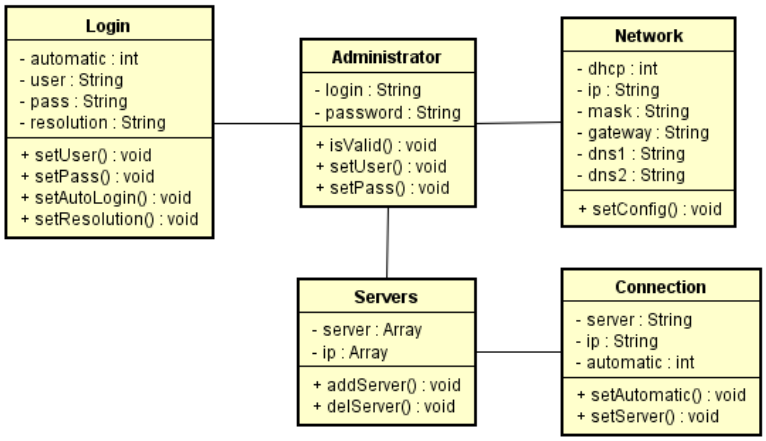

Figura 5. Diagrama de Classes

A classe Administrator é responsável por conter os dados do administrador que estiver acessando o sistema, realizar a validação do acesso e alterar o usuário e a senha quando solicitado.

A classe Login é responsável por administrar os dados do usuário que for realizar o acesso ao servidor, quando definida as credenciais de acesso. A resolução é definida com base no monitor que estiver instalado o Raspberry Pi, mas pode-se usar qualquer outra resolução, quando assim definido.

A classe Rede contém as configurações de acesso à rede local em que estiver instalado o Raspberry Pi. Sem essa configuração, torna-se impossível o ingresso ao

servidor, mesmo já cadastrado. As definições de IP e máscara são obrigatórias, desde que o servidor em questão estiver na mesma *subnet* e *lan* que o Raspberry Pi.

As classes *Servers* e *Connection* são responsáveis por cadastrar e listar os servidores que estão aptos a conectar-se e definir qual terá o acesso realizado, de forma automática, quando iniciado o Raspberry Pi.

4.1.5. Construir por funcionalidade

Seu objetivo é produzir uma função com valor para o cliente. Ao final dessa atividade, será entregue uma funcionalidade testada e pronta. Tal processo inicia-se com o Pacote de Arquitetura pronto, construído na atividade anterior [FDD 2017].

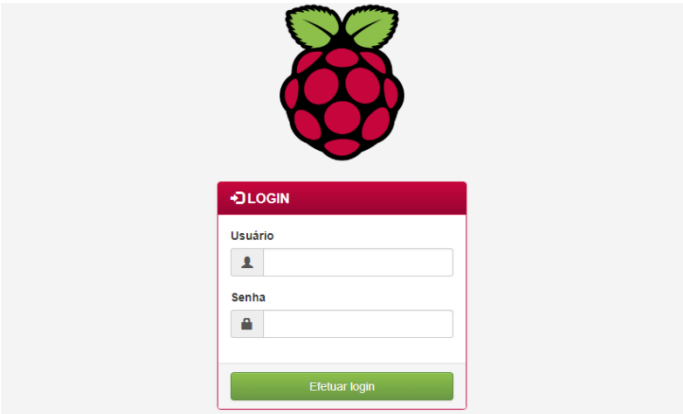

Figura 6. Tela de login

A *interface* inicia com a tela de login, Figura 6, onde o usuário faz sua autenticação para a utilização do sistema. Na configuração padrão, foi utilizado o usuário *admin* e senha *admin*.

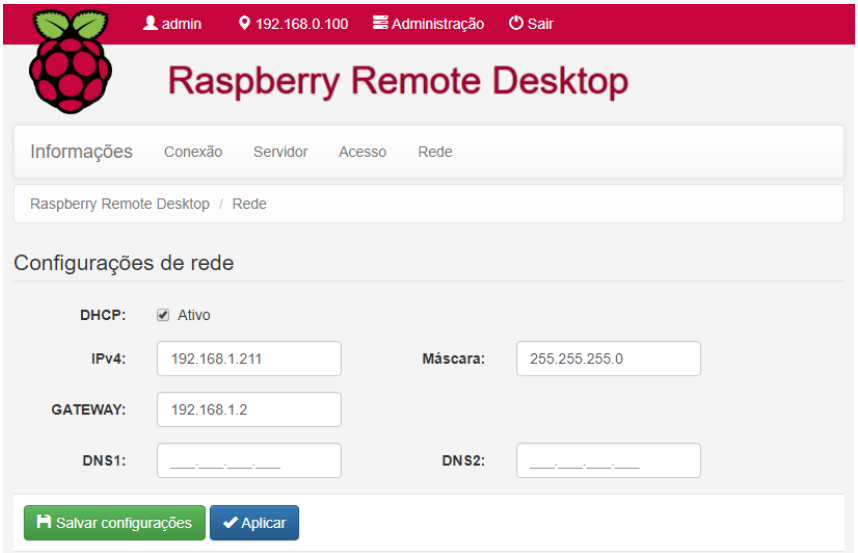

Figura 7. Configurações de rede

A próxima etapa na customização do ambiente é realizar a configuração do acesso ao servidor da área de trabalho remota, onde é preciso se certificar que as definições de rede estão configuradas de forma correta a fim de ingressar na mesma *subnet* do servidor a ser acessado. Essa configuração é apresentada na Figura 7.

Na Figura 8 é possível observar algumas configuração opcionais que podem ser realizadas pelo sistema, para tornar a experiência do usuário mais automática, como definir o login automático no servidor remoto com nome de usuário e senha predefinidos, além da definição de resolução recomendado ao monitor que estiver conectado no Raspberry Pi.

Figura 8. Configurações de acesso ao servidor remoto

Na Figura 9 é mostrada a *interface* de cadastro do servidor na qual será criada a conexão remota. Neste cadastro é preciso especificar o IP de acesso ao servidor, um nome pra sua distinção e a porta de acesso ao protocolo RDP.

Figura 9. Configurações de conexão ao servidor remoto

Figura 10. Quando esta configuração estiver definida como manual o usuário poderá escolher qual servidor dentre os previamente cadastrados deseja conectar,

conforme representado na Figura 9. Caso seja configurada automática a conexão com o servidor escolhido ocorrerá de forma persistente.

Figura 10. Configurações de conexão preferencial

A Figura 11 apresenta todas as configurações realizadas anteriormente para uma visão única das definições. Existe uma funcionalidade extra para desconectar o servidor caso esteja conectado no momento e também a possibilidade de conectar a qualquer outro servidor previamente cadastrado.

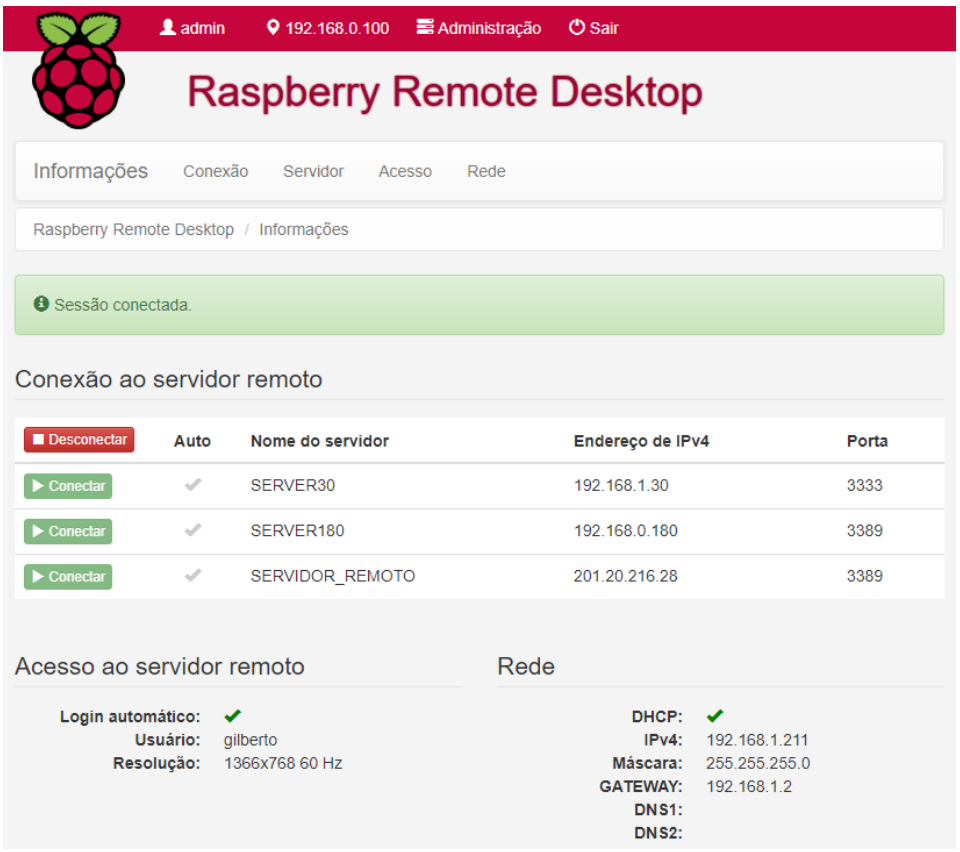

Figura 11. Resumo de todas as configurações e estado da conexão remota

5. Resultados

Após a configuração do hardware e implementação das interfaces para customização das conexões, foi verificado que:

- Foi possível realizar um estudo sobre tecnologias referente á virtualização de estações de trabalho e seus benefícios;
- A viabilidade quanto ao custo de compra de um Raspberry Pi é inferior ao preço de um PC normal e faz o mesmo trabalho básico quando utilizado a ideologia de área de trabalho remota;
- Foram realizados testes individuais de todos os componentes que compõem o projeto, quando recebidos de sua compra;
- Foi desenvolvido um software para configuração da aplicação, utilizando a linguagem de programação PHP, juntamente com o framework Bootstrap e comandos Linux;
- O software desenvolvido em ambiente de produção, com a finalidade de validação e obtenção de resultados, com os usuários acostumados a desktop virtual não observaram diferença na sua produtividade quando usando Raspberry Pi;

Além disso, cabe ressaltar as dificuldades encontradas para desenvolver o projeto, explicadas a seguir.

5.1. Dificuldades encontradas

Este projeto proporcionou diversos desafios ao longo de sua execução, foram questões técnicas que a primeira vista parecia de fácil implementação, mas como o projeto exigia uma interação muito grande entre o SO, *hardware* e o *software* proposto ocorriam que por muitas vezes algumas ações que deveriam ser executadas simultaneamente ficavam pelo caminho e para solucionar, na maioria das ocasiões precisava ser modificada a forma como determinada instrução era realizada em detrimento do sequenciamento que deveria acontecer.

Dentre todos os obstáculos vencidos, três situações marcaram minha trajetória de desenvolvimento até a finalização do projeto.

O primeiro obstáculo ocorreu logo após o término da instalação e configuração do SO do Raspberry Pi 3 denominado Raspbian. Como o *software* é escrito em PHP era preciso executar comandos Linux para interações com o SO e em algumas instruções havia o requerimento de ser executado por um “super usuário” o qual não era possível por padrão. Após muitas tentativas de diversas formas foi encontrada a solução disposta na Figura 12.

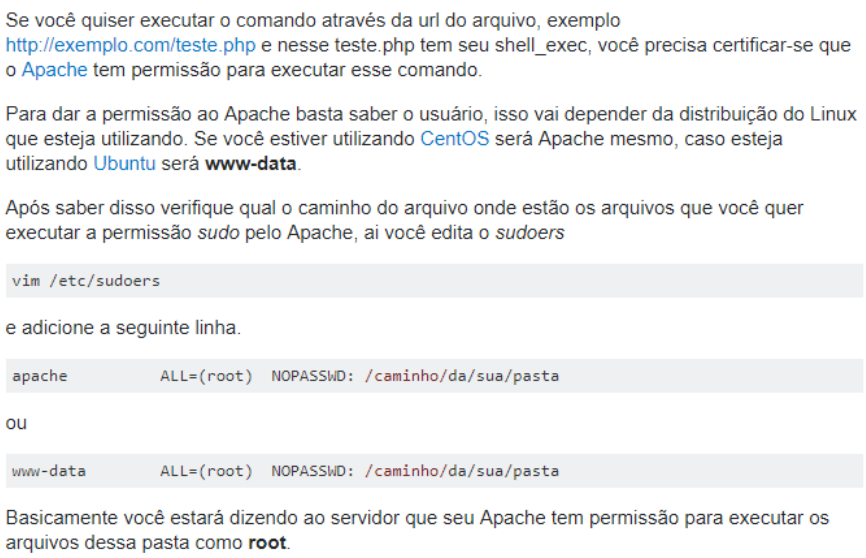

Figura 12. PHP executando comando sudo com shell_exec() [Stackoverflow 2017]

O segundo obstáculo ocorreu quando foi preciso executar uma instrução Linux através do PHP, mas como se fosse outro usuário executando, isto era preciso porque o comando em questão iniciava a instância gráfica da área de trabalho remota e como no Raspberry Pi a GUI pertence ao usuário "pi" é um pré-requisito que seja chamado por ele, outro requerimento para o funcionamento da instância era especificar aonde seria exibida, isto era uma situação nunca antes ocorrida, mas a solução para ambas as questões estão dispostas nas Figuras 13 e 14 respectivamente.

```
sudo -H -u otheruser bash -c 'echo "I am $USER, with uid $UID"'
```

Figura 13. Run a shell script as another user that has no password [Askubuntu 2017]

```
ssh pi@raspberrypi.local 'DISPLAY=:0 name-of-program-to-run'
```

Figura 14. Start local application with GUI, over ssh [Stackexchange 2017]

O terceiro e último obstáculo, mas não menos importante aconteceu com a necessidade de não apenas executar determinado *script* de comandos Linux, mas de ter o controle sobre determinado conjunto de interação com o SO. Para esse problema, a solução encontrada foi criar um serviço no Raspbian, tendo a possibilidade de recursos que não existiam como: iniciar, parar e iniciar automaticamente [Github 2017].

6. Resultados

Baseado nos resultados atingidos por esse trabalho, foi possível obter sucesso naqueles objetivos específicos desejados no início do projeto. Foi realizado um estudo sobre tecnologias de virtualização, entendendo os benefícios e aplicando eles em um ambiente real e funcional. A viabilidade do projeto foi considerada positiva, visto a diminuição de custo aplicando o conceito de virtualização com Raspberry Pi na construção de um ambiente de uso computacional ao invés da utilização de computadores tradicionais considerados bem mais caros.

Foi também desenvolvido um software para configuração do ambiente de uso do usuário final, utilizando conhecimentos de linha de comando no SO Linux, *interface* baseada em PHP integrada com o *framework* Bootstrap. Por fim, foram aplicados os conceitos do protótipo em ambiente de produção validando a usabilidade de todo o projeto como um todo.

Conclui-se que o projeto atingiu os objetivos positivamente e como trabalhos futuros espera-se utilizá-lo em ambientes de produção diminuindo custos e aumentando a inserção da virtualização com Raspberry Pi na construção de "salas computacionais".

Para trabalhos futuros, imagina-se que seria pertinente o desenvolvimento de uma interface para um ambiente de configuração único, que pudesse customizar vários sistemas como esse desenvolvido no projeto. Ao invés de configurar máquina a máquina, em um laboratório poderia haver uma maneira de configuração de todas as máquinas ao mesmo tempo.

Referências

Apache (2017). About Apache. Disponível em: http://httpd.apache.org/docs-project/. Acessado em Maio de 2017.

Barbosa, A., Azevedo, B., Pereira, B., Campos, P. and Santos, P. (2009). Metodologia Ágil: Feature Driven Development.

Libardi, Paula L. O.; Barbosa, Vladimir. (2010). Métodos Ágeis. Limeira: Universidade Estadual de Campinas.

Bootstrap. (2017). Getting Started. Disponível em: http://getbootstrap.com/getting-started/. Acessado em Maio de 2017.

CETIC.BR. (2017). TIC Empresas 2015: Pesquisa sobre o uso das Tecnologias de Informação e Comunicação nas empresas brasileiras. ed. São Paulo: CGI.br.

CETIC.BR. (2016). TIC Educação 2015: Pesquisa sobre o uso das Tecnologias de Informação e Comunicação nas escolas brasileiras. ed. São Paulo: CGI.br.

Chandra, D. G. and Borah, M. D. (2012). Cost benefit analysis of cloud computing in education: In International Conference on Computing. Communication and Applications.

Da Silva, D. A. et al. (2012). Enabling green IT through building a virtual desktop infrastructure. In Proceedings - 8th International Conference on Semantics, Knowledge and Grids, SKG.

FDD. (2017). Feature Driven Development. Disponível em: http://www.featuredrivendevelopment.com. Acessado em Maio de 2017.

Jquery. (2017). jQuery. Disponível em: https://jquery.com/. Acessado em Junho de 2017.

Microsoft. (2017). Remote Desktop Protocol: Features and Performance. Disponível em: http://www.microsoft.com/. Acessado em Abril de 2017.

Palmer, Stephen R.; Felsin, John. M. (2002). A Practical Guide to Feature Driven Development. Prentice Hall.

PHP. (2017). Documentation, Disponível em: http://php.net/docs.php. Acessado em Maio de 2017.

Raspberry. (2017). What is a Raspberry Pi. Disponível em: https://www.raspberrypi.org. Acessado em Maio de 2017.

Raspbian. (2017). Raspbian. Disponível em: http://www.raspbian.org. Acessado em Maio de 2017.

Rdesktop. (2017). rDesktop: A Remote Desktop Protocol Client. Disponível em: http://www.rdesktop.org/. Acessado em Maio de 2017.

Ribeiro, T. A.; Pereira, H. G. G. (2015). Raspberry Pi: uma solução para monitoramento e controle do tráfego de dados em micro e pequenas empresas.

Sviercoski, A. and Ranthum, G. (2013). Virtualização de estações de trabalho. Estudo de caso da batavo cooperativa agroindustrial.

W3C. (2017). Disponível em: https://www.w3.org. Acessado em Maio de 2017.

Stackoverflow (2017). PHP executar comando sudo com shell_exec(). Disponível em: https://pt.stackoverflow.com/questions/164842/php-executar-comando-sudo-com-shell-exec. Acessado em Outubro de 2017.

Stackexchange (2017). Start local application with GUI, over ssh. Disponível em: https://raspberrypi.stackexchange.com/questions/6753/start-local-application-with-gui-over-ssh. Acessado em Outubro de 2017.

Askubuntu (2017). Run a shell script as another user that has no password. Disponível em: https://askubuntu.com/questions/294736/run-a-shell-script-as-another-user-that-has-no-password. Acessado em Outubro de 2017.

Github (2017). Contador de Seguidores Instagram com Raspberry Pi. Disponível em: https://github.com/giobauermeister/ig-followers-counter-raspberrypi. Acessado em Outubro de 2017.