UM SISTEMA DE GESTÃO REMOTA PARA SERVIDORES EM DATA CENTERS

Ewerton Vasconcelos da Silva

Projeto de Graduação apresentado ao Curso de Engenharia Eletrônica e de Computação da Escola Politécnica, Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Engenheiro.

Orientador: Rodrigo de Souza Couto

Rio de Janeiro

Julho de 2021

UM SISTEMA DE GESTÃO REMOTA PARA SERVIDORES EM DATA CENTERS

Ewerton Vasconcelos da Silva

PROJETO DE GRADUAÇÃO SUBMETIDO AO CORPO DOCENTE DO CURSO DE ENGENHARIA ELETRÔNICA E DE COMPUTAÇÃO DA ESCOLA POLITÉCNICA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE ENGENHEIRO ELETRÔNICO E DE COMPUTAÇÃO

Autor:

Ewerton Vasconcelos da Silva

Orientador:

Prof. Rodrigo de Souza Couto, D.Sc.

Examinador:

Prof^{a}. Fernanda Duarte Vilela Reis de Oliveira, D.Sc.

Examinador:

Prof. Roberto Gonçalves Pacheco, M.Sc.

Rio de Janeiro

Julho de 2021

Declaração de Autoria e de Direitos

Eu, *Ewerton Vasconcelos da Silva* CPF *159.747.267-07*, autor da monografia *UM SISTEMA DE GESTÃO REMOTA PARA SERVIDORES EM DATA CENTERS*, subscrevo para os devidos fins, as seguintes informações:

1. O autor declara que o trabalho apresentado na disciplina de Projeto de Graduação da Escola Politécnica da UFRJ é de sua autoria, sendo original em forma e conteúdo.

2. Excetuam-se do item 1. eventuais transcrições de texto, figuras, tabelas, conceitos e ideias, que identifiquem claramente a fonte original, explicitando as autorizações obtidas dos respectivos proprietários, quando necessárias.

3. O autor permite que a UFRJ, por um prazo indeterminado, efetue em qualquer mídia de divulgação, a publicação do trabalho acadêmico em sua totalidade, ou em parte. Essa autorização não envolve ônus de qualquer natureza à UFRJ, ou aos seus representantes.

4. O autor pode, excepcionalmente, encaminhar à Comissão de Projeto de Graduação, a não divulgação do material, por um prazo máximo de 01 (um) ano, improrrogável, a contar da data de defesa, desde que o pedido seja justificado, e solicitado antecipadamente, por escrito, à Congregação da Escola Politécnica.

5. O autor declara, ainda, ter a capacidade jurídica para a prática do presente ato, assim como ter conhecimento do teor da presente Declaração, estando ciente das sanções e punições legais, no que tange a cópia parcial, ou total, de obra intelectual, o que se configura como violação do direito autoral previsto no Código Penal Brasileiro no art.184 e art.299, bem como na Lei 9.610.

6. O autor é o único responsável pelo conteúdo apresentado nos trabalhos acadêmicos publicados, não cabendo à UFRJ, aos seus representantes, ou ao(s) orientador(es), qualquer responsabilização/ indenização nesse sentido.

7. Por ser verdade, firmo a presente declaração.

Ewerton Vasconcelos da Silva

Ewerton Vasconcelos da Silva

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
Escola Politécnica - Departamento de Eletrônica e de Computação
Centro de Tecnologia, bloco H, sala H-217, Cidade Universitária
Rio de Janeiro - RJ CEP 21949-900

Este exemplar é de propriedade da Universidade Federal do Rio de Janeiro, que poderá incluí-lo em base de dados, armazenar em computador, microfilmar ou adotar qualquer forma de arquivamento.

É permitida a menção, reprodução parcial ou integral e a transmissão entre bibliotecas deste trabalho, sem modificação de seu texto, em qualquer meio que esteja ou venha a ser fixado, para pesquisa acadêmica, comentários e citações, desde que sem finalidade comercial e que seja feita a referência bibliográfica completa.

Os conceitos expressos neste trabalho são de responsabilidade do(s) autor(es).

DEDICATÓRIA

À minha mãe e avó, as mulheres mais fortes que já conheci.

AGRADECIMENTO

Minha curta trajetória só tem seus marcos e sucessos devido às pessoas que fui encontrando em meu caminho e que foram boas, me orientaram, ajudaram e deram oportunidades das quais eu sempre serei grato.

Obrigado à Escola Técnica Henrique Lage e seus integrantes, nesse lugar eu conheci o poder transformador da educação, ganhei amigos e recebi grandes oportunidades.

Obrigado também aos amigos da Nokia que me acolheram, expandiram minha visão de mundo e me deram oportunidades incríveis, vocês foram essenciais.

E ainda, obrigado à UFRJ por me fornecer um ensino de tanta qualidade, professores compromissados e capacitados, colegas e amigos que sempre me fizeram crescer em vários aspectos.

Por fim, agradeço a todos que estiveram sempre ao meu lado. Ter com quem contar faz a diferença.

RESUMO

Os centros de processamento de dados, mais comumente chamados de *data centers*, são estruturas físicas nas quais conjuntos de dispositivos necessários para o funcionamento de um sistema de computação são alocados. Um exemplo desses dispositivos são os servidores de computação.

A gestão e manutenção dos servidores do *data center* pode ser uma tarefa complexa, dependendo da quantidade de dispositivos existentes. Depender da presença de seres humanos junto aos servidores para manutenções críticas, como reinstalar um sistema operacional ou reiniciar um servidor travado em processo de inicialização, se torna um ponto de falha na capacidade de disponibilidade de um *data center*.

Essa limitação é resolvida por diversos fabricantes de servidores, que já possuem microcontroladores integrados em suas placa-mães. Esses microcontroladores são responsáveis por disponibilizar ao administrador do *data center* uma série de recursos de gestão remota, possibilitando a execução de tarefas que, outrora só poderiam ser realizadas localmente. Esses microcontroladores são chamados de BMC (*Baseboard Management Controller*).

Para as aplicações e instituições nas quais o acesso a servidores profissionais não é possível ou justificável pelo seu custo-benefício, geralmente os *data centers* são montados com componentes destinados a computadores pessoais, por terem menor custo e atenderem a requisitos de performance. Nesses casos, recursos como um BMC raramente estão disponíveis. Visando esse segmento, o projeto apresentado tem como objetivo desenvolver todos os módulos, físicos e de *software*, para a gestão remota e centralizada dos servidores pertencentes a um *data center*. Com isso, o administrador tem a sua disposição todos os principais recursos de gestão de servidores disponíveis para qualquer computador. A implementação proposta utiliza componentes de fácil acesso e baixo custo, visando justificar a sua aplicação quando comparada com soluções comerciais.

Palavras-Chave: *data center*, Baseboard Management Controller, gestão remota, servidores.

ABSTRACT

Data centers are physical structures where sets of devices necessary for the proper functioning of a computing system are allocated. Among these devices, we have, for the most part, computing servers.

The management and maintenance of these servers can be a complex task depending on the number of existing devices. Relying on the presence of human beings at the servers for critical maintenance, such as reinstalling an operating system or restarting a frozen server in the process of booting, becomes a point of failure in the availability of a data center.

This limitation is solved by several server manufacturers, who already have microcontrollers integrated in their motherboards. These microcontrollers are responsible for providing to the data center administrator with a series of remote management resources. These microcontrollers are called BMC (Baseboard Management Controller).

For applications and institutions in which access to professional segment servers is not possible or justifiable for their cost-effectiveness, data centers are generally assembled with components intended for personal computers, as they are less expensive and meet the performance requirements. In these cases, features such as a BMC are rarely available. Aiming at this segment, the presented project goal is to develop all modules, hardware and software, for remote and centralized management of servers belonging to a data center. With the proposed system, the administrator has at his disposal all the main server management resources available for any computer. The proposed implementation uses components that are easy to access and low cost, aiming to justify its application when compared to commercial solutions.

Keywords: data center, BMC, Baseboard Management Controller, remote management, servers.

SIGLAS

BMC - *Baseboard Management Controller*

RDP - *Remote Desktop Protocol*

SSH - *Secure Shell*

HTML - *Hypertext Markup Language*

CSS - *Cascading Style Sheets*

IPMI - *Intelligent Platform Management Interface*

BIOS - *Basic Input/Output System*

KVM - *keyboard, Video and Mouse*

IP - *Internet Protocol*

PCI - *Peripheral Component Interconnect*

VPN - *Virtual Private Network*

HID - *Human Interface Device*

CPU - *Central Processing Unit*

RAM - *Random Access Memory*

USB - *Universal Serial Bus*

HCI - *Host Controller Interface*

OHCI - *Open Host Controller Interface*

UHCI - *Universal Host Controller Interface*

EHCI - *Enhanced Host Controller Interface*

xHCI - *eXtensible Host Controller Interface*

HDMI - *High Definition Multimedia Interface*

ATX - *Advanced Technology eXtended*

GPIO - *General Purpose Input/Output*

PCB - *Printed Circuit Board*

LED - *Light-emitting Diode*

HTTP - *Hypertext Transfer Protocol*

DCM App - *Data Center Manager Application*

API - *Application Programming Interface*

SQL - *Structured Query Language*

WSGI - *Web Server Gateway Interface*

Sumário

1 Introdução **1**

1.1 Tema e Justificativa 1

1.2 Objetivos do projeto 2

1.3 Metodologia 3

1.4 Organização do texto 3

2 Projetos e soluções relacionadas **5**

2.1 Soluções de código aberto 5

2.1.1 OpenBMC 5

2.1.2 PiBMC 6

2.1.3 Pi-KVM 7

2.2 Soluções Proprietárias 8

2.2.1 MegaRAC Serie G, Peppercorn eRIC e Aten IP8000: 8

2.2.2 AdderLink IPEPS e DOMINION KX IV-101: 9

2.3 Análise da pesquisa 9

3 Arquitetura e desenvolvimento do sistema **11**

3.1 Visão geral e topologia da solução 11

3.2 Arquitetura do *hardware* 14

3.2.1 Servidor central 14

3.2.2 Módulo embarcado de gerência 18

3.3 Arquitetura do *software* 28

3.3.1 *Software* do servidor central 29

3.3.2 *Software* do módulo embarcado de gerência 41

4 Apresentação do Sistema **42**

4.1 Tela de *login* 42

4.2 *Dashboard* 43

4.3 Perfil do Usuário 44

4.3.1 Troca de foto 44

4.3.2 Administração de perfil 45

4.4 Lista de servidores 46

4.4.1 Cadastro de um novo servidor 46

4.4.2 Remoção de servidor da lista 47

4.5 Tela de gerência 48

4.6 Gestão de usuários 51

4.7 Casos de uso 53

4.7.1 Acesso à BIOS 53

4.7.2 Monitoramento de tempo ativo 54

4.7.3 Configurações iniciais de maneira remota 55

4.7.4 Controle de energia 55

4.8 Sobre o sistema 56

5 Testes e resultados **57**

5.1 Uso dos controladores USB 57

5.2 Ocupação de recursos do servidor central 61

6 Conclusões e Trabalhos Futuros **64**

Bibliografia **66**

A Ferramentas para o monitoramento do sistema do servidor central **68**

A.1 Instalação do *usbtop* 68

A.2 Como instalar o *usbview* 70

Lista de Figuras

3.1 Visão geral da arquitetura do sistema. 12
3.2 Acesso ao servidor gerenciado por VPN. 13
3.3 Módulo de gerência e número de portas USBs utilizadas. 15
3.4 *Stack* USB (baseado em [1]). 16
3.5 Visão interna do módulo embarcado de gerência. 19
3.6 Gabinete sem tampa lateral do protótipo do servidor gerenciado com módulo embarcado de gerência (em destaque). 20
3.7 Raspberry Pi e módulo conversor HDMI para CSI-2. 21
3.8 Placa de captura HDMI-USB. 22
3.9 Segunda placa de captura HDMI-USB adquirida. 23
3.10 Arduino Pro Micro e interfaces utilizadas. 23
3.11 Conversor Serial-USB. 25
3.12 Conector do painel frontal de uma placa-mãe. 26
3.13 Circuito da conexão do Arduino com pinos de PWR_SW e RST_SW. 27
3.14 Representação simplificada do *hardware*, sistema operacional e *software* do servidor central. 29
3.15 Fluxo de funcionamento do µStreamer. 31
3.16 Criação do cliente da aplicação no Keycloak. 32
3.17 Autenticação da aplicação com Keycloack. 33
3.18 DCM App. 35
3.19 *Templates* Flask para telas do sistema web. 36
3.20 Fluxo de comunicação do teclado remoto. 37
3.21 Fluxo de comunicação do módulo de verificação do estado de energia. 38
3.22 Definição da classe de servidores utilizando SQLAlchemy. 40
3.23 Tabela criada pelo SQLAlchemy no banco de dados. 40

3.24 Implementação de códigos no Arduino 41

4.1 Tela de *login* do sistema. 42
4.2 Tela inicial da aplicação 43
4.3 Tela de perfil do usuário. 44
4.4 Carregar foto de perfil. 45
4.5 Configurações do perfil no Keycloack. 45
4.6 Lista de servidores 46
4.7 Formulário de adição de novo servidor. 47
4.8 Remoção servidor. 48
4.9 Cartão de confirmação. 48
4.10 Lista de servidores com destaque ao botão de acesso à tela de gerência. 49
4.11 Tela de gerência. 49
4.12 Cartões de estado de energia. 50
4.14 Tela inicial do Keycloack. 52
4.15 *Login* na interface de administração. 52
4.16 *Login* na interface de administração. 53
4.17 Acesso à BIOS de um servidor. 54
4.18 Gráfico de tempo ativo dos servidores. 55
4.19 Controles de energia. 55
4.20 Sobre o sistema. 56

5.1 Distribuição das placas de captura nos controladores USB. 58
5.2 Consumo de banda USB sem capturas. 58
5.3 Consumo de banda USB com acesso a um servidor. 59
5.4 Consumo de banda USB com acesso a dois servidores. 60
5.5 Exemplificação da realização dos testes de hardware. 62

A.1 Ferramenta *usbtop* 69
A.2 Ferramenta *usbview* 70

Lista de Tabelas

3.1 Matriz de conexão entre o conversor Serial-USB e o Arduino. 25
3.2 Requisitos mínimos de *hardware* do Linux Lubuntu 18.04. 30

5.1 Acesso a um servidor por cliente. 62
5.2 Acesso a dois servidores por cliente. 62

Capítulo 1

Introdução

1.1 Tema e Justificativa

A gestão e manutenção de servidores de um *data center* pode ser uma tarefa complexa dependendo dos recursos disponibilizados pelos mesmos. Depender da presença de seres humanos junto aos servidores para manutenções, como a reinstalação de um sistema operacional, pode ser um ponto de falha na disponibilidade de um *data center*, já que nem sempre é possível garantir que haja um técnico disponível para realizar manutenções inesperadas.

Comercialmente é possível encontrar uma variedade de servidores de uso profissional que possuem, já integrados às suas placas-mãe, módulos destinados ao gerenciamento do hardware. Esses módulos que fornecem recursos de gestão remota ao administrador são denominados BMC (*Baseboard Management Controllers*).

Entretando, os modelos de servidores que fornecem tal recurso possuem alto custo de aquisição pois tais servidores, por serem direcionados a aplicações comerciais, possuem um alto poder computacional, o que eleva o seu preço. Assim, uma prática observada regularmente em pequenas empresas, instituições de ensino ou *data centers* experimentais é a construção dos próprios servidores com a utilização de componentes direcionados ao segmento de computadores pessoais. Com isso, tais servidores não possuem módulos BMC, o que limita sua gestão remota.

Assim, o administrador é limitado a acessar o servidor através de aplicações como clientes RDP (*Remote Desktop Protocol*) ou SSH (*Secure Shell*). Dessa forma, não é possível realizar customizações em parâmetros que não são acessíveis pelos

sistemas operacionais, como as configurações de BIOS (*Basic Input/Output System*), ou efetuar tarefas que interrompam os serviços de Internet. Logo sendo necessário estar presencialmente com o servidor para realizar essas atividades. Considerando a existência de tais limitações, o projeto propõe o desenvolvimento de um sistema de gestão remota para servidores em *data centers*, tendo em vista centralizar a gestão de todos os servidores através de uma única aplicação via Internet, sem depender de aplicações sendo executadas nos servidores gerenciados.

Dessa forma, a solução necessita de um servidor dedicado para o monitoramento dos servidores do *data center*. Esse servidor é chamado de servidor central e que, além do monitoramento dos servidores do data center, é responsável por disponibilizar a interface web, prover o sistema de autenticação e executar o *backend* da aplicação do projeto proposto por este trabalho. Adicionalmente, tem-se um módulo que é embarcado em cada um dos servidores geridos.

Com isso, é possível proporcionar, integralmente, a capacidade de gestão remota a servidores e computadores mais simples e que não possuem tais recursos nativamente. Portanto, tal sistema de gerenciamento supre as necessidades de diversas instituições que utilizam servidores montados com componentes destinados a computadores pessoais.

1.2 Objetivos do projeto

O objetivo deste projeto é desenvolver módulos necessários de *software* e *hardware* que compõem um sistema de gestão remota para servidores em *data centers* de forma centralizada, através de uma única aplicação web.

Dessa forma, tem-se como objetivos específicos:

- Desenvolver a aplicação web do sistema de gerenciamento e integrá-la a um sistema de autenticação de usuários;
- Conceder ao usuário do sistema de gerência uma interface em que seja possível visualizar a saída de vídeo dos servidores gerenciados, além de emular um teclado remoto capaz de enviar comandos pelo servidor central;
- Elaborar um módulo embarcado de gerência de modo que seja instalado no

gabinete de forma simples e sem impacto ao servidor gerenciado;

- Disponibilizar o *software* do servidor central de maneira que possa ser facilmente instalado e configurado, no qual um *script* instalador é responsável por criar e configurar todo o ambiente.

1.3 Metodologia

Para o desenvolvimento do sistema são utilizadas diferentes tecnologias de *software* e *hardware*, com o objetivo de atingir a melhor usabilidade e possibilitar a implementação das funcionalidades propostas.

A parte responsável pelo monitoramento e interfaces é um servidor central, onde se concentra a maior parte do desenvolvimento e das tecnologias utilizadas.

Para a implementação da interface web são utilizadas linguagens como HTML (*Hypertext Markup Language*), CSS (*Cascading Style Sheets*), JavaScript, Python, além dos *frameworks* Bootstrap e Flask. O *backend* da aplicação é composto por diversos módulos que incluem o servidor web *Apache*, aplicações em *Python*, um banco de dados *PostgreSQL*, um sistema de autenticação Keycloak, entre outros módulos. Alguns dos módulos são executados em contêineres Docker.

No desenvolvimento do sistema são executados testes para a seleção de componentes e módulos, como a escolha da placa de captura USB (Universal Serial Bus) e o modelo de Arduino que atendam às necessidades do projeto.

1.4 Organização do texto

Este trabalho está organizado da seguinte forma. No Capítulo 2 é apresentada uma relação de trabalhos relacionados, de sistemas que objetivam resolver o problema abordado neste projeto. Já no Capítulo 3 é feita uma análise detalhada de todos os módulos que compõem o sistema, desde os componentes de *hardware* e *software*, explicitando e justificando todas as escolhas de projeto. O Capítulo 4 tem o objetivo de apresentar o sistema e demonstrar sua operação ao usuário final. Ademais, o Capítulo 5 apresenta uma série de testes e resultados, que objetivam fornecer uma especificação precisa do sistema. Por fim, o Capítulo 6 apresenta as

considerações finais do trabalho e a descrição de possíveis melhorias para o sistema em trabalhos futuros.

Capítulo 2

Projetos e soluções relacionadas

O objetivo deste capítulo é identificar projetos de segmentos similares e apontar soluções existentes para os problemas detectados, a partir de uma relação de trabalhos relacionados. A pesquisa está dividida entre soluções de código aberto e soluções proprietárias de código fechado.

2.1 Soluções de código aberto

Dentre as soluções de código aberto encontradas, são descritas nessa seção o OpenBMC, PiBMC e o Pi-KVM.

2.1.1 OpenBMC

O OpenBMC [2] é um projeto de código aberto, que é composto unicamente por software, que tem o objetivo de ser instalado em chips BMC comerciais.

Esse projeto é resultado da colaboração de grandes indústrias do setor de tecnologia como Facebook, Amazon, IBM e Microsoft. Esse projeto é definido como uma pilha de *software* padrão, que reúne uma série de recursos, como um aplicativo web para interagir com a pilha de *firmware* e suporte *Redfish* [3], que é uma API para gerenciamento de hardware. Além disso, o OpenBMC é capaz de funcionar em um conjunto predefinido de chips BMC, presentes em servidores comerciais.

Tal projeto se diferencia deste trabalho, principalmente por ser desenvolvido para chips BMC comerciais e servidores que os contém. Isso faz seu uso estar condicionado ao alto investimento em um *hardware* específico e não atender servidores

que não possuem um BMC integrado.

A análise do OpenBMC aborda os seguintes aspectos:

- **Objetivo do projeto.** O alvo do projeto são servidores comerciais. Assim, o OpenBMC é projetado para funcionar em uma lista predefinida de modelos de BMC e fabricantes.
- **Licença do *software*.** O código é aberto e regido pela licença Apache.
- **Tipo de *hardware*.** O projeto é desenvolvido para uma lista específica de chips BMC que já vem integrados às placas-mãe dos servidores.
- **Funcionalidades.** O OpenBMC dispõe de todas as funcionalidades de gestão remota disponíveis atualmente como uma aplicação web, teclado e mouse virtual, entre outras funções. Como o *hardware* está na placa-mãe do servidor, a aplicação dispõe de acessos direto a módulos e sensores do servidor. Isso viabiliza uma série de funcionalidades adicionais, como a IPMI (*Intelligent Platform Management Interface*), que é uma interface padronizada para gerência de hardware.

2.1.2 PiBMC

O PiBMC [4] é um projeto de código aberto, que inclui também o desenvolvimento de um hardware e tem por objetivo a implementação de um BMC utilizando um Raspberry Pi. Trata-se de um projeto tecnicamente simples, pois possui como código principal um *script shell* e que provê uma interface de linha de comando no qual o usuário pode realizar as operações de gestão.

O PiBMC, quando comparado com o projeto deste trabalho, se mostra limitado e não fornece funcionalidades como vídeo e teclado remoto ao administrador do *data center*. Por isso, o seu uso limita ações do administrador.

Para a análise do PiBMC os seguintes aspectos são abordados:

- **Objetivo do projeto.** O foco do PiBMC são computadores e servidores que não possuem nativamente um chip BMC.
- **Licença do *software*.** O código é aberto e está disponibilizado no GitHub do autor.

- **Tipo de *hardware*.** O projeto utiliza um Raspberry Pi.
- **Funcionalidades.** É um projeto limitado em termos de funcionalidades. Possui somente um acesso via SSH com funções básicas de comando como ligar e desligar o servidor. O autor propõe também um acesso via RS232 [5], é que um padrão de comunicação, à placa-mãe para o acesso às funções da BIOS. Porém, essa funcionalidade não está disponível para todo tipo de *hardware*, sendo mais comumente encontrada em servidores mais antigos e do tipo *Rackmount* [6].

2.1.3 Pi-KVM

O PiKVM [7] é um projeto de código aberto iniciado pelo desenvolvedor Maxim Devaev. Ele implementa um KVM (*Keyboard, video and mouse*) sobre IP (*Internet Protocol*) baseado em Raspberry Pi, que ajuda a gerenciar servidores ou estações de trabalho remotamente. É um projeto que se assemelha com a proposta deste trabalho em termos de funcionalidades e utiliza como plataforma principal o Raspberry Pi, onde cada módulo é responsável por gerenciar um único servidor.

O ponto que pode ser destacado referente à comparação do PiKVM com o projeto deste trabalho, é que com o uso do projeto PiKVM, a administração é feita de uma maneira não centralizada e cada servidor possui seu hardware e sistema. O projeto proposto centraliza as operações do *data center* em uma única interface e sistema de autenticação, o que simplifica a administração.

Para o projeto Pi-KVM, os aspectos analisados são:

- **Objetivo do projeto.** O Pi-KVM é destinado a computadores e servidores que não possuem nativamente um chip BMC.
- **Licença do *software*.** O código é aberto e está disponibilizado no GitHub do autor.
- **Tipo de *hardware*.** Esse projeto utiliza um Raspberry Pi como plataforma principal e outros periféricos adicionais.
- **Funcionalidades.** É um projeto completo em termo de funcionalidades, disponibilizando acesso, teclado e mouse remotos, além de montagem de mídia externas, entre outras funções. O que distancia esse projeto deste trabalho é a plataforma para o qual ele é desenvolvido e a proposta de gestão centralizada.

O Pi-KVM é baseado no Raspberry Pi e projetado para que cada módulo gerencie um servidor. O sistema deste trabalho utiliza a combinação de um servidor central com um módulo embarcado em cada um dos servidores gerenciados. Com isso, o sistema proposto realiza uma gestão que centraliza e todos os servidores do *data center* são geridos através de uma plataforma comum.

Como as funcionalidades de ambos os sistemas são semelhantes, a realização de escolha entre o uso do sistema deste trabalho e o Pi-KVM deve levar em consideração o número de servidores a serem gerenciados e o custo total de implementação.

2.2 Soluções Proprietárias

Existem dois tipos de soluções proprietárias para a gestão remota de servidores. Inicialmente podemos destacar as que se conectam com o servidor gerido por meio de sua interface PCI (*Peripheral Component Interconnect*), acessando diretamente os módulos internos da placa-mãe do servidor. Outro tipo são as que se conectam pelas interfaces externas (Saídas de vídeo, portas USB e etc...) do servidor. Essas são comercialmente conhecidas como *Switch KVM-Over-IP*, portanto, uma solução baseada em hardware para acesso remoto no qual a unidade se conecta às portas de teclado, vídeo e mouse de um computador ou servidor e as transmite para um usuário conectado através de uma rede [8].

2.2.1 MegaRAC Serie G, Peppercorn eRIC e Aten IP8000:

As soluções desta seção são conectadas com os servidores por meio da interface PCI e sua análise pode ser resumida em um mesmo tópico por compartilharem as mesmas características. No entanto, um aspecto relevante sobre essas soluções é que todas estão descontinuadas pelos seus fabricantes, com isso, torna-se inviável sua utilização.

- **Objetivo dos produtos.** Tem por objetivo adicionar um chip BMC à placa-mãe do servidor na qual é instalada por meio de uma conexão na interface PCI.

- **Tipo de *hardware*.** É composta por hardware proprietário em forma de uma placa de circuito que contém os componentes necessários para a gestão do servidor e habilitação das funções de acesso remoto.
- **Funcionalidades.** Por se comunicar com o servidor por meio de uma interface interna, a PCI, esses dispositivos têm conexão direta com os componentes internos do servidor. Tal característica adiciona vantagens no funcionamento do produto, como por exemplo, a obtenção dos sinais de vídeo independente da saída de vídeo padrão do servidor. Outra vantagem é o suporte a IPMI e a possibilidade de controlar as opções de energia sem necessitar de conexões extras na placa-mãe.

2.2.2 AdderLink IPEPS e DOMINION KX IV-101:

Nesta seção são apresentadas as soluções do tipo *KVM-Over-IP*, sendo essas as mais comumente encontradas no mercado.

- **Objetivo dos produtos.** Tem por objetivo adicionar aos servidores a capacidade de gestão remota utilizando as interfaces externas.
- **Tipo de *hardware*.** É composta por hardware proprietário, sendo comercializado como um módulo em uma caixa.
- **Funcionalidades.** Fornecem a capacidade de visualização da saída de vídeo, teclado e mouse remoto via rede. Sobre a capacidade de desligar e ligar os servidores:
  - Adder AdderLink IPEPS: Não há a capacidade de desligar e ligar o servidor, pois não tem comunicação com os comandos de energia.
  - Raritan DOMINION KX IV-101: Pode ligar e desligar os servidores remotamente.

2.3 Análise da pesquisa

É possível identificar, a partir da pesquisa deste capítulo, a existência de diferentes soluções para a gestão remota de servidores. Com isso, dentro do conjunto

de soluções de código aberto, pode-se distinguir projetos interessantes e relevantes em termos de funcionalidades. Todavia, o desenvolvimento deste trabalho ainda se mostra justificável devido ao aspecto inovador da solução, que tem como objetivo gerenciar um conjunto de servidores utilizando um sistema central, que resulta na simplificação da operação e diluição de custos de implementação em um *data center*.

É possível verificar nas Seções 2.1 e 2.2 que as soluções identificadas funcionam sempre de maneira não centralizada, isto é, com um módulo em cada um dos servidores. Este trabalho apresenta uma solução que centraliza o ponto de conexão e gestão em um único servidor, tal característica reduz os custos de implementação e simplifica a operação, pois o acesso a todos os servidores gerenciados é feito de um ponto comum.

Capítulo 3

Arquitetura e desenvolvimento do sistema

O projeto proposto neste trabalho requer uma série de módulos para seu funcionamento e implementação das funcionalidades. Este capítulo apresenta uma visão geral da arquitetura, detalhando também os elementos de *hardware* e *software* que compõem o sistema.

3.1 Visão geral e topologia da solução

Esta seção apresenta uma introdução sobre a arquitetura geral do sistema e fornece detalhes sobre os módulos que o compõe. Dessa forma, torna-se possível compreender a função de cada um dos módulos e como eles interagem entre si, facilitando o entendimento das seções posteriores.

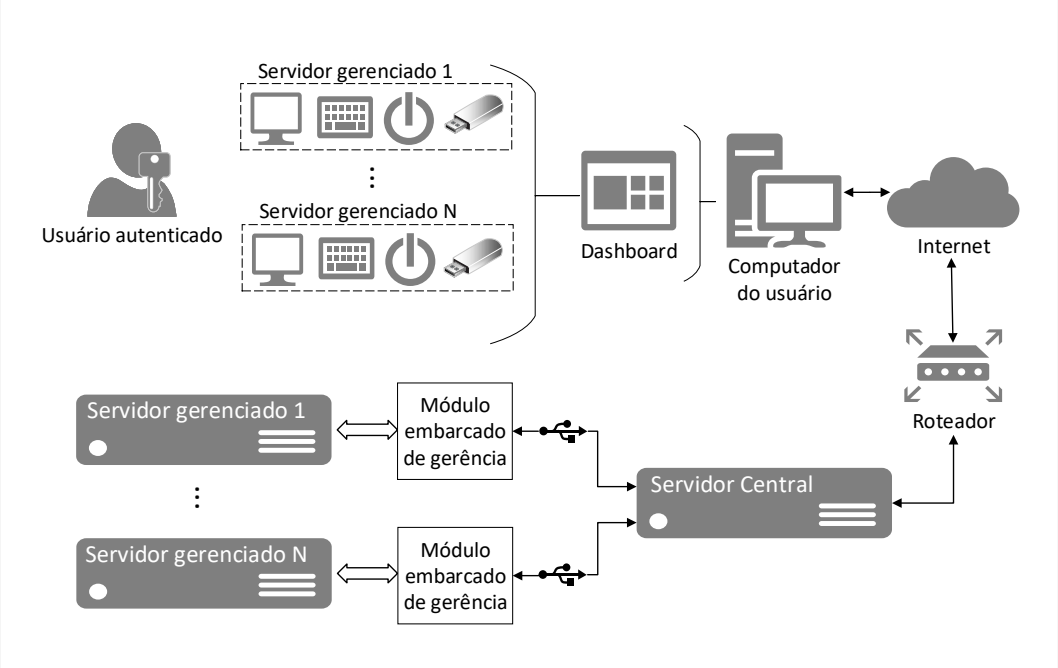


Figura 3.1: Visão geral da arquitetura do sistema.

A Figura 3.1 apresenta os módulos envolvidos em cada uma das etapas do processo de conexão entre o usuário e os servidores gerenciados.Através de um computador, utilizando um navegador, usuário pode se autenticar no sistema e ter acesso a um *dashboard*, no qual pode verificar informações sobre os servidores cadastrados e realizar todo processo de gerência remota. Há duas possibilidades para o acesso remoto ao servidor central, onde a primeira é o acesso via Internet. Para isso, o servidor central deve receber um endereço IP público visível na Internet.

Uma alternativa para manter o sistema mais seguro é a segunda possibilidade, onde o acesso ao servidor é feito somente por meio da rede interna da instituição que o utiliza. Nesse caso, é necessário que o administrador tenha à sua disposição o acesso a essa rede por meio de uma uma VPN *(Virtual Private Network)*.

Através desse sistema, o sistema passa a não ser exposto na internet com um IP público, o que dificulta tentativas de acessos não autorizadas e por consequência, não ocupa um IP público da instituição. A Figura 3.2 apresenta como é feita uma conexão ao sistema através de uma VPN. Através do do seu computador, o usuário do sistema pode se autenticar na VPN e ter acesso aos elementos da rede interna da instituição, podendo assim alcançar o servidor central e acessar a ferramenta.

O servidor central é o módulo no qual as principais aplicações são executadas.

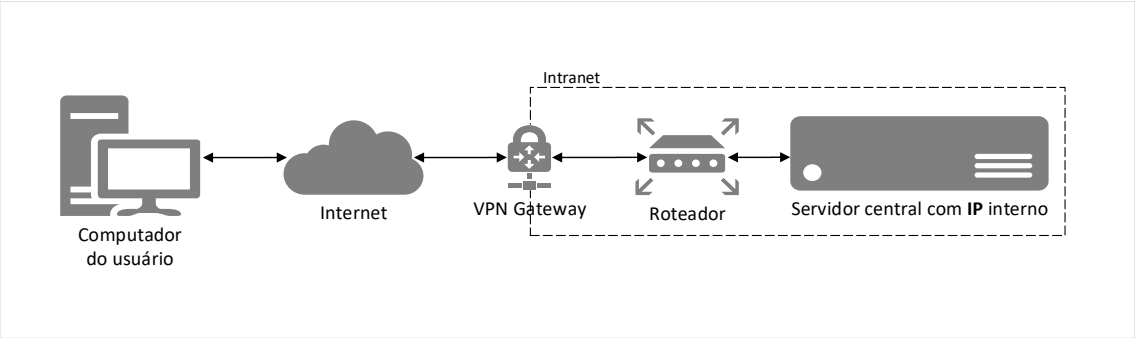


Figura 3.2: Acesso ao servidor gerenciado por VPN.

Para ter acesso ao *dashboard* da aplicação, o usuário deve estar autenticado. O sistema de autenticação utilizado no projeto é o Keycloack [9], uma solução de gerenciamento de identidade e acesso que possui seu código aberto. Essa ferramenta é explicada em detalhes na Seção 3.3.1.3.

Ainda no servidor central, existe o servidor web responsável por disponibilizar o *dashboard*. Outro serviço contido no servidor central é o *streamer* de vídeo, responsável por disponibilizar para o usuário a saída de vídeo do servidor gerenciado dentro da página web da aplicação. E, por fim, o servidor central é o responsável por se conectar com todos os módulos embarcados nos servidores gerenciados presentes no *data center*.

Os módulos embarcados de gerência são circuitos alocados nos servidores gerenciados que possuem componentes que viabilizam a gestão de tais servidores. A placa de desenvolvimento Arduino Pro Micro é responsável por receber os comandos do servidor central e executar-los nos servidores gerenciados. A conexão de mídia externa, leitura de estado do servidor e execução de opções de energia, como ligar, desligar e reiniciar, são exemplos de comandos que podem ser executados pelo Arduino.

Escolheu-se o Arduino Pro Micro pois seu microcontrolador, o *ATMega32U4*, tem uma interface USB que pode funcionar como um dispositivo HID *(human interface device)*, emulando um mouse ou um teclado. A utilização de um dispositivo com suporte a HID é mandatória para a implementação da funcionalidade de teclado remoto.

3.2 Arquitetura do *hardware*

Conforme descrito no Seção 3.1, o funcionamento do sistema depende de elementos de *hardware* para a hospedagem da aplicação e também para a atuação dentro de cada um dos servidores gerenciados. Esta seção tem o objetivo de discutir em detalhes os elementos físicos do sistema. A ênfase desta seção está no servidor central, por ser o módulo do sistema responsável por executar as principais aplicações e ser o componente central da topologia. E também no módulo embarcado de gerência, por ser a parte responsável por atuar no servidor gerenciado.

3.2.1 Servidor central

O servidor central é o núcleo do sistema, orquestrando todas as interações entre os módulos e os serviços necessários. A determinação do *hardware* necessário para o servidor central se fez inicialmente de maneira empírica, partindo de um *hardware* que atendesse às especificações mínimas para o uso do sistema operacional escolhido. A determinação do sistema operacional e seus requisitos mínimos são discutidos em detalhes na Seção 3.3.1.1.

Os principais aspectos observados foram o uso de CPU (*Central Process Unit*) e ocupação de memória RAM (*Random Access Memory*), conforme o número de clientes na aplicação crescia. Os resultados desse teste comprovaram que mesmo em sua configuração básica de *hardware*, o servidor central não apresentou redução na qualidade da experiência do usuário na aplicação web. O detalhamento do teste de ocupação de memória RAM e CPU se encontra na Seção 5.2.

Entretanto, a especificação dos controladores HCI (*Host Controller Interface*) se mostrou mais relevante do que os parâmetros inicialmente considerados (uso de CPU e ocupação de memória RAM) no processo de especificação. Os detalhes sobre essa especificação são apresentados a seguir.

Para que o sistema possa cumprir com os objetivos estabelecidos, os módulos de *hardware* utilizados no desenvolvimento são de fácil acesso e utilizados para fins diversos. Com isso, vários desses módulos funcionam através de uma interface USB, o que faz com que o servidor central necessite de diversas portas disponíveis para o controle dos servidores do *data center*.

A Figura 3.3 apresenta as conexões entre o servidor central e o módulo embarcado de gerência. Esse módulo ocupa duas portas USB do servidor gerenciado, sendo uma dedicada ao Arduino que emula o teclado remoto, e a outra dedicada à conexão do componente de mídia externa, sendo responsável por fazer a conexão remota de um pendrive. No servidor central, o módulo ocupa três portas USB, sendo uma dedicada à captura de vídeo, conexão com o conversor serial do Arduino e o componente de mídia externa. Detalhes do funcionamento de cada um dos componentes internos do módulo embarcado de gerência são apresentados na Seção 3.2.2.

Com isso, sempre que o número de servidores cresce, expandir o número de portas USB através de *HUBs* ou módulos USB PCI se faz necessário.

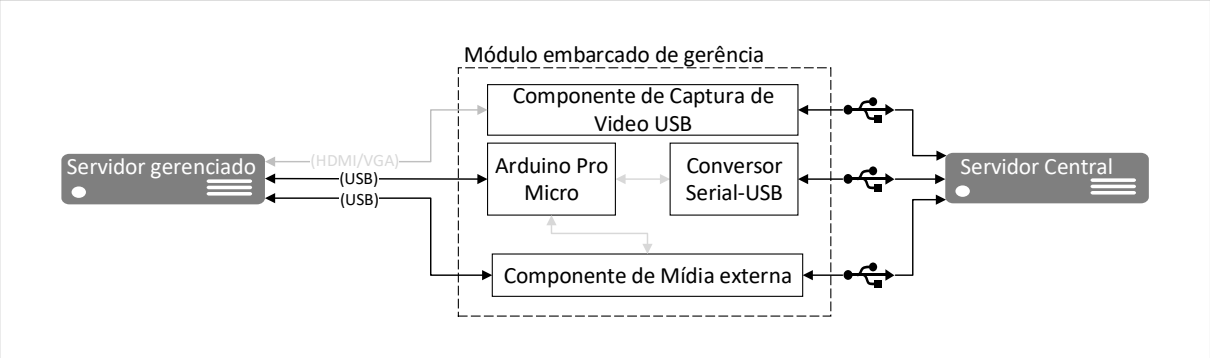


Figura 3.3: Módulo de gerência e número de portas USBs utilizadas.

Dado o exposto, o número de portas USB do sistema é um limitante importante e que deve ser contornado para que o sistema possa ser expandido ao ponto de poder gerenciar todos os servidores do *data center*. Porém, um limitante implícito que tem relação com o número de portas USB, é a quantidade de controladores USB disponíveis.

Todo computador atual possui um conjunto de componentes chamados de *Host Controller Interface*, de diferentes tipos, que são responsáveis por fazer a interface e o controle das portas USB com a CPU e com os demais componentes. O módulo HCI viabiliza a comunicação do computador com dispositivos externos via USB.

A Figura 3.4 exemplifica como os módulos de uma pilha USB estão dispostos. A aplicação que deseja acessar um dispositivo conectado em uma das portas USB utiliza recursos do sistema operacional (*drivers* e implementações) que, por sua

vez, se comunicam com os controladores HCI. Os controladores HCI acessam os dispositivos conectados às portas USB e coordenam a conexão.

Analisar a disponibilidade dos controladores no servidor central é de extrema importância, pois cada controlador tem um limite de banda para comportar dispositivos USBs. Com isso, o número de dispositivos conectados ao servidor central fica proporcionalmente limitado ao número de controladores, já que um controlador pode gerir múltiplas portas USBs.

Os controladores são classificados em diferentes tipos, dependendo de fatores como tecnologia utilizada e capacidade de banda disponibilizada. Mais detalhes sobre os tipos de controladores e suas bandas são apresentados a seguir. Deve-se atentar a uma particularidade em relação aos dispositivos de captura de vídeo USB: Como esses dispositivos trafegam um grande volume de dados, controladores mais simples e com menor banda disponível não são capazes de intermediar suas conexões. Sendo assim, as placas de captura ficam limitadas a serem direcionadas a controladores que possuem maior capacidade em termos de largura de banda.

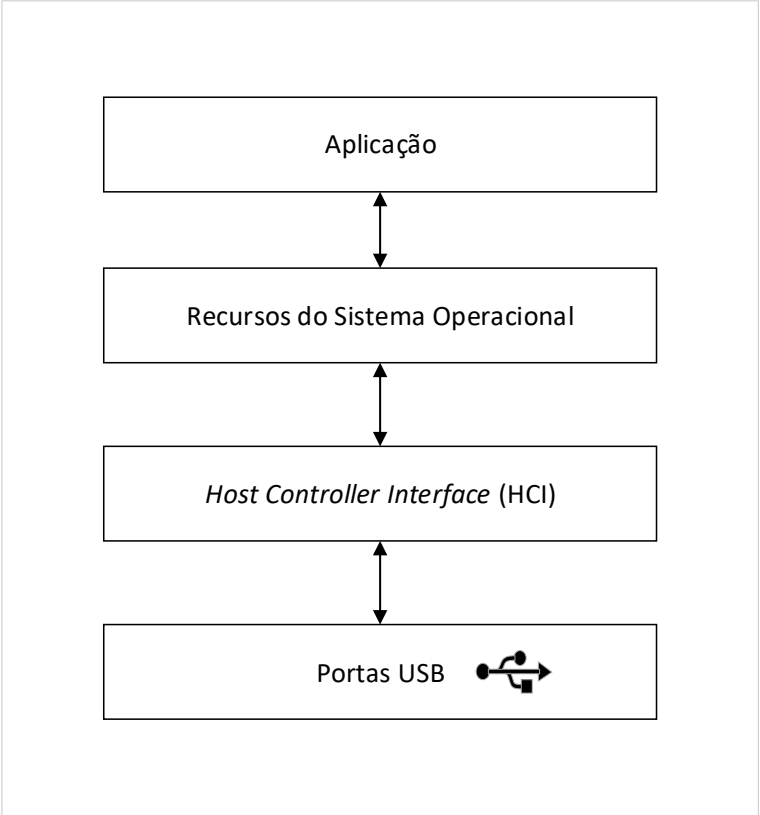


Figura 3.4: *Stack* USB (baseado em [1]).

Os tópicos seguintes descrevem as características básicas de cada um dos atu-

ais controladores USB, com o objetivo de compreender melhor o seu funcionamento e fornecer embasamento para a justificativa de uma das limitações do sistema. As descrições a seguir foram baseadas em [10]:

- **OHCI *(Open Host Controller Interface)*:** é um padrão que suporta USB 1.1. Além do USB, outras interfaces também são suportadas, como a interface Firewire da Apple. Esse controlador possui banda máxima de **12 Mbps**.

- **UHCI *(Universal Host Controller Interface)*:** é um padrão de interface para USB 1.0 e 1.1 que possui um *hardware* mais barato e mais simples e é uma interface proprietária criada pela Intel. Esse controlador possui banda máxima de **12 Mbps**.

- **EHCI *(Enhanced Host Controller Interface)*:** é o padrão de interface do USB 2.0, desenvolvido pela Intel. O EHCI fornece apenas funções de alta velocidade, suportando uma banda de até **480 Mbps** para USB 2.0, enquanto depende de controladores UHCI ou OHCI para fornecer suporte a dispositivos de baixa velocidade.

- **xHCI *(eXtensible Host Controller Interface)*:** é o padrão de interface USB 3.0 mais recente. Ele tem melhorias significativas em termos de velocidade e economia de energia em comparação com os três anteriores. O xHCI suporta todos os tipos de dispositivos USB (USB 3.0, USB 2.0 e USB 1.1). O objetivo do xHCI é substituir os três anteriores (isto é, UHCI, OHCI e EHCI). Sua banda máxima, na versão mais básica do controlador, pode chegar a **5 Gbit/s**.

Nos testes, foi observado que cada controlador EHCI só comporta uma placa de captura de vídeo USB por vez. Com isso, o número de servidores gerenciados distintos que podem ser observados pela aplicação web se limitam ao número de controladores EHCI disponíveis no servidor central. Mais detalhes sobre os testes de capacidade dos controladores USBs estão disponíveis na Seção 5.1. Não foi possível realizar os testes de funcionamento do sistema em um servidor com controlador xHCI para se obter a capacidade de captura de video por USB nesse controlador.

3.2.2 Módulo embarcado de gerência

O módulo embarcado de gerência é o conjunto de componentes que fazem a interface entre o servidor gerenciado e o servidor central.Esse módulo é responsável por permitir que o servidor central faça a gerência remota. A Figura 3.5 apresenta visão expandida dos componentes internos. Essa figura representa a conexão de um servidor gerenciado ao servidor central através do módulo embarcado de gerência, no qual o componente de captura de vídeo, através da interface HDMI (*High-Definition Multimedia Interface*), obtém a saída de vídeo do servidor gerenciado e a transmite ao servidor central por uma porta USB.

Tem-se ainda o Arduino Pro Micro que emula o teclado remoto no servidor gerenciado, controla as opções de energia e ainda controla o componente de mídia externa. O Arduino se comunica com o servidor central através de um conversor Serial-USB. Esse conversor é necessário pois a porta de comunicação USB do Arduino é conectada ao servidor gerenciado para a emulação do teclado, com isso, o módulo embarcado de gerência necessita de um outra interface para se comunicar com o servidor central. Com isso, utiliza-se a interface serial do Arduino conectada ao servidor central através do conversor Serial-USB.

O componente de mídia externa é uma mídia USB (Pendrive) que fica permanentemente conectada ao servidor gerenciado. Esse pendrive tem o sistema operacional do servidor gerenciado gravado, e é utilizado para uma reinstalação do sistema operacional do servidor gerenciado, caso seja necessário. Mais detalhes sobre o componente de mídia externa são tratados na Seção 3.2.2.7.

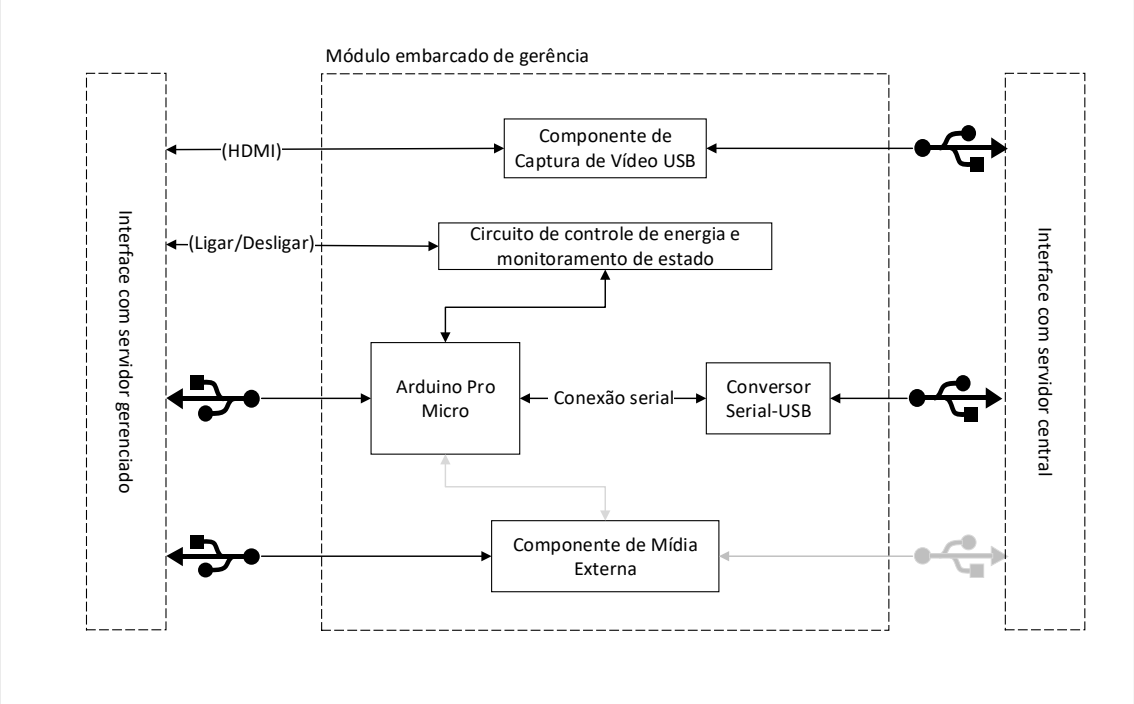


Figura 3.5: Visão interna do módulo embarcado de gerência.

3.2.2.1 Espaço interno nos servidores gerenciados

Inicialmente, é necessário considerar que, para que a instalação do sistema seja viável, os servidores que serão gerenciados devem possuir espaço interno suficiente para que o módulo embarcado de gerência seja instalado. Durante o desenvolvimento do protótipo, como pode-se observar na Figura 3.6, o servidor gerenciado utilizado possuía um gabinete modelo ATX (*Advanced Technology eXtended*), que é uma especificação de tamanho de componentes internos de um computador, que dispunha de espaço interno necessário. Todavia, um gabinete no padrão ATX não é mandatório, o único requisito é que haja espaço para a acomodação do módulo embarcado de gerência.

Outro modelo de gabinete que é capaz de comportar o sistema sem apresentar problemas de espaço físico são os chamados gabinetes para *rack*, de um ou mais *Us* [1], pois esses possuem o espaço necessário para a instalação dos módulos embarcados do sistema.

[1] O *U* significa "Rack Unit" e é a unidade de medida utilizada para descrever a altura de servidores. Cada "Rack Unit" equivale a 1,75 polegadas.

Figura 3.6: Gabinete sem tampa lateral do protótipo do servidor gerenciado com módulo embarcado de gerência (em destaque).

3.2.2.2 Solução inicial para o componente de captura de vídeo

Durante as pesquisas iniciais de viabilidade do projeto, o primeiro fator analisado foi tornar a captura de vídeo confiável e independente do sistema do sistema operacional do servidor gerenciado. A independência do sistema operacional é necessária, pois o projeto prevê o acessos ao servidor gerenciado em estados que o sistema operacional não está em execução como, por exemplo, durante o acesso à BIOS, para modificação de configurações.

A tentativa inicial deste projeto para a captura de vídeo envolvia a utilização de um Raspberry Pi, por meio da sua entrada de câmera embarcada e um conversor de HDMI para CSI-2, que é baseado no chip *Toshiba TC358840* [11]. Os módulos e suas conexões estão exemplificados na Figura 3.7.

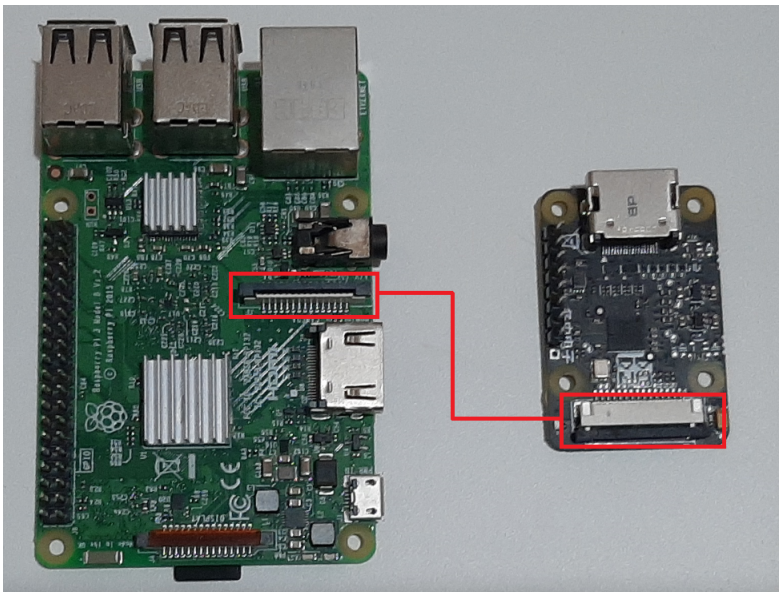


Figura 3.7: Raspberry Pi e módulo conversor HDMI para CSI-2.

Com o uso do Raspberry Pi, a captura de vídeo poderia ser implementada e seria possível utilizar os pinos GPIO *(General Purpose Input/Output)* na implementação de outras funcionalidades, como o controle de ligar e desligar o servidor. Além disso, com o uso de uma plataforma tão completa como o Raspberry Pi, cada servidor poderia ter sua interface web de gerência independente e o uso de um servidor central poderia ser considerado opcional. Porém, durante os testes, alguns pontos acabaram inviabilizando esta opção.

O primeiro resultado que inviabilizou o uso do conjunto supracitado foi a confiabilidade na captura de imagens. O processo parou de responder algumas vezes durante a captura. Tal comportamento se deve, provavelmente, à limitações na capacidade de processamento do Raspberry Pi.

Outro problema foi a necessidade de o componente de captura ser reinicializado fisicamente por um pino em sua PCB (*Printed circuit board*) sempre que o servidor que estava enviando o sinal de vídeo era reinicializado. Esse comportamento insere uma instabilidade no sistema que deveria ser tratada via *software*, o que aumenta a complexidade do desenvolvimento e reduz a confiabilidade da aplicação.

A integração do componente de conversão HDMI para CSI-2 com o *kernel* do sistema operacional do Raspberry Pi se mostrou pouco madura, com muitos pontos de falha em aberto, e pouco confiável para um uso em grande escala em um sistema em produção.

Por fim, outra característica importante para a decisão de descarte desta solução foi a dificuldade de aquisição desse componente de captura HDMI - CSI-2. Não foram encontrados fornecedores nacionais, portanto o exemplar obtido para os testes teve de ser importado.

3.2.2.3 Solução definitiva para o componente de captura de vídeo

Tendo em vistas todos os possíveis problemas que poderiam ser enfrentados, baseados nos testes realizados, o projeto foi repensado e uma nova solução proposta. Nessa nova solução, não são utilizados os Raspberry Pi em cada um dos servidores gerenciados. Nessa proposta, foi inserida a figura do servidor central, que facilita a implementação e habilita a possibilidade do desenvolvimento de uma solução centralizada.

Para a captura de vídeo, o módulo baseado no chip *Toshiba* foi substituído por uma placa de captura HDMI-USB, que tem baixo preço de aquisição, alta disponibilidade no mercado (local e de importados) e alta confiabilidade, demonstrando um desempenho adequado e atendendo a todas as necessidades durante os testes. As Figuras 3.8 e 3.9 apresentam os dois modelos adquiridos para os testes do projeto.

O problema antes observado com o módulo *Toshiba* de captura, que ocorria quando o servidor gerenciado era reiniciado, não ocorre com o uso da placa de captura HDMI-USB.

Figura 3.8: Placa de captura HDMI-USB.

Figura 3.9: Segunda placa de captura HDMI-USB adquirida.

3.2.2.4 Arduino Pro Micro

O Arduino Pro Micro é uma placa para protótipos baseada no micro-controlador ATmega32U4. Esse componente é utilizado no módulo embarcado de gerência como um atuador local que executa tarefas recebidas do servidor central.

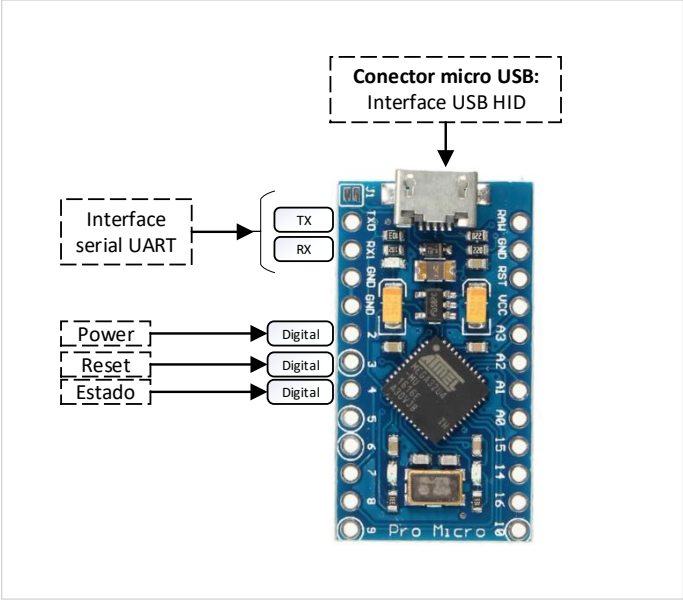


Figura 3.10: Arduino Pro Micro e interfaces utilizadas.

A Figura 3.10 representa o Arduino com a indicação dos pinos e funcionalidades que foram utilizadas no projeto, detalhados a seguir.

- **Conector micro USB:** Através do conector micro USB na placa do Arduino Pro Micro, é possível ter acesso à interface USB integrada do microcontrola-

dor ATmega32u4, o que elimina a necessidade de um processador secundário. Além disso, o conector permite que Arduino tenha a capacidade de emular dispositivos HID, como mouse e teclado e de estabelecer uma conexão serial/COM virtual [12].

- **Interface serial:** Os pinos *Rx e Tx* do Arduino proveem acesso a uma porta serial secundária do microcontrolador. Esta interface é utilizada para a comunicação do módulo embarcado de gerência com o servidor central, através de um conversor de Serial para USB, apresentado na Figura 3.11.

- **Power:** O pino que tem a etiqueta *Power* na figura se trata de uma saída digital do Arduino, utilizada para fazer o controle dos ciclos de energia (comandos de ligar e desligar) do servidor gerenciado. Com isso, por meio desse pino e de um pequeno circuito com um transistor operando como chave eletrônica, o servidor central é capaz de ligar e desligar servidor gerenciado. Os detalhes sobre esse circuito estão descritos na Seção 3.2.2.6.

- **Reset:** Assim como o pino supracitado, o *Reset* é o pino digital do Arduino responsável pela operação de reiniciar o servidor gerenciado.

3.2.2.5 Conversor Serial-USB TTL PL2303

O Conversor Serial-USB TTL PL2303 é baseado no chip da Prolific PL2303HX, que é uma solução em chip único utilizada para converter sinais RS232 de dispositivos para uma interface USB. A versão utilizada no projeto se trata de um módulo com chip, porta USB, pinos de comunicação e todos os periféricos adicionais necessários para o funcionamento do circuito.

Esse módulo é necessário para habilitar a conexão entre o Arduino Pro Micro e o servidor central, pois a porta USB do Arduino é utilizada no servidor gerenciado para emular o teclado remoto.

O módulo utilizado tem 5 pinos de comunicação e uma porta USB. Para a aplicação deste trabalho, a porta USB é conectada ao servidor central, que enxerga o módulo como uma porta serial e pode enviar comandos ao Arduino. Os pinos de comunicação são conectados conforme mostrado na Tabela 3.1.

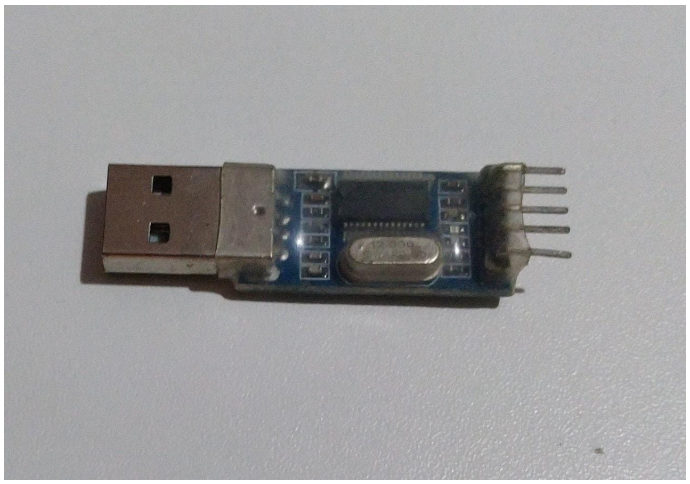


Figura 3.11: Conversor Serial-USB.

Conversor	Arduino
+3.3V	-
TX	RX
RX	TX
GND	GND
+5V	+5V

Tabela 3.1: Matriz de conexão entre o conversor Serial-USB e o Arduino.

3.2.2.6 Circuito de controle de energia e monitoramento de estado dos servidores gerenciados

As placas-mãe dos computadores possuem um conjunto de pinos que são comumente identificados como "conector do painel frontal" ou "conector do painel do sistema". Esse conjunto de pinos é responsável por enviar os sinais de controle que ligam, desligam e reiniciam o computador. Ademais, há ainda pinos de sinais que possuem informações sobre o estado do computador e sobre a leitura do disco rígido. Há um exemplo desse conector na Figura 3.12.

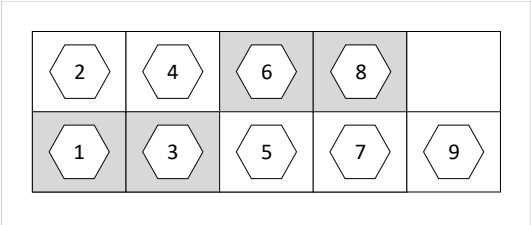


Figura 3.12: Conector do painel frontal de uma placa-mãe.

- Os pinos **1** e **3** são identificados como **HD_LED** e são destinados à conexão da luz de atividade do LED (*light-emitting diode*) do disco rígido. Esse indicador é a luz que pisca conforme as informações são gravadas e lidas no disco rígido;

- Os pinos **2** e **4** são identificados como **P_LED** e são direcionados para a conexão do LED de energia, um sinal que indica quando o computador está ligado, desligado ou em espera;

- Os pinos **6** e **8** são identificados como **PWR_SW** e se conectam ao botão de energia do gabinete *(Power SW)*. Esses pinos são responsáveis pelas funções quem permitem ligar e desligar o computador. Somente um dos pinos é responsável pela função de ligar/desligar e o outro se trata de um GND, já que sua ativação se dá em nível lógico baixo, o que permite a conexão de um botão entre os pinos 6 e 8 que liga ou desliga o computador;

- Os pinos **5** e **7** são identificados como **RST_SW** e são direcionados para a o botão de reinicialização do computador. Assim como no caso do botão de energia, o botão de reinício também é ativado por um nível lógico baixo (GND).

- O pino **9** não é utilizado.

A posição dos pinos pode variar de acordo com o fabricante ou modelo da placa-mãe utilizada. Antes de fazer as conexões, os pinos devem ser identificados pelas inscrições na PCB ou por meio de testes e medições.

Os sinais do conector do painel frontal do gabinete são ativados em nível lógico baixo (GND). Assim, o módulo embarcado de gerência tem conexões diretas a esses pinos. Além disso, por meio de um circuito de chave eletrônica com um transistor bipolar NPN, o Arduino pode executar o acionamento desses botões. A

Figura 3.13 apresenta o circuito simplificado de acionamento dos botões de energia pelo Arduino.

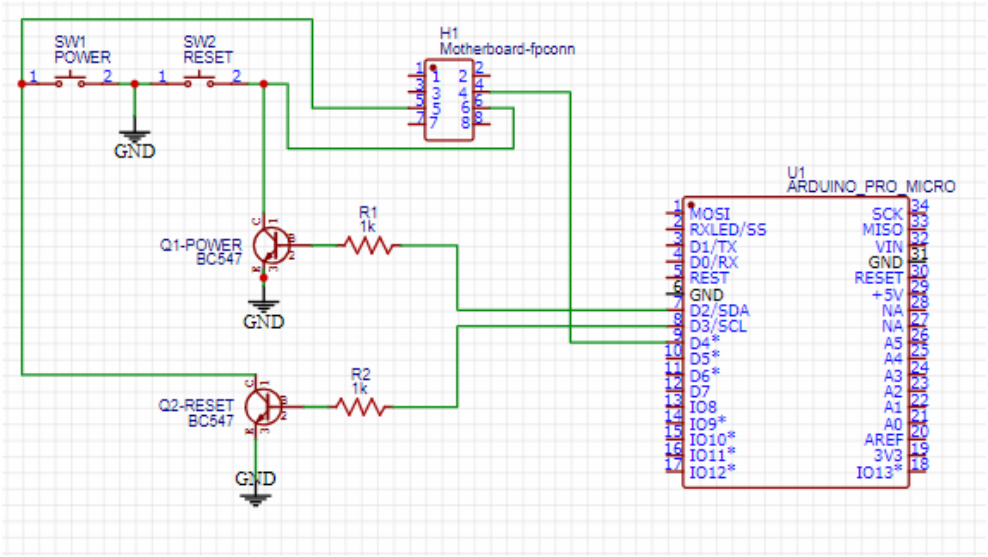


Figura 3.13: Circuito da conexão do Arduino com pinos de PWR_SW e RST_SW.

O circuito apresentado na Figura 3.13 não contempla todos os módulos da placa de circuito do módulo embarcado de gerência, pois foi simplificado para mostrar somente os componentes envolvidos no mecanismo de ligar e desligar e algumas outras funcionalidades do servidor gerenciado que serão detalhadas a seguir.

Os pinos número 5 e 6 do componente H1 são respectivamente os pinos PWR_SW e RST_SW do conector do painel frontal. Já os $SW1$ e $SW2$ representam os botões de ligar e reiniciar que existem no gabinete do servidor gerenciado. Com isso, vemos que a integração do controle de energia não afeta o funcionamento normal dos botões de energia já existentes no gabinete dos servidores gerenciados.

Outra conexão que observa-se no circuito é a do pino 9 (Estado, na Figura 3.10) do Arduino ao pino 4 conector do painel frontal da placa-mãe, representado pelo componente $H1$. Como detalhado na Seção 3.2.2.4, esse pino foi reservado como uma entrada digital do Arduino para fazer a leitura do estado do servidor. Quando o servidor está ligado, o pino 4 do conector da placa-mãe mantém sua tensão em $5V$, e quando desligado esse valor cai para próximo de $0V$. Com isso, o módulo embarcado de gerência pode consultar o valor de tensão e ter o estado atualizado de funcionamento do servidor, sempre que necessário.

3.2.2.7 Componente de mídia externa

A proposta presente na arquitetura do módulo embarcado de gerência, presente na Figura 3.5, é de que o componente de mídia externa pudesse funcionar como uma ponte para a transferência de dados entre o servidor central e o servidor gerenciado. Através do sistema de administração, usuário poderia conectar o pendrive presente no componente de mídia externa ao servidor central, enviar arquivos ou gravar sistemas operacionais e, de maneira remota e pela interface, pudesse comutar a conexão e conectar o pendrive ao servidor gerenciado.

Tal proposta depende de que haja um chip ou circuito capaz de comudar a conexão USB entre os servidores. Como Não foi possível encontrar um elemento comutador que atendesse aos requisitos, decidiu-se que o componente de mídia externa fosse um pendrive pré-gravado com o sistema operacional do servidor gerenciado e que estivesse sempre conectado ao mesmo. Dessa forma, pode-se realizar uma reinstalação do sistema operacional, caso necessário.

Por isso, a Figura 3.5 apresenta as conexões entre o componente de mídia externa, Arduíno Pro Micro e servidor central em uma cor mais clara. Isso representa que as conexões foram pensadas para funcionar como descrito acima, porém a sua implementação não foi possível. Com isso, tal implementação é endereçada a uma evolução futura do projeto.

3.3 Arquitetura do *software*

Os códigos necessários para o desenvolvimento do sistema são executados por dois dos módulos de *hardware* citados nas seções anteriores. O módulo principal, responsável por concentrar a maior parte da lógica e regras do sistema, é o servidor central. Esse módulo possui as bases de dados com cadastros de servidores do *data center*, um sistema de autenticação, um servidor web da aplicação e os demais módulos necessários para o funcionamento da aplicação, que são detalhados nas seções subsequentes.

O segundo módulo que executa um código desenvolvido neste projeto é o microcontrolador presente no Arduino Pro Micro do módulo embarcado de gerência. Esse módulo é responsável por se comunicar com o servidor central, recebendo e

enviando comandos. Além disso, o módulo funciona como um atuador que executa ações de energia no servidor em que está instalado.

Descrevem-se a seguir os detalhes do *software* do servidor central e do módulo embarcado de gerência. Todo os *software* mencionados neste capítulo e desenvolvidos para este trabalho podem ser encontrados no GitHub [13].

3.3.1 *Software* do servidor central

A Figura 3.14 apresenta um diagrama de blocos de alto nível dos componentes de *software* que são executados no servidor central e como eles interagem entre si.

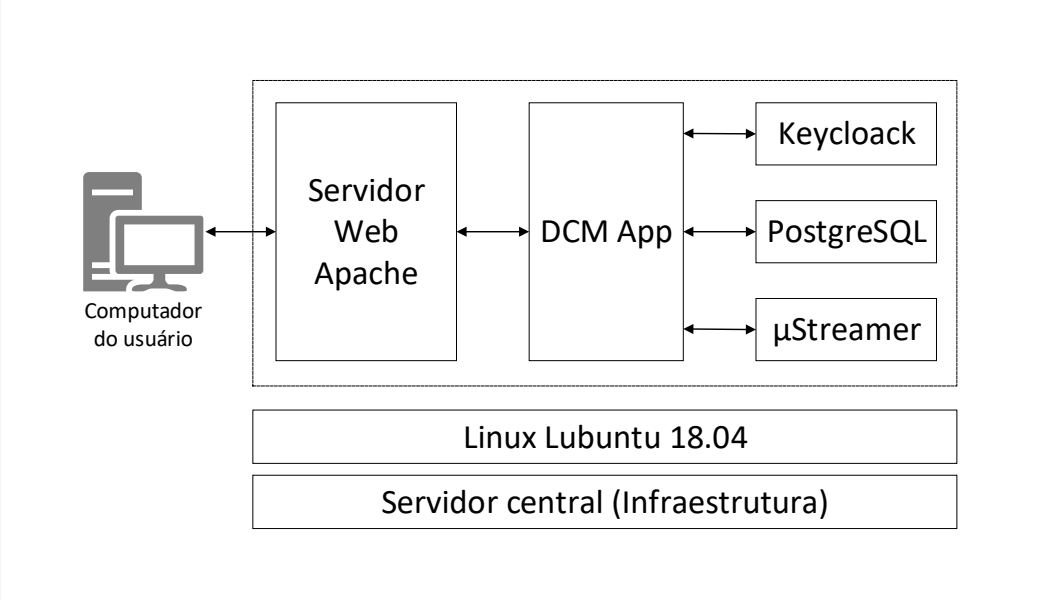


Figura 3.14: Representação simplificada do *hardware*, sistema operacional e *software* do servidor central.

A camada mais inferior da pilha mostrada na Figura 3.14 representa o *hardware* do servidor central, que é o componente principal da infraestrutura do sistema e foi descrito em detalhes na Seção 3.2.1. As demais camadas e módulos são explicados em detalhes nas seções seguintes.

3.3.1.1 Linux Lubuntu 18.04

A camada intermediária representa o sistema operacional para o qual a solução foi desenvolvida, o *Lubuntu 18.04* [14]. A versão do Lubuntu utilizada é de suporte de longo prazo e baseada na distribuição Ubuntu.

O Lubuntu trata-se de um sistema operacional leve e com poucas aplicações pré-carregadas. Sua escolha se deu por se tratar de um sistema operacional que tem vasto suporte a aplicações e baixos requisitos de *hardware*. A Tabela 3.2 apresenta os requisitos de memória RAM e CPU mínimos para o funcionamento do sistema operacional.

Além disso, o sistema possui uma interface gráfica mínima que facilitou o desenvolvimento,os testes e a resolução de problemas por meio do uso de aplicações diretamente instaladas no servidor central.

Requisito	**Desktop**
Memória *RAM*	2GB
CPU	Pentium 4

Tabela 3.2: Requisitos mínimos de *hardware* do Linux Lubuntu 18.04.

Como o servidor central tem a proposta de ser um *hardware* dedicado à gestão dos demais servidores e seu poder computacional não faz parte do total que compõe o *data center*, seu aproveitamento acaba sendo limitado à execução das aplicações do sistema de gestão. Com isso, fazer com que a sua capacidade seja ajustada a somente o necessário pela aplicação se torna um aspecto desejado. O uso de um sistema operacional com as características do Lubuntu se torna uma decisão de projeto estratégica e visa atender a esse requisito.

3.3.1.2 µStreamer

Para que seja possível visualizar o vídeo obtido pelo componente de captura USB na aplicação web do projeto, é necessário o uso de um *software* que seja capaz de transmitir esse vídeo em um servidor HTTP (*Hypertext Transfer Protocol*).

O µStreamer [15] é um servidor leve e muito rápido para transmitir vídeo em um formato MJPG (*Motion JPEG*), de dispositivos de captura USB para a rede. Esse formato de transmissão de vídeo MJPEG é um formato de compactação de vídeo no qual cada quadro de vídeo é compactado separadamente como uma imagem JPEG.

Como visto na Figura 3.15, sempre que o usuário solicita a saída de vídeo

de um servidor gerenciado o DCM App (*Data Center Manager Application*) inicia o µStreamer. O uStreamer, por sua vez, se conecta ao componente de captura de vídeo, obtém o sinal de vídeo e o transmite para a aplicação web.

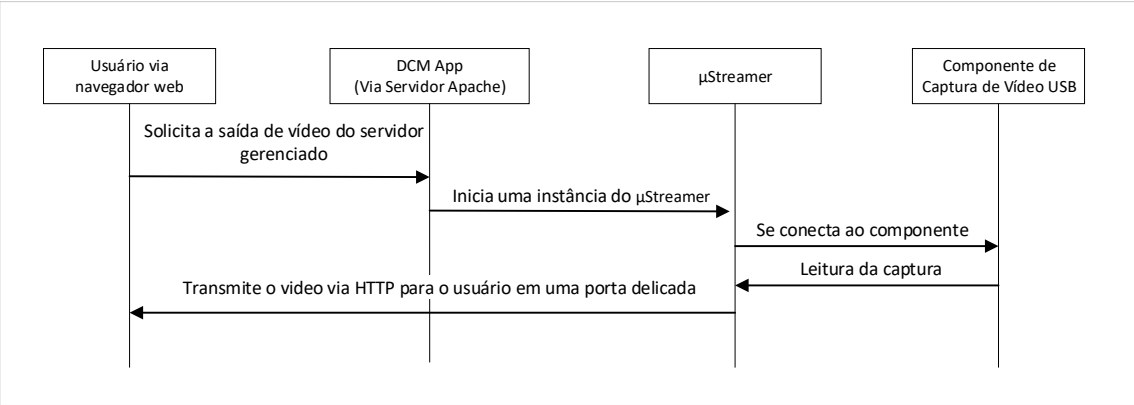


Figura 3.15: Fluxo de funcionamento do µStreamer.

3.3.1.3 Keycloack

A funcionalidade principal do sistema proposto por este trabalho é a gestão remota e monitoramento de um conjunto de servidores de um *data center*. A partir do momento em que o sistema proposto é implantado, todos os servidores podem responder a ações de maneira remota. Com isso, se torna obrigatório garantir que somente pessoas autorizadas a gerir esses servidores tenham acesso a essas funcionalidades.

Desenvolver sistemas seguros e confiáveis de autenticação de usuários e ferramentas web é uma tarefa complexa e que demanda muito conhecimento e esforço para que o produto final tenha desempenho satisfatório. Tendo em vista isso, uma decisão de projeto foi de utilizar um sistema autenticação e autorização de código aberto já desenvolvido com as melhores práticas e tecnologias existentes e integra-lo à aplicação deste projeto. Isso garante um produto final mais confiável, seguro e profissional.

O sistema de autenticação utilizado é Keycloack [9], que é um projeto suportado pela RedHat. Trata-se de um *software* código aberto feito para trabalhar em conjunto com aplicações em implementações mais comuns de autenticação e autorização. Se trata de um sistema completo e customizável, de fácil integração.

Neste projeto, o Keycloack foi instanciado como um contêiner no Docker o que

facilita o processo de instalação e configuração. Para tal, utilizou-se um Dockerfile, que é um arquivo que descreve os parâmetros de configuração e pré-requisitos para a execução da aplicação.

O Keycloack é integrado com o DCM App através da camada de identidade *OpenID Connect*. Essa ferramenta permite que aplicações se registrem como clientes em um servidor de autenticação e, verifique a identidade de um usuário e obtenham informações sobre seu perfil. Neste trabalho, o DCM App é registrado no Keycloack e pode verificar a identidade dos usuários cadastrados.

A Figura 3.16 mostra como o DCM App é criado como um cliente externo no Keycloak.

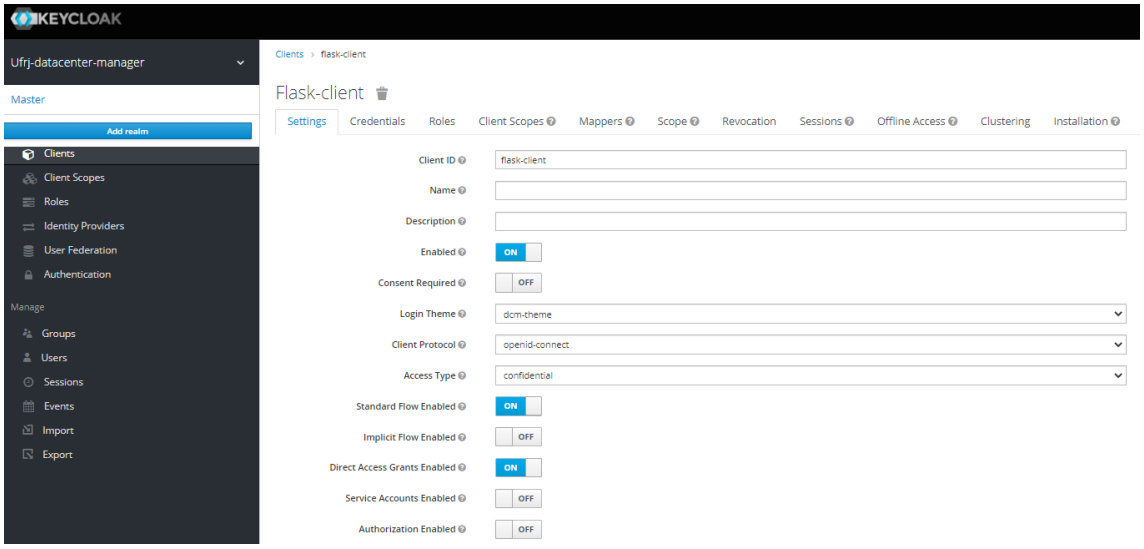


Figura 3.16: Criação do cliente da aplicação no Keycloak.

O processo de acesso à aplicação web com a autenticação feita pelo Keycloack está representado na Figura 3.17. Inicialmente, o usuário acessa a aplicação web e é redirecionado para a tela de *login* da aplicação (Figura 4.1). Uma vez na página de login, o usuário pode inserir suas credenciais. O usuário é validado e, em caso de sucesso, ele é redirecionado para a tela inicial da aplicação web (Figura 4.2).

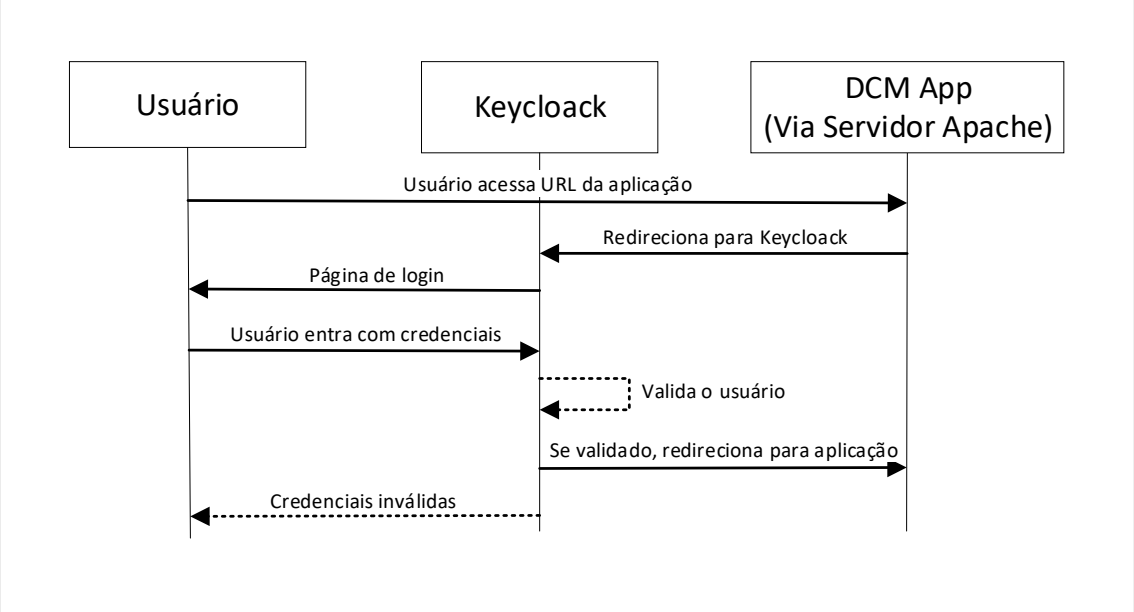


Figura 3.17: Autenticação da aplicação com Keycloack.

3.3.1.4 PostgreSQL:

O PostgreSQL é o banco de dados relacional que utiliza a linguagem SQL (Structured Query Language) , também utilizada pela aplicação, tanto para armazenar os seus dados e servidores cadastrados quanto para armazenar as informações dos usuários obtidas pelo Keycloack. Este banco de dados é originalmente instanciado pelo mesmo *Docker file* que cria o *Keycloack*. O *software* do sistema acessa o PostgreSQL através de uma biblioteca de mapeamento objeto-relacional SQL chamada *SQLAlchemy* e através dessa biblioteca, o sistema pode realizar manipulações no banco. A Seção 3.3.1.6 trata em detalhes o funcionamento desse sistema.

3.3.1.5 Servidor Web Apache

O servidor web Apache é o *software* responsável por prover as aplicações web, distribuindo as páginas HTML geradas pela aplicação para os usuários através do protocolo HTTP, que é utilizado pelos navegadores de internet. Para isso, o servidor web Apache tem que de interagir com o o DCM App através do WSGI *(Web Server Gateway Interface)*, e com isso executar os módulos internos do projeto. Detalhes sobre o funcionamento do WSGI são tratados na Seção 3.3.1.6.

3.3.1.6 DCM App *(Data Center Manager Application)*

O DCM App, da Figura 3.14, é a aplicação desenvolvida em Python com o uso de Flask [16], um *micro-framework* modularizado que facilita o desenvolvimento de aplicações web, como sites e APIs (*Application Programming Interface*).

Outra ferramenta importante que integra o Flask é o WerkZeug WSGI *(Web Server Gateway Interface)*, que é trata-se de uma forma para servidores conversarem com *frameworks*, e vice-versa. O WSGI é responsável pela comunicação entre a aplicação em Python e o servidor web Apache. Sem isso, o servidor web Apache não teria a capacidade de enviar aos usuários as páginas geradas pelo *back-end* da aplicação.

O DCM App é responsável por controlar, se comunicar com os módulos embarcados de gerência em cada um dos servidores e prover a aplicação web. Na Figura 3.18 tem-se uma visão geral da aplicação separada em dois grandes grupos de funções, o *front-end* e o *back-end*.

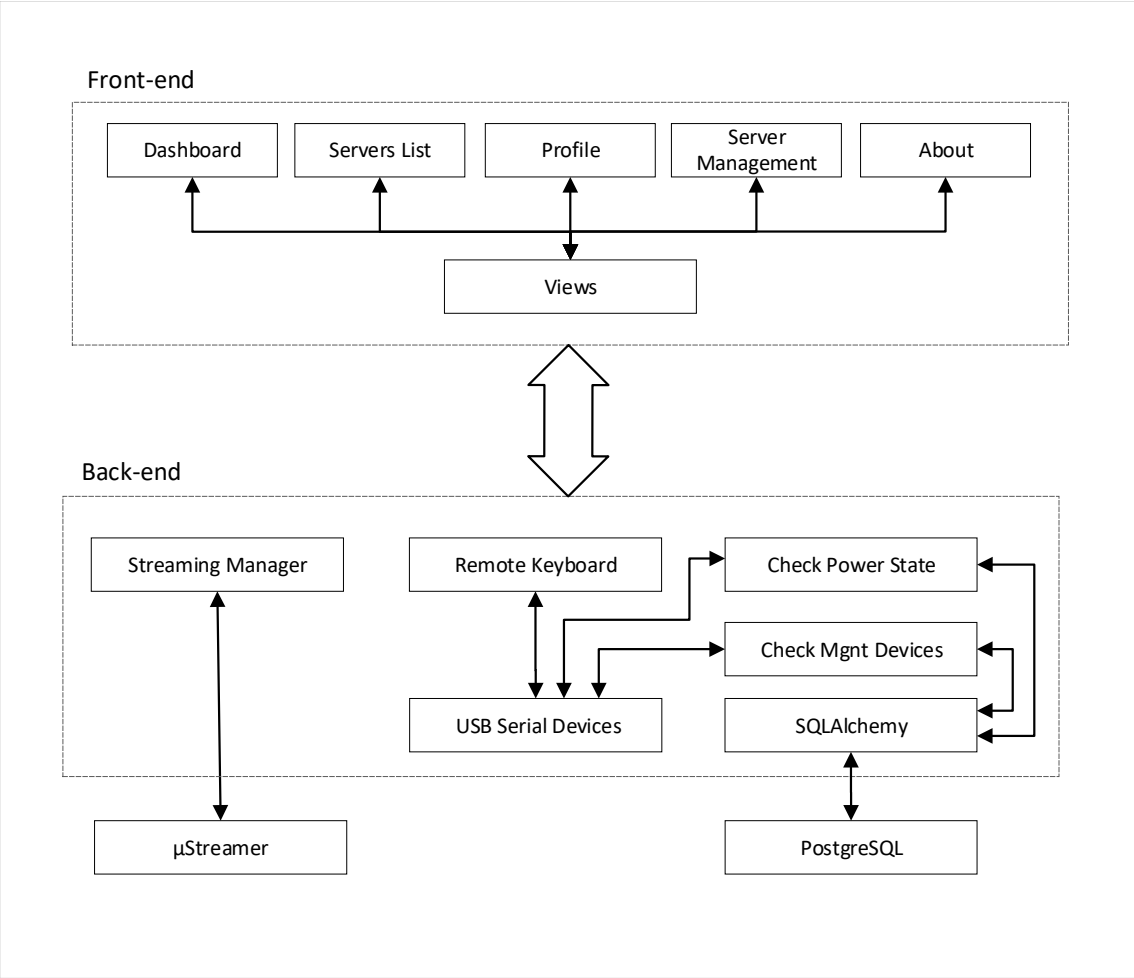


Figura 3.18: DCM App.

O DCM App concentra grande parte das lógicas do sistema, do controle de comunicação e informação. Os módulos mostrados na Figura 3.18 foram implementados por funções distintas, com o objetivo de segregar as funções de cada módulo, facilitando o desenvolvimento e resolução de problemas. A seguir detalha-se cada um dos módulos do sistema, suas atribuições e pontos relevantes no seu desenvolvimento.

O *front-end* da aplicação possui as telas de gerência. Seus componentes gráficos foram desenvolvidos com HTML, CSS, JavaScript e com o uso do *framework* web Bootstrap. Esse *framework* possui um conjunto de componentes de interface e *front-end* para aplicações web, que melhoram a experiência do usuário e facilitam o desenvolvimento de aplicações visualmente agradáveis e responsivas.

Com o uso de Flask para o desenvolvimento das telas do sistema, foi possível utilizar o recursos de *templates* disponibilizados pelo *micro-framework*. Atra-

vés desse mecanismo, todas as telas da aplicação web podem ser desenvolvidas em HTML e possuírem, nos campos que possuem informações variáveis, marcadores que serão substituídos pelas informações desejadas quando acessadas pelo usuário final. Na Figura 3.19, tem-se os *templates* criados para a aplicação web deste trabalho.

index.html	Documento HTML	4,5 KiB
list.html	Documento HTML	14,7 KiB
main-template.html	Documento HTML	6,0 KiB
profile.html	Documento HTML	5,6 KiB
server.html	Documento HTML	10,9 KiB
thirdparty.html	Documento HTML	7,5 KiB

Figura 3.19: *Templates* Flask para telas do sistema web.

Com isso, cada um dos módulos do *front-end* apresentados na Figura 3.18 são providos pelos *templates*. O Capítulo 4 apresenta e detalha as telas da aplicação web, além de fornecer informações relevantes sobre a implementação.

O *back-end* da aplicação é composto por módulos desenvolvidos em Python e apresentados na Figura 3.18. O tópicos a seguir detalham esses módulos.

- **Streaming Manager:**

  Esse é o módulo responsável por iniciar e parar as instâncias do µStreamer. Cada vez que a tela de console de um novo servidor gerenciado é acessada pela aplicação web, uma nova instância do µStreamer é iniciada. Com isso, é necessário que haja um módulo que gerencie essas instâncias e que as interrompa quando não são mais utilizadas, com objetivo de liberar recursos do servidor central.

  O µStreamer faz o *streaming* dos vídeos dos servidores gerenciados através de uma porta HTTP reservada a cada um. Essa porta é escolhida de acordo com a identificação da placa de captura no sistema. Assim, a porta de *streaming* escolhida é dada por 801**x**, onde **x** é o identificador do dispositivo de vídeo.

Para que o *Streaming Manager* possa interromper as instâncias do µStreamer não utilizadas, ele consulta o sistema de portas do computador para verificar se há clientes com conexão ativa nas portas alocadas para os *streamers*. Caso uma porta esteja livre, o módulo interrompe a execução da instância associada.

- **Remote Keyboard:**

Este é o módulo responsável por receber as teclas pressionadas na tela de console e redirecionar para o Arduino, que emula o teclado no módulo embarcado de gerência. Seu funcionamento se baseia na disponibilização de uma URL que recebe as teclas enviadas da aplicação web através de métodos HTTP POST.

A Figura 3.20 apresenta o funcionamento do módulo *Remote Keyboard*. A partir da interface de gerência do servidor, as teclas de controle são enviadas via HTTP *post* para uma URL do DCM App. Após as manipulações necessárias (mais detalhes sobre o sistema de códigos na Seção 3.3.2), a solicitação é enviada para o Arduino que, por sua vez, emula a tecla pressionada no servidor gerenciado.

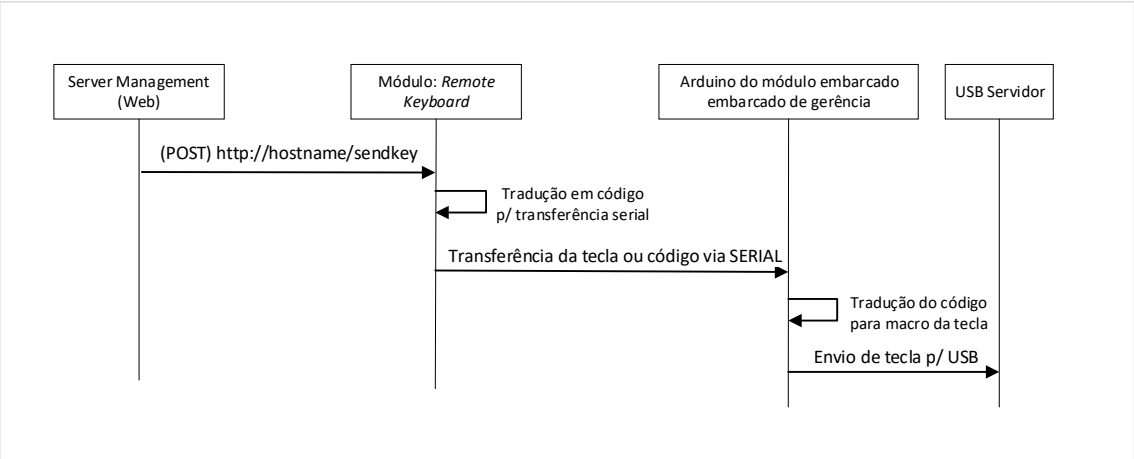


Figura 3.20: Fluxo de comunicação do teclado remoto.

- **Check Power State:**

Este módulo é responsável por verificar o estado de energia dos servidores remotos, ou seja, se estão ligados ou desligados. Essas informações são utili-

zadas para atualizar o banco de dados PostgreSQL da aplicação, para que se tenha sempre a informação mais atualizada sobre todos os servidores.

Assim que a aplicação é iniciada, uma *thread*, independente do fluxo principal, é iniciada com a execução deste módulo, e ele busca periodicamente o estado dos servidores junto ao módulo embarcado de gerência. Após obter esta informação, o módulo solicita a atualização da informação, se necessário, no banco de dados.

A Figura 3.21 apresenta o funcionamento do módulo *Check Power State*. A *thread* responsável por fazer a verificação solicita o dado periodicamente ao Arduino presente no módulo embarcado de gerência. Uma vez que tenha a informação atualizada, solicita atualização da informação no banco de dados através do SQLAlchemy.

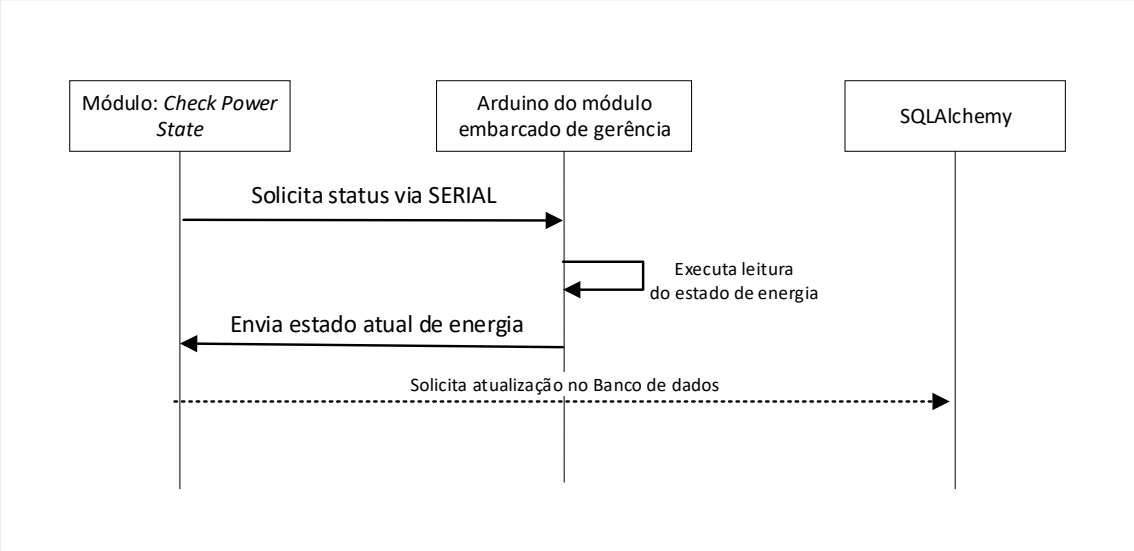


Figura 3.21: Fluxo de comunicação do módulo de verificação do estado de energia.

- **Check Mgnt Devices:**

Esse módulo é responsável por monitorar todos os módulos embarcados de gerência conectados ao servidor central e atualizar a lista de dispositivos disponíveis para conexão. Com isso, sempre que novos módulos são conectados, eles são disponibilizados na plataforma. Outra função desse módulo é garantir

que os módulos já cadastrados são sempre mapeados corretamente ao perfil dos seus respectivos servidores.

Isso é necessário pois a identificação dos dispositivos conectados através das interfaces USB do servidor central é feita de maneira sequencial conforme eles são reconhecidos. Ou seja, o primeiro componente de vídeo conectado ao servidor receberia a identificação video0, e assim sucessivamente. Entretanto, caso o servidor central seja reiniciados, os dispositivos USB podem ser atribuídos a identificações diferentes das iniciais, e por isso esse módulo verifica essas atualizações nas identificações e garante sempre a localização e identificação correta dos servidores gerenciados na aplicação.

- **SQLAlchemy:**

  Lidar com a conexão entre a aplicação e o banco de dados sem o uso de um intermediário não é uma boa prática pois isso requer um grande esforço de desenvolvimento para que se alcance um mínimo de qualidade. Assim, a opção mais eficiente e rápida de projeto foi utilizar a biblioteca *SQLAlchemy*, que é responsável por tratar da conexão com o banco de dados e mapear classes em Python às tabelas do banco de dados SQL. Com isso, toda a interação feita com o banco de dados é tratada pela biblioteca e a manipulação dos dados é feita por funções disponibilizadas.

```python
from flask_sqlalchemy import SQLAlchemy
from app import app

db = SQLAlchemy(app)


class Server(db.Model):
    __tablename__ = 'servers'

    id = db.Column(db.Integer, primary_key=True)
    hostname = db.Column(db.String())
    datacenter = db.Column(db.String())
    rack = db.Column(db.String())
    position = db.Column(db.String())
    video_port = db.Column(db.String())
    mgnt_port = db.Column(db.String())
    status = db.Column(db.String())
    uptime = db.Column(db.String())

    def __init__(self, hostname, datacenter, rack, position, \
    video_port, mgnt_port, uptime="0", status="DISCOVERING"):
    
        self.hostname = hostname
        self.datacenter = datacenter
        self.rack = rack
        self.position = position
        self.video_port = video_port
        self.mgnt_port = mgnt_port
        self.uptime = uptime
        self.status = status
```

Figura 3.22: Definição da classe de servidores utilizando SQLAlchemy.

```
dcm=> select * from servers;
 id | hostname | datacenter | rack | position |     video_port     |     mgnt_port     | status | uptime
----+----------+------------+------+----------+--------------------+-------------------+--------+--------
 46 | server 1 | GTA UFRJ   | 1    | 1        | video0 @ 1-2:1.0   | ttyUSB0 @ 4-2:1.0 | ON     | 2096
 47 | server 2 | GTA UFRJ   | 1    | 2        | video1 @ 2-6:1.0   | ttyUSB0 @ 4-2:1.0 | ON     | 2095
(2 rows)
```

Figura 3.23: Tabela criada pelo SQLAlchemy no banco de dados.

Para criar a tabela que armazena os dados dos servidores cadastrados no sistema, a classe da Figura 3.22 foi definida com o uso da biblioteca. Nesse código, os atributos do objeto criado representam as colunas mapeadas no banco de dados. Com isso, após feitas as configurações necessárias com os dados de acesso ao banco de dados, a biblioteca faz a criação da tabela no banco de dados, espelhando a estrutura definida na classe, conforme observa-se na Figura 3.23. Para criar um novo elemento, basta instanciar um objeto definido pela classe. As manipulações do banco de dados são feitas por meio de métodos herdados pela classe criada, fazendo com que toda manipulação feita no banco de dados com *queries* SQL seja transparente para a aplicação.

3.3.2 *Software* do módulo embarcado de gerência

O módulo embarcado de gerência se comunica com o servidor central por meio de uma comunicação serial. O componente do módulo responsável por esta comunicação é um Arduino Pro Micro, conforme foi descrito na Figura 3.20 com o exemplo de comunicação do teclado remoto, e também no fluxo de comunicação da leitura do estado de energia, da Figura 3.21.

O Arduino Pro Micro presente no módulo funciona como um atuador, executando tarefas recebidas do servidor central, através da tradução de comandos recebidos em ações.

Para o desenvolvimento do projeto, decide-se que a comunicação entre o Arduino e o servidor central seria feita por um sistema de códigos decimais. Cada ação solicitada pelo servidor central ao Arduino é um código decimal distinto e pré configurado no *software* que foi desenvolvido para ambas as plataformas.

Esse sistema de códigos utiliza valores decimais. A cada nova função implementada, um código livre é alocado e sua implementação é feita no *software* do servidor central, onde se associa a função desejada ao código definido, e também no Arduino do módulo embarcado de gerência.

A Figura 3.24(a) apresenta a implementação do código 1019. O código 1019 implementa o pressionamento das teclas Ctrl+ Alt + Del. Já a Figura 3.24(b) apresenta a implementação do código de leitura do estado de energia do servidor gerenciado, código 1202.

```
case 1019:
  BootKeyboard.press(KEY_RIGHT_CTRL);
  BootKeyboard.press(KEY_LEFT_ALT);
  BootKeyboard.press(KEY_DELETE);
  delay(100);
  BootKeyboard.releaseAll();
  delay(10);
break;
```

(a) Código 1019: Envio das teclas Ctrl + Alt + Del

```
case 1202:
  buttonState = digitalRead(powerState);
  if (buttonState == HIGH) {
    Serial1.write('1');
  } else {
    Serial1.write('0');
  }
break;
```

(b) Código 1202: Realiza a leitura do estado de energia e retorna informação

Figura 3.24: Implementação de códigos no Arduino

Capítulo 4

Apresentação do Sistema

Este capítulo apresenta as páginas web fornecidas pelo servidor web apache da aplicação, juntamente com o DCM App.

São mostradas as principais telas do sistema, de forma apresentar a visão do usuário do sistema desenvolvido. São demonstradas como são feitas algumas operações na aplicação, e ainda, se apresentam alguns casos de uso do sistema.

4.1 Tela de *login*

A tela de *login* do sistema, vista na Figura 4.1, é provida pelo sistema de autenticação Keycloack. Quando o usuário não está autenticado e tenta acesso a algum dos links da aplicação, ele é redirecionado para a tela de login. Nela é possível ingressar na aplicação com um usuário e senha pré-cadastrados.

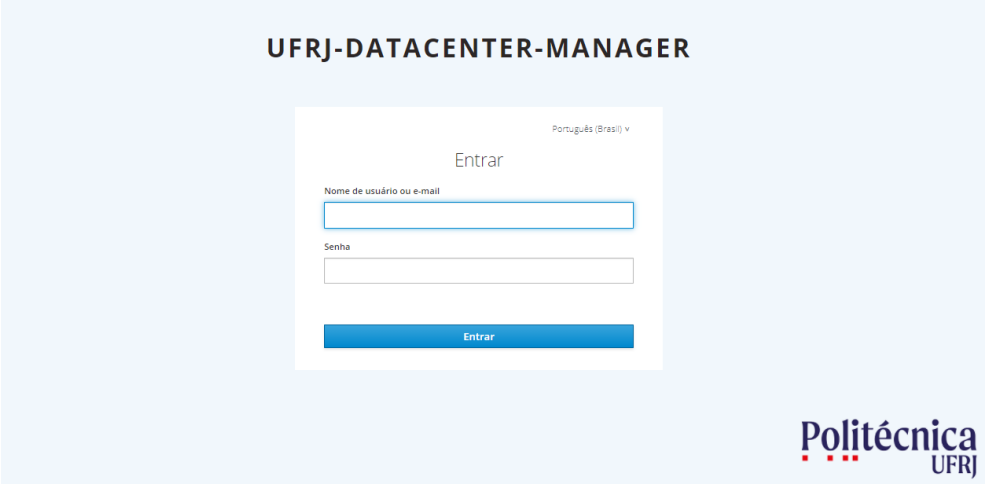


Figura 4.1: Tela de *login* do sistema.

4.2 *Dashboard*

O *Dashboard* é a tela inicial que o usuário visualiza assim que faz login no sistema. Nela é possível ter um resumo geral do sistema e informações sobre os servidores cadastrados na plataforma.

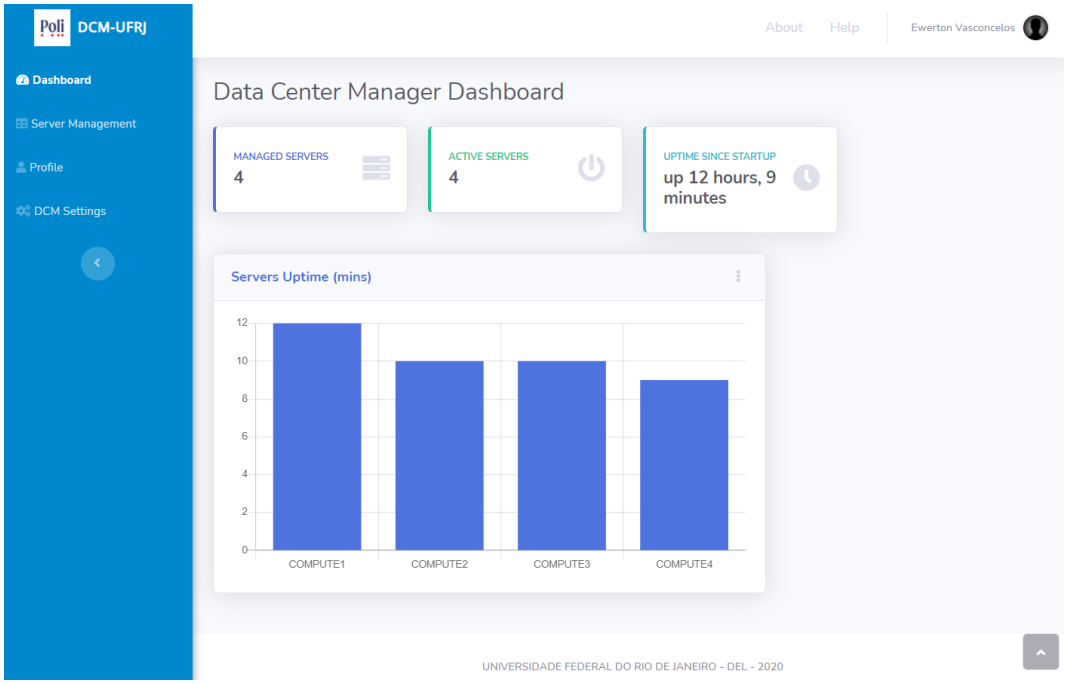


Figura 4.2: Tela inicial da aplicação

Esses itens são as informações que podem ser obtidas na tela inicial do sistema, apresentada na Figura 4.2:

- **Indicador de número de servidores cadastrados:** O total de servidores cadastrados e gerenciados pela plataforma.
- **Indicador de número de servidores ativos:** O total de servidores monitorados que estão ligados.
- **Tempo de atividade do sistema:** Tempo total em que o servidor central está ativo e monitorando o *data center*.
- **Gráfico de *uptime* dos servidores:** Indica, em cada uma das barras, o tempo total que cada um dos servidores está ligado de forma ininterrupta. Toda vez que um servidor é desligado, sua contagem de tempo total é reiniciada. Como podemos observar na Figura 4.2, existem quatro servidores sendo

monitorados pelo sistema e estão ligados a aproximadamente 12, 10, 10 e 9 minutos, de acordo com o gráfico.

4.3 Perfil do Usuário

A tela de perfil do usuário pode ser acessada pelo menu lateral da aplicação, como podemos verificar na Figura 4.3 , através da opção *Profile*. Nessa tela é possível ver as informações do cadastro do usuário, trocar a foto do perfil, e acessar o perfil de administração para troca de senha e outras informações. As operações no perfil do usuário são demonstradas na Seção 4.3.2.

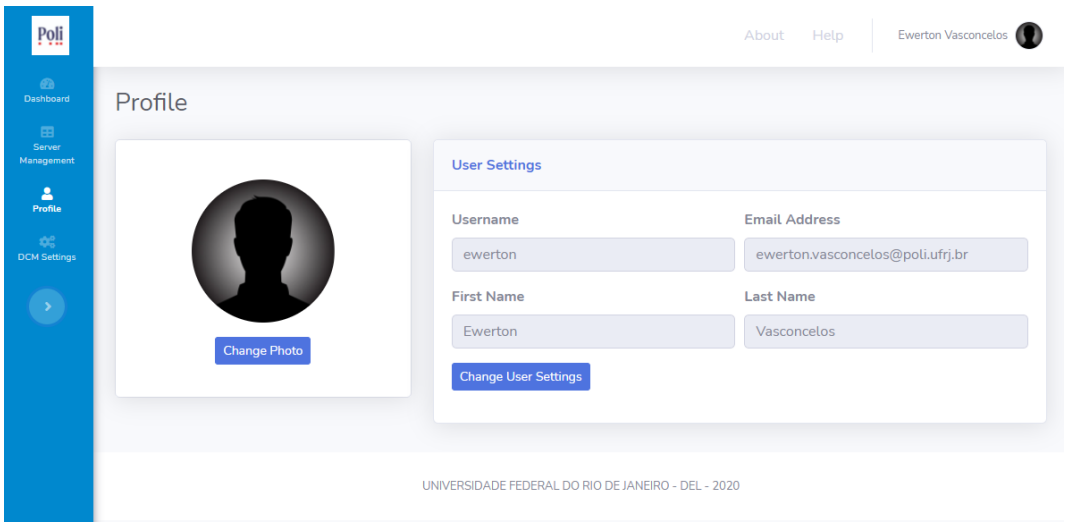


Figura 4.3: Tela de perfil do usuário.

4.3.1 Troca de foto

Com o botão *Change Photo* é possível buscar uma foto localmente e carregá-la ao perfil do usuário. O processo pode ser visto na Figura 4.4. Quando o botão para abrir a imagem é pressionado, a nova imagem é enviada para e atualizada no perfil do usuário.

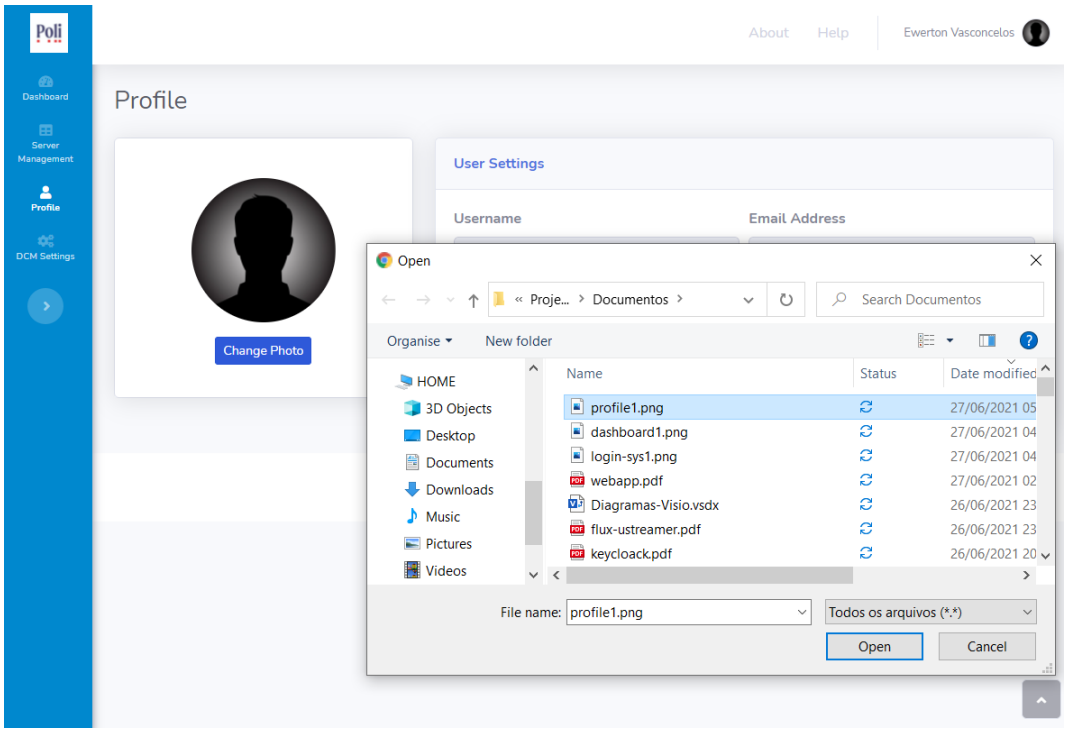


Figura 4.4: Carregar foto de perfil.

4.3.2 Administração de perfil

Ainda na tela de perfil do usuário, há um botão chamado *Change User Settings*, no qual o usuário pode acessar o seu perfil no Keyclock, conforme apresentado na Figura 4.5, e alterar dados de cadastro ou atualizar a senha.

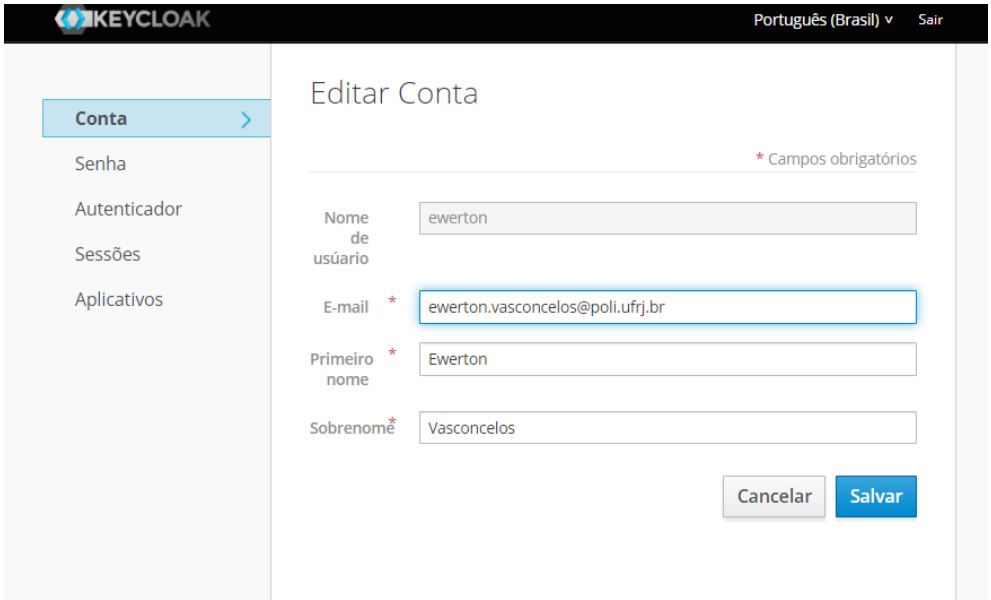


Figura 4.5: Configurações do perfil no Keycloack.

4.4 Lista de servidores

A lista de servidores cadastrados e gerenciados pela plataforma é acessada pelo menu lateral do sistema, selecionando o item *Server Management*.

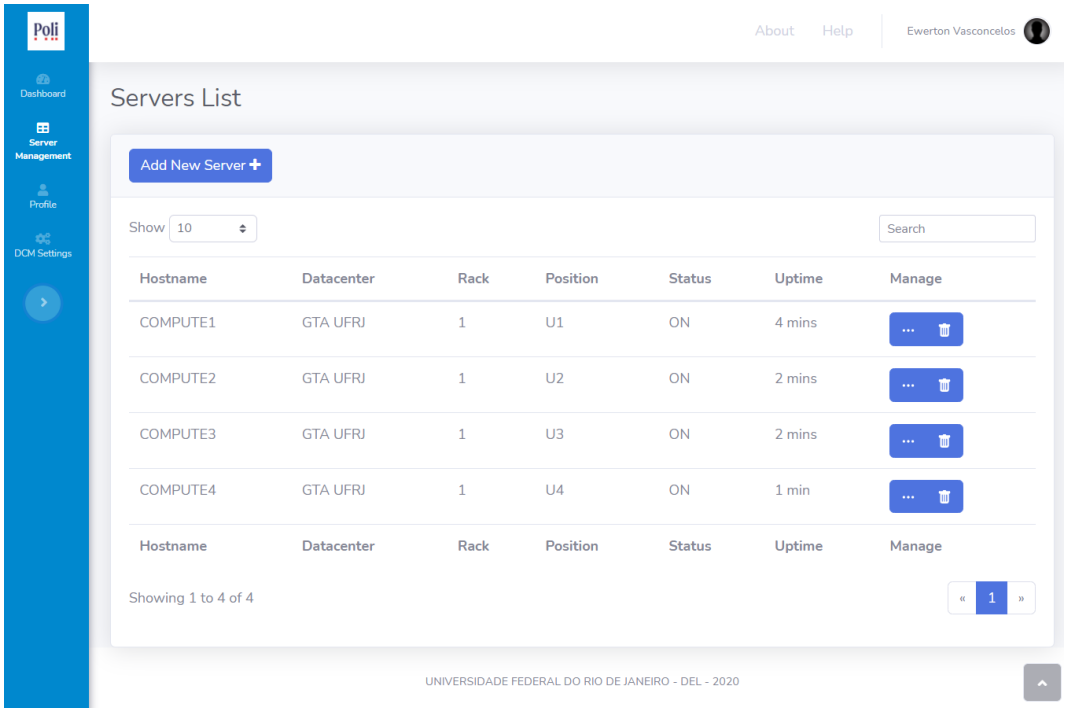


Figura 4.6: Lista de servidores

Na lista de servidores, há dados da sua localização física, botões de acesso ao console e um botão utilizado para deletar um servidor da lista. A lista pode ser navegada por páginas e é possível limitar o número de servidores exibidos em cada página. Um exemplo da lista de servidores e da tela em que ela é apresentada está na Figura 4.6.

Para cada servidor, é possível configurar um *hostname*, que é um identificador do *hardware*, o nome do *data center* (ou sala) a qual ele pertence, número do *rack* e a posição do servidor no *data center*. Além disso, o estado atual de energia e tempo em atividade também são mostrados na lista.

4.4.1 Cadastro de um novo servidor

Ainda na aba da lista de servidores, acima da lista, está o botão de cadastro de novo servidor. Ao selecionar esse botão, o usuário recebe um formulário, apresentado na Figura 4.7, no qual pode preencher as informações do novo servidor.

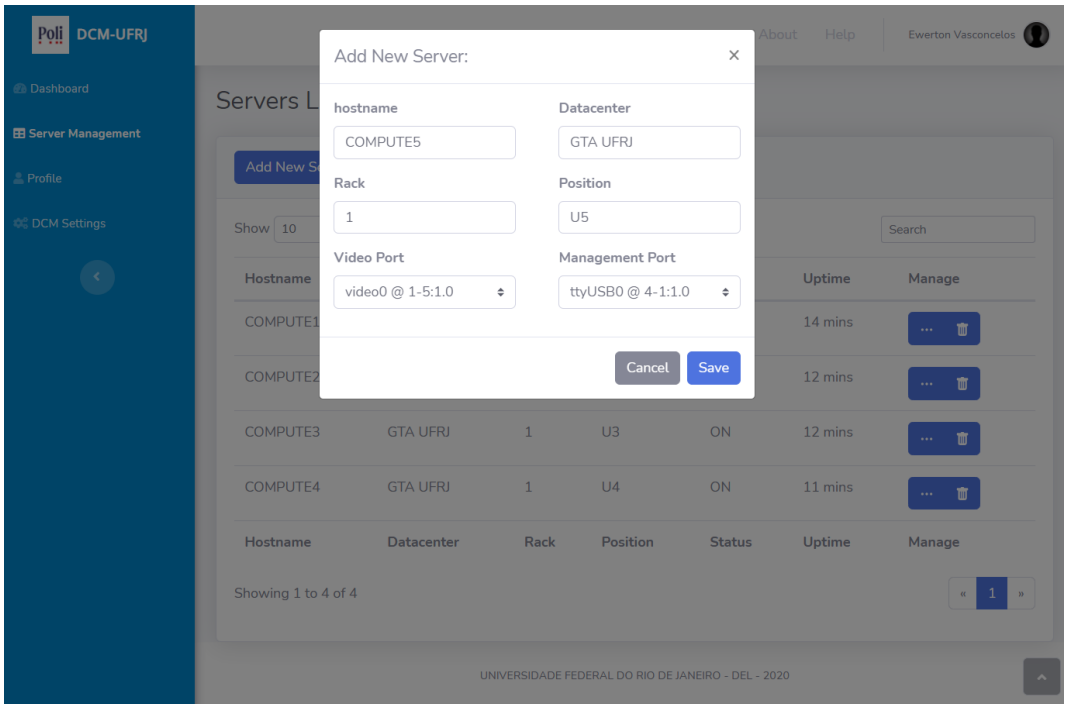


Figura 4.7: Formulário de adição de novo servidor.

4.4.2 Remoção de servidor da lista

Todos os servidores cadastrados na plataforma precisam ter seu *hardware* devidamente instalado e conectado ao servidor central. Com isso, o servidor pode ser monitorado e gerenciado. Cada servidor cadastrado no sistema é constantemente consultado sobre seu estado de energia, para que essa informação seja refletida de maneira atualizada.

Caso haja problema com o módulo embarcado de gerência de um servidor ou ele seja removido do *rack*, ele deve também ser removido da lista de servidores. Cada servidor tem um botão com ícone de "lixeira" na coluna *Manage*. Assim, para remoção, é necessário clicar nesse botão e confirmar a operação no formulário que surge, indicado na Figura 4.8.

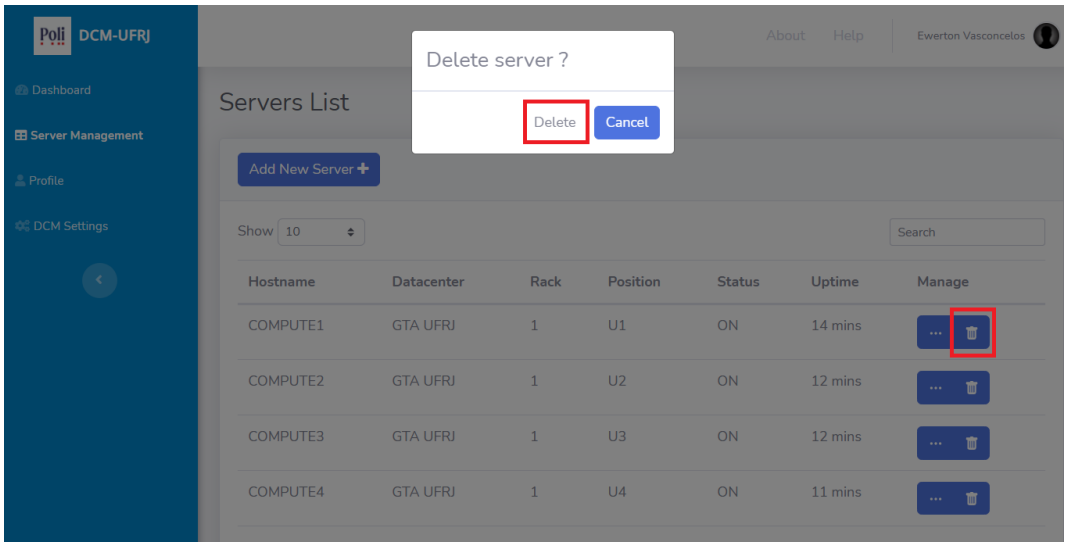


Figura 4.8: Remoção servidor.

Assim que um servidor é removido da lista, o resultado da operação é indicado em um cartão acima da lista de servidores, como visto na Figura 4.9.

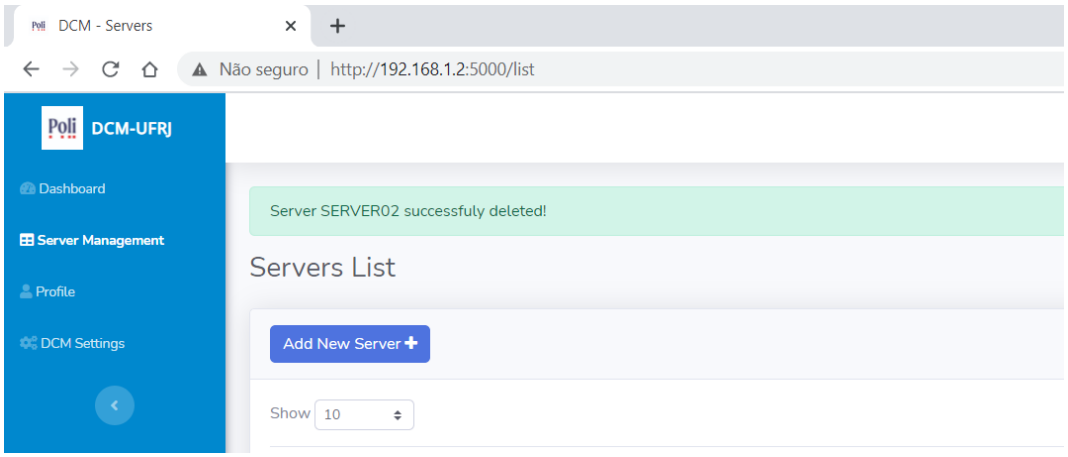


Figura 4.9: Cartão de confirmação.

4.5 Tela de gerência

É na tela de gerência que se realizam as operações em cada um dos servidores administrados pela plataforma. Ela pode ser acessada através da lista de servidores, no botão presente na coluna *Manage*, como indicado na Figura 4.10.

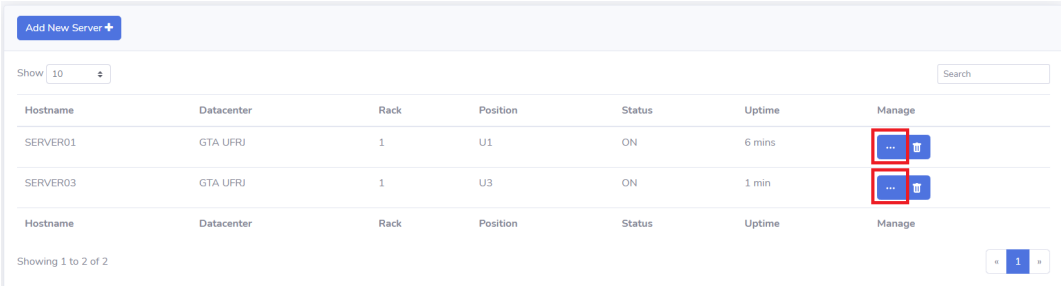


Figura 4.10: Lista de servidores com destaque ao botão de acesso à tela de gerência.

A tela de gerência dos servidores possui uma série de recursos. Para que seja possível explicar de maneira mais didática as funcionalidades do sistema na tela de gerências, a Figura 4.11 possui índices para seus elementos, que são explicados a seguir.

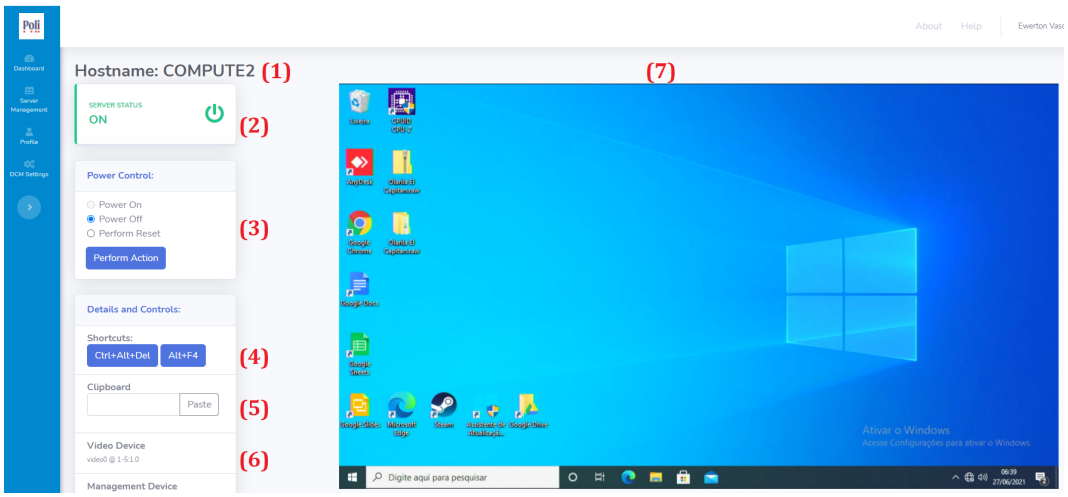
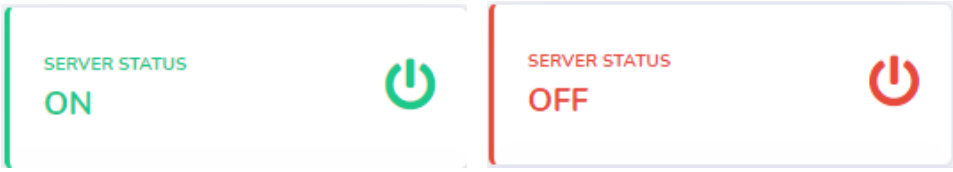


Figura 4.11: Tela de gerência.

1. É nome do servidor cadastrado na aplicação. Como este campo é do tipo *string*, qualquer tipo de dado pode ser adicionado, como *hostname* ou IP de acesso.

2. Representa o estado atual de energia do servidor. Quando o servidor está ligado, indica o estado **ON** e as cores do cartão ficam em verde como visto na Figura 4.12(a). Quando desligado, o estado é **OFF** e sua cor padrão é o vermelho, como mostrado na Figura 4.12(b).

3. Este cartão possui os comandos de energia do servidor. Com ele, executam-se as opções de energia de ligar, desligar e reiniciar. As opções ficam liberadas



(a) Cartão *on* (b) Cartão *off*

Figura 4.12: Cartões de estado de energia.

de acordo com o estado atual do servidor. Caso ele esteja desligado, somente a opção de ligar fica disponível. No estado de ligado, as opções de desligar e reiniciar se habilitam. Assim que a opção desejada é selecionada, basta pressionar o botão Perform Action e confirmar a ação.

4. Os botões de *shortcuts* são atalhos para funções que não podem ser enviadas diretamente do teclado do cliente para o servidor gerenciado, pois são capturadas e executadas localmente devido ao funcionamento padrão do sistema operacional. Teclas de Ctrl + Alt + Del e Alt + F4 foram as teclas adicionadas nesse campo, pois são atalhos de uso frequente que se enquadram na limitação de que não podem ser pressionadas diretamente pelo cliente.

5. O campo de *clipboard* é uma região onde é possível colar textos do cliente para o servidor gerenciado. Como não existe uma integração entre os sistemas operacionais, não é possível enviar um texto copiado para o servidor gerenciado. Com isso, para que seja possível que um conteúdo salvo na região de cópia do cliente seja enviado ao servidor gerenciado, esse conteúdo deve ser colado na região de *clipboard* e o botão Paste pressionado. Assim, o sistema irá enviar o conteúdo ao servidor gerenciado.

6. Os campos *video device* e *management device* apresentam as informações dos dispositivos de vídeo e controle utilizados para gerir o servidor ao qual se está conectado.

7. É a região da tela de controle que mostra a saída de vídeo do servidor gerenciado. Com ela, é possível ver todas as informações de vídeo enviadas para a saída de vídeo padrão do servidor gerenciado e obtidas pelo componente de captura de vídeo. Esta região também é responsável pela captura do teclado

remoto. Quando o usuário faz um clique com o mouse em qualquer posição da região de vídeo, as teclas do seu teclado passam a ser capturadas e toda tecla pressionada é enviada para o servidor gerenciado. Para desativar a captura, basta realizar um novo clique fora da região de vídeo.

4.6 Gestão de usuários

Como já mencionado na Seção 3.3.1.3, o Keycloack é o sistema responsável pelas tarefas de autenticação e gestão de usuários. Com isso, o procedimento de adicionar, remover ou modificar um usuário é feito diretamente na interface de administração do Keycloack. O acesso a essa interface é feito por uma URL específica. No caso do ambiente de desenvolvimento deste trabalho, essa interface está no mesmo IP do servidor central e com uma porta específica, diferente da porta de acesso à aplicação. O usuário de acesso é um usuário administrador do sistema, diferente dos usuários da aplicação.

Os seguintes passos exemplificam o procedimento para acesso à interface de gerência dos usuários.

1. Acesse a tela inicial de administração do Keycloack e selecione ***Administration Console***:

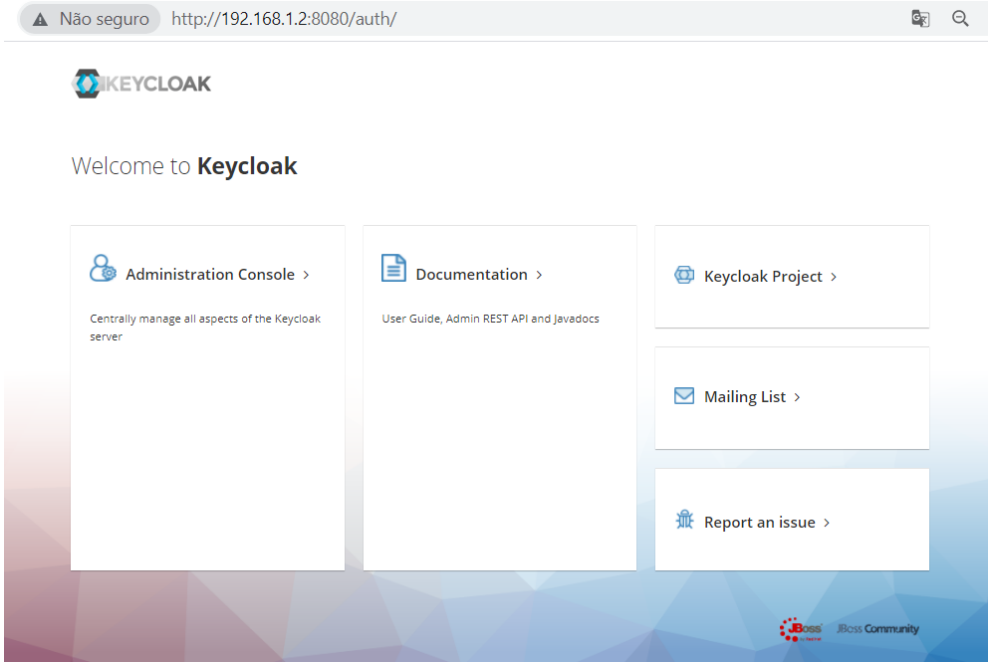


Figura 4.14: Tela inicial do Keycloack.

2. Insira o usuário de administração do sistema:

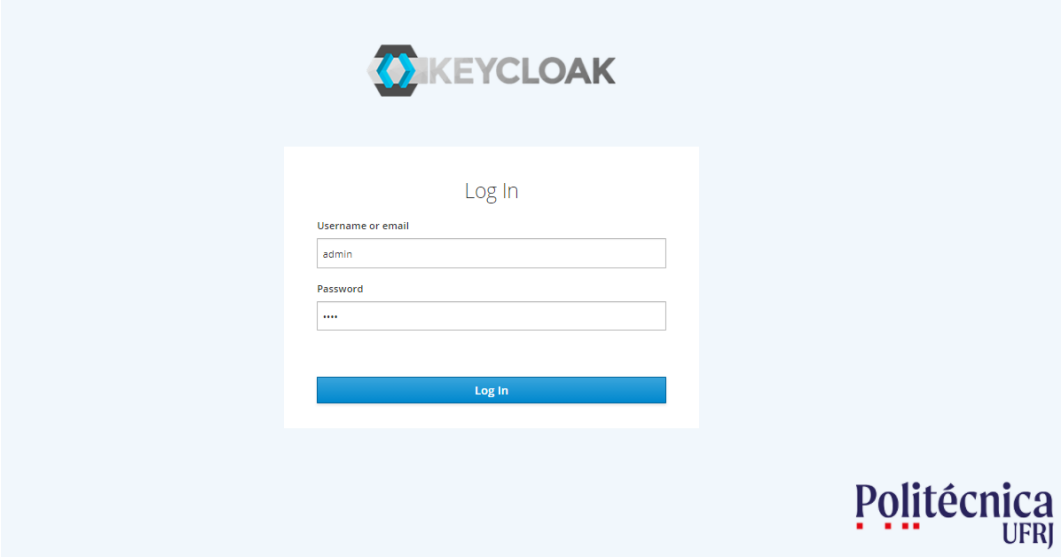


Figura 4.15: *Login* na interface de administração.

3. A tela de gestão dos usuários aparecerá e é possível realizar as seguinte ações destacadas na Figura 4.16 :

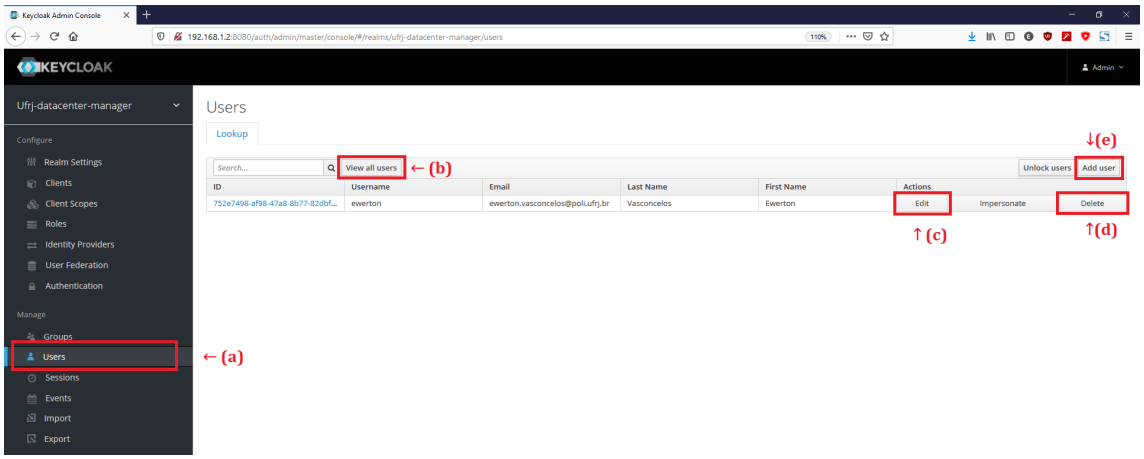


Figura 4.16: *Login* na interface de administração.

(a) Entrar na aba de administração de usuário, que é a representada na figura.

(b) Listar todos os usuários da aplicação.

(c) Editar um usuário existente.

(d) Remover um usuário existente.

(e) Adicionar um novo usuário.

4.7 Casos de uso

Esta seção busca mostrar alguns dos casos de uso propostos para este trabalho, exemplificando assim as possibilidades que são adicionadas com o uso da solução.

4.7.1 Acesso à BIOS

Possibilidade de acesso à BIOS dos servidores de maneira remota, como podemos ver na Figura 4.17. Tornando possível a mudança de parâmetros, manutenção, ou customização temporária para viabilizar a solução de um problema.

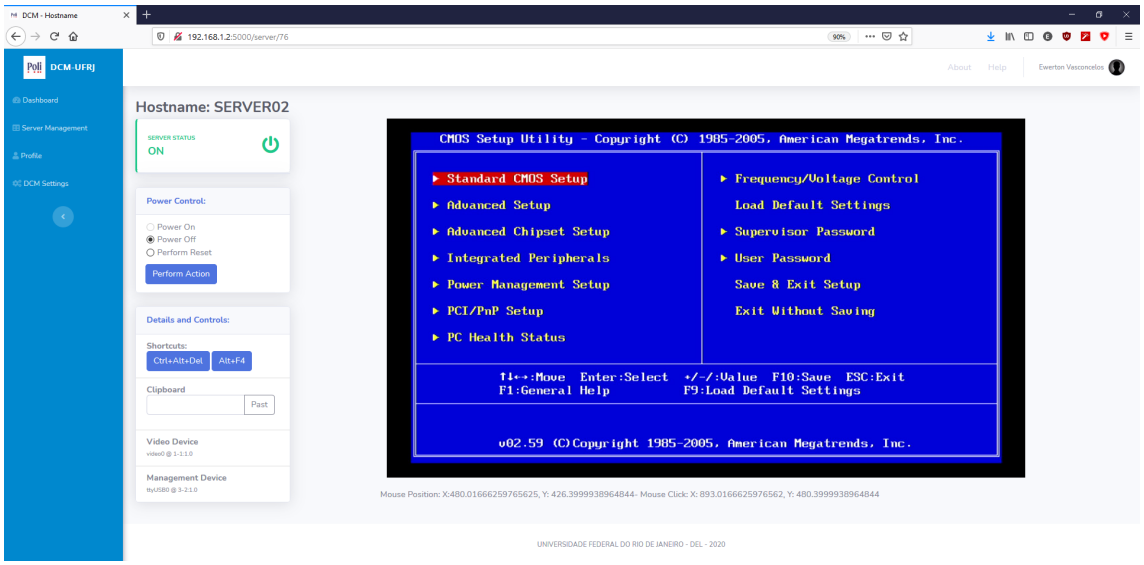


Figura 4.17: Acesso à BIOS de um servidor.

4.7.2 Monitoramento de tempo ativo

Através do *Dashboard* da aplicação, podemos ver um gráfico com o tempo em que os servidores estão ligados, sem precisar se quer acessa-los. Com isso, pode-se monitorar quedas ou a reinicialização dos servidores. A Figura 4.18 apresenta um gráfico de quatro servidores monitorados, onde um dos servidores foi recentemente reiniciado. Portanto seu tempo ativo é visivelmente menor que dos demais servidores cadastrados.

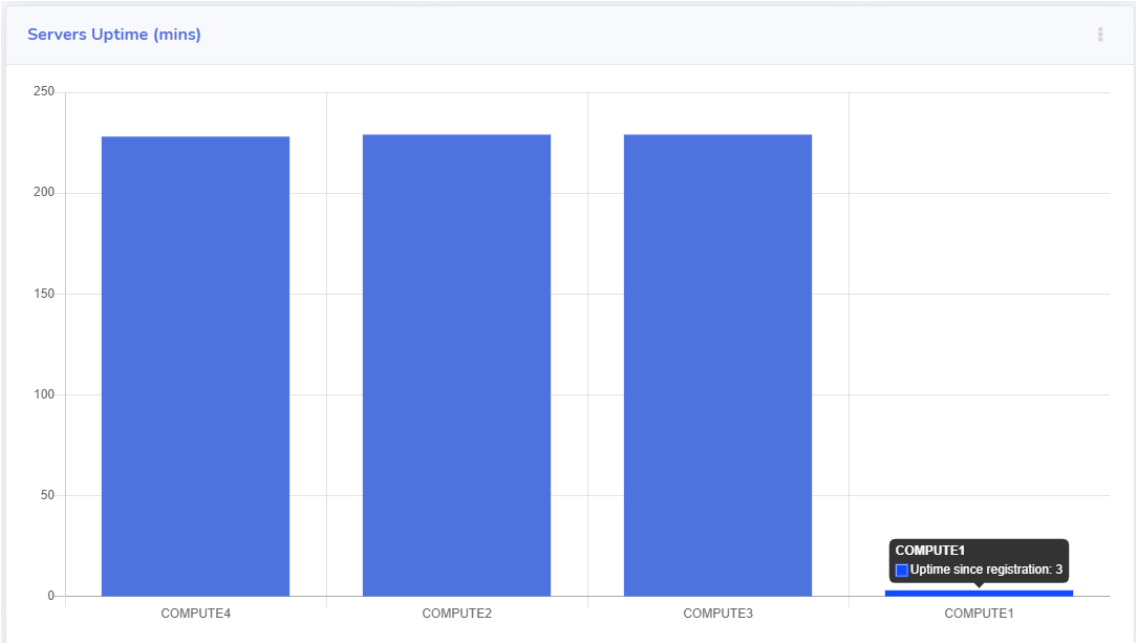


Figura 4.18: Gráfico de tempo ativo dos servidores.

4.7.3 Configurações iniciais de maneira remota

Caso haja algum problema e o servidor gerenciado perca suas configurações de rede, ele pode ser acessado e as configurações de rede podem ser reestabelecidas , com o uso dos recursos de gestão dos servidores apresentados na Seção 4.5.

4.7.4 Controle de energia

Caso necessário, é possível fazer uma reinicialização "fisica" do servidor gerenciado através dos controles de energia, como demonstrados na Figura 4.19.

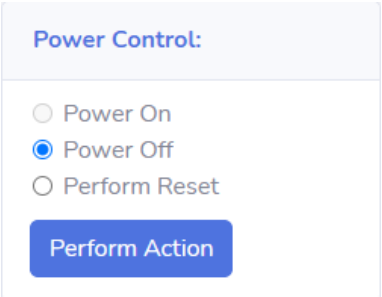


Figura 4.19: Controles de energia.

4.8 Sobre o sistema

Essa é uma aba informativa do sistema que pode ser acessada selecionando a opção *about* na barra superior do sistema. Como podemos verificar na Figura 4.20, tem-se um resumo sobre o projeto, informações de contato e a referência a *softwares* de terceiros utilizados.

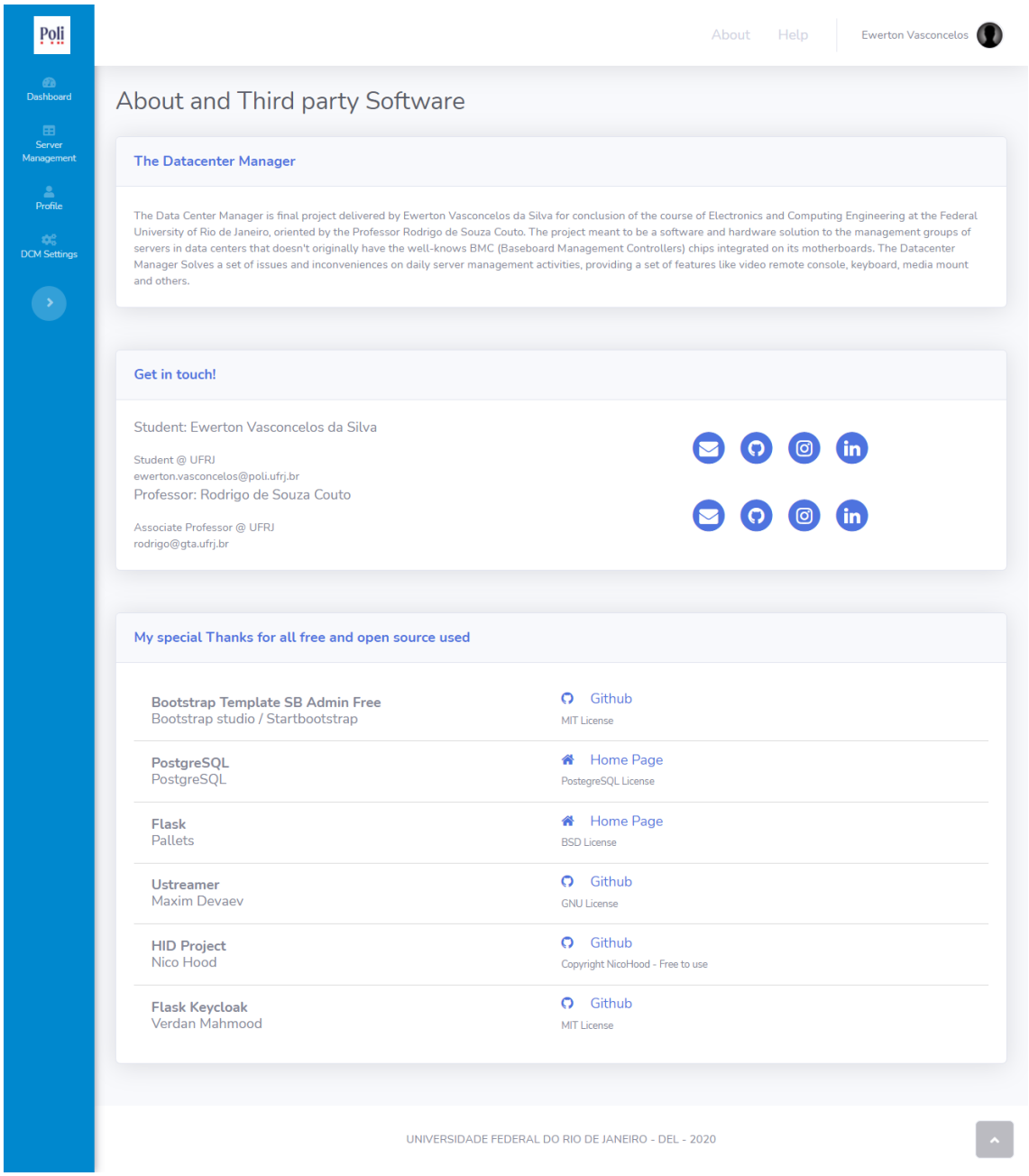


Figura 4.20: Sobre o sistema.

Capítulo 5

Testes e resultados

Este capítulo descreve os testes realizados durante o desenvolvimento do projeto. Tais testes tem como objetivo auxiliar na determinação das especificações do sistema, guiar a escolha de componentes e auxiliar no entendimento dos limites da aplicação.

5.1 Uso dos controladores USB

As interfaces do servidor central com o módulo embarcado de gerência utilizam conexões USB, que é um padrão mantido pela *USB Implementers Forum* [17]. Esse padrão especifica conectores, cabos e protocolos de comunicação para conexão e fornecimento de energia entre computadores e periféricos. Compreender o quanto de banda os periféricos USB do sistema necessitam se mostrou necessário para a definição das especificações do sistema, assim como compreender suas limitações, e também implementar partes do sistemas que auxiliam no tratamento dessas limitações.

As aplicações utilizadas para a realização desse teste foram o ***usbtop*** e o ***usbview 2.0***, que foram instaladas e utilizadas de acordo com os procedimentos fornecidos nos Anexos A.1 e A.2, respectivamente.

O consumo de banda USB das placas de captura de video foi medido da seguinte forma:

- Inicialmente, como indicado na Figura 5.1, conectam-se duas placas de captura ao servidor central, e observa-se que elas são direcionadas para os controladores

EHCI, independentemente de qual porta USB seja conectada. Os campos da Figura 5.1 destacados em amarelo mostram, à esquerda, a placa de captura no controlador EHCI e, à direita, a banda alocada.

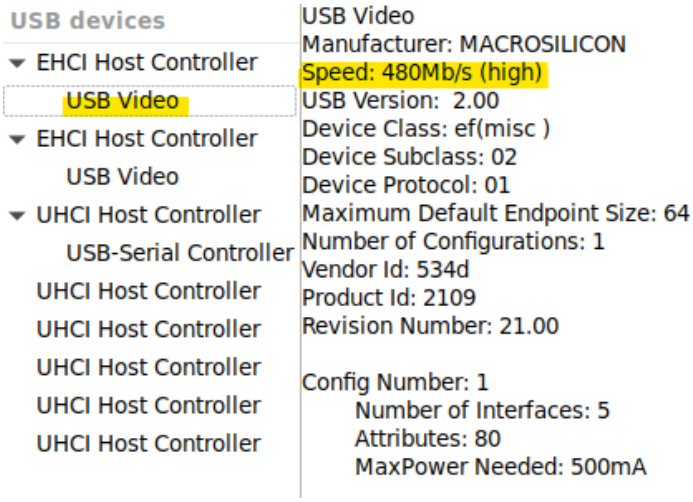


Figura 5.1: Distribuição das placas de captura nos controladores USB.

- No monitoramento do *usbtop*, indicado na Figura 5.2, observa-se que não há tráfego de informações nas portas USB, já que nenhuma captura foi iniciada e o monitor remoto de nenhum servidor foi acessado pela aplicação.

```
Bus ID 0 (All USB buses)        To device       From device
  Device ID 1 :                 0.00 KiB/s      0.00 KiB/s
  Device ID 2 :                 0.00 KiB/s      0.00 KiB/s
Bus ID 1 (USB bus number 1)     To device       From device
  Device ID 1 :                 0.00 KiB/s      0.00 KiB/s
  Device ID 2 :                 0.00 KiB/s      0.00 KiB/s
Bus ID 2 (USB bus number 2)     To device       From device
  Device ID 1 :                 0.00 KiB/s      0.00 KiB/s
  Device ID 2 :                 0.00 KiB/s      0.00 KiB/s
Bus ID 3 (USB bus number 3)     To device       From device
  Device ID 1 :                 0.00 KiB/s      0.00 KiB/s
  Device ID 2 :                 0.00 KiB/s      0.00 KiB/s
Bus ID 4 (USB bus number 4)     To device       From device
  Device ID 1 :                 0.00 KiB/s      0.00 KiB/s
Bus ID 5 (USB bus number 5)     To device       From device
  Device ID 1 :                 0.00 KiB/s      0.00 KiB/s
Bus ID 6 (USB bus number 6)     To device       From device
  Device ID 1 :                 0.00 KiB/s      0.00 KiB/s
Bus ID 7 (USB bus number 7)     To device       From device
  Device ID 1 :                 0.00 KiB/s      0.00 KiB/s
Bus ID 8 (USB bus number 8)     To device       From device
  Device ID 1 :                 0.00 KiB/s      0.00 KiB/s
```

Figura 5.2: Consumo de banda USB sem capturas.

- Em seguida, realiza-se o acesso ao monitor de um servidor da plataforma e monitora-se o consumo de banda do controlador USB, como visto na Fi-

gura 5.3. A primeira linha com valores destacados em amarelo apresenta o consumo total de todos os dispositivos conectados ao controlador. A segunda linha, indica o consumo de banda da placa de captura de vídeo USB, e por fim a terceira linha indica o conversor serial USB utilizado para a comunicação com o Arduino. A divergência entre o valor total e a soma do consumo individual se dá pelo atraso entre a observação e a disponibilização da informação pela ferramente no momento que a captura dos dados foi obtida.

```
Bus ID 0 (All USB buses)        To device       From device
  Device ID 1 :                 0.00 KiB/s      0.00 KiB/s
  Device ID 2 :                 141.72 KiB/s    15259.30 KiB/s
Bus ID 1 (USB bus number 1)     To device       From device
  Device ID 1 :                 0.00 KiB/s      0.00 KiB/s
  Device ID 2 :                 141.73 KiB/s    15257.78 KiB/s
Bus ID 2 (USB bus number 2)     To device       From device
  Device ID 1 :                 0.00 KiB/s      0.00 KiB/s
  Device ID 2 :                 0.00 KiB/s      0.00 KiB/s
Bus ID 3 (USB bus number 3)     To device       From device
  Device ID 1 :                 0.00 KiB/s      0.00 KiB/s
  Device ID 2 :                 1.03 KiB/s      1.02 KiB/s
Bus ID 4 (USB bus number 4)     To device       From device
  Device ID 1 :                 0.00 KiB/s      0.00 KiB/s
Bus ID 5 (USB bus number 5)     To device       From device
  Device ID 1 :                 0.00 KiB/s      0.00 KiB/s
Bus ID 6 (USB bus number 6)     To device       From device
  Device ID 1 :                 0.00 KiB/s      0.00 KiB/s
Bus ID 7 (USB bus number 7)     To device       From device
  Device ID 1 :                 0.00 KiB/s      0.00 KiB/s
Bus ID 8 (USB bus number 8)     To device       From device
  Device ID 1 :                 0.00 KiB/s      0.00 KiB/s
```

Figura 5.3: Consumo de banda USB com acesso a um servidor.

- Finalmente, testa-se a ocupação de banda com o acesso ao monitor de dois servidores cadastrados na aplicação, apresentado na Figura 5.4. Observamos que, a primeira linha em destaque mostra o consumo total instantâneo dos dispositivos conectados ao controlador. A segunda linha em destaque apresenta o consumo da primeira placa de captura de vídeo, e a terceira linha apresenta os dados de banda da segunda placa de captura adicionada ao teste. Nota-se, que no momento da captura não há consumo de banda pelo conversor serial USB, como observado no item anterior.

```
Bus ID 0 (All USB buses)        To device       From device
  Device ID 1 :                 0.00 KiB/s      0.00 KiB/s
  Device ID 2 :                 283.44 KiB/s    30517.17 KiB/s
Bus ID 1 (USB bus number 1)     To device       From device
  Device ID 1 :                 0.00 KiB/s      0.00 KiB/s
  Device ID 2 :                 141.73 KiB/s    15259.00 KiB/s
Bus ID 2 (USB bus number 2)     To device       From device
  Device ID 1 :                 0.00 KiB/s      0.00 KiB/s
  Device ID 2 :                 141.71 KiB/s    15259.70 KiB/s
Bus ID 3 (USB bus number 3)     To device       From device
  Device ID 1 :                 0.00 KiB/s      0.00 KiB/s
  Device ID 2 :                 0.00 KiB/s      0.00 KiB/s
Bus ID 4 (USB bus number 4)     To device       From device
  Device ID 1 :                 0.00 KiB/s      0.00 KiB/s
Bus ID 5 (USB bus number 5)     To device       From device
  Device ID 1 :                 0.00 KiB/s      0.00 KiB/s
Bus ID 6 (USB bus number 6)     To device       From device
  Device ID 1 :                 0.00 KiB/s      0.00 KiB/s
Bus ID 7 (USB bus number 7)     To device       From device
  Device ID 1 :                 0.00 KiB/s      0.00 KiB/s
Bus ID 8 (USB bus number 8)     To device       From device
  Device ID 1 :                 0.00 KiB/s      0.00 KiB/s
```

Figura 5.4: Consumo de banda USB com acesso a dois servidores.

Observando os dados obtidos pelas aplicações e o comportamento do sistema, os dispositivos de captura sempre se alocam um em cada controlador do tipo EHCI. Além disso, de acordo com dados da aplicação *usbview*, os dispositivos negociam o uso da velocidade máxima do controlador USB.

Outro comportamento identificado, mostrado na Figura 5.2, indica que as placas de captura de video só trafegam dados com o servidor quando a captura de vídeo é iniciada, pois o *Bus ID 2* só indicou consumo da banda quando o vídeo do segundo servidor foi acessado.

Durante os testes de captura de vídeo, os servidores gerenciados que foram utilizados estavam conectados à interface gráfica do sistema operacional e executando um vídeo através do navegador para garantir variabilidade nas imagens. Apesar disso, durante os experimentos realizados, o consumo das placas de captura de vídeo de *uplink* (dervidor para a placa de captura) não ultrapassou $141,73\ KiB/s$ e o consumo de *downlink* (placa de captura para o servidor) não ultrapassou $15259,70$ KiB/s, o que mostra que usa-se menos de 27% do total do barramento.

Todavia, mesmo não ocupando completamente a capacidade do controlador EHCI, cada placa de captura deve ser ativada em um controlador dedicado. Com isso, o número de servidores que podem ser acessados pela interface do sistema fica limitado ao número de controladores EHCI disponíveis no servidor central. O servidor central utilizado neste trabalho possui dois controladores EHCI, como observa-se

na Figura 5.1, o que permitira o acesso ao vídeo de dois servidores gerenciados simultaneamente.

5.2 Ocupação de recursos do servidor central

Esta seção tem o objetivo de identificar a quantidade de recursos consumida no servidor central, com base no número de servidores que são gerenciados e na quantidade de clientes conectados. Assim, executa-se um teste de monitoramento de métricas de CPU, memória RAM e consumo de rede do servidor central em alguns cenários.

Como foi constatado na Seção 5.1, existe uma limitação de acesso ao vídeo de dois servidores distintos por vez, devido ao número de controladores EHCI disponíveis no servidor central do protótipo.Para a realização dos testes, define-se a seguinte nomenclatura. Os clientes são os usuários da plataforma que fazem o acesso desde seus computadores utilizando um navegador. Portanto, o acesso simultâneo de 2 clientes a um servidor gerenciado representa que dois usuários da plataforma estão conectados à tela de gerência e vendo a saída de vídeo de um mesmo servidor.

Com isso, o primeiro teste foi realizado com o acesso de um, dois e três clientes a um servidor gerenciado. O segundo teste foi feito com um, dois e três clientes com cada um deles acessando a dois servidores gerenciados. A Figura 5.5 exemplifica essa descrição dos testes. Os cliente e servidores de cor mais clara exemplificam os novos clientes e servidores adicionados progressivamente aos testes.

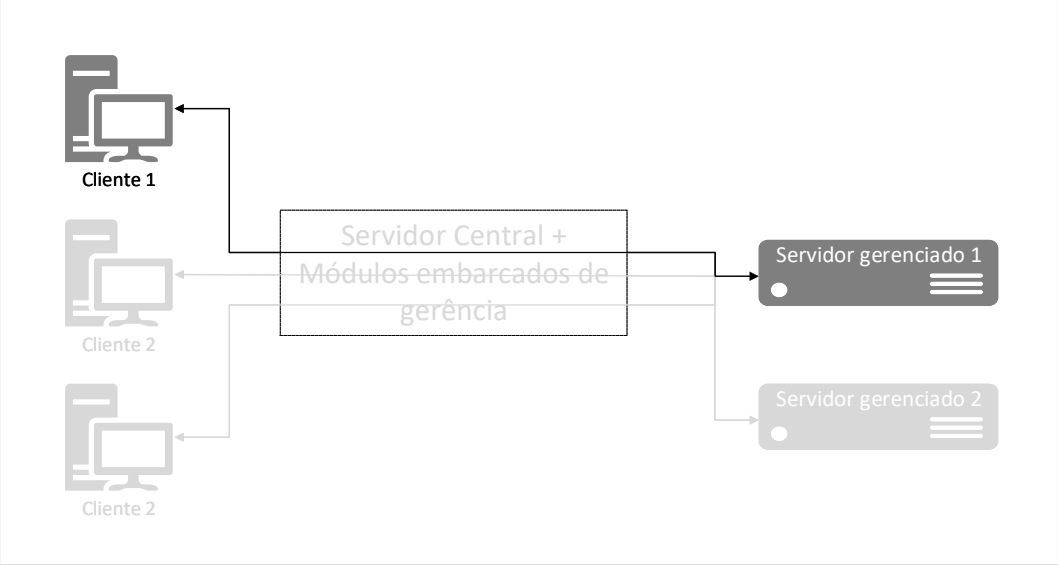


Figura 5.5: Exemplificação da realização dos testes de hardware.

A aplicação utilizada para monitoramento foi o *dstat*, que é uma ferramenta que fornece de maneira clara e direta dados sobre CPU, memória e rede em forma de uma tabela com trinta amostras tomadas a cada um segundo. O resultados do monitoramento mostrados nas Tabelas 5.1 e 5.2 são a média dos dados obtidos por um conjunto dessas amostras.

N° de Clientes	**CPU**	**Recebidos**	**Enviados**	**RAM**
1	4,8%	5,5 kiB/s	532,3 kiB/s	1236,0 Mb
2	5,0%	14,5 kiB/s	1022,2 kiB/s	1239,6 Mb
3	5,0%	27,9 kiB/s	1449,1 kiB/s	1235,1 Mb

Tabela 5.1: Acesso a um servidor por cliente.

N° de Clientes	**CPU**	**Recebidos**	**Enviados**	**RAM**
1	8,0%	10,7 kiB/s	761,0 kiB/s	1251,8 Mb
2	8,6%	25,4 kiB/s	1497,0 kiB/s	1260,1 Mb
3	8,0%	28,4 kiB/s	1629,2 kiB/s	1258,8 Mb

Tabela 5.2: Acesso a dois servidores por cliente.

Como o desenvolvimento deste trabalho não tem foco no desempenho da plataforma, as análises realizadas tem o objetivo de fornecer um panorama geral da relação entre a aplicação e hardware no qual ela foi instalada. Com isso, análises com mais amostras e um maior números de servidores gerenciados precisam ser analisados futuramente para se obter o desempenho do sistema.

O resultado obtido pelas medições condiz com o esperado para o funcionamento do sistema. Primeiramente, analisando o comportamento da CPU, seu uso cresce quando dois servidores são utilizados em comparação ao uso de um servidor. Isso é esperado pois, a cada novo servidor que inicia o monitoramento, uma nova instância do µStreamer é iniciada.

Analisando o consumo da rede, nota-se que a cada novo cliente cliente, há um acréscimo nos pacotes enviados e recebidos pelo servidor central. E, por fim, o consumo de memória RAM apresentou pouca variação considerando a quantidade de clientes e servidores analisada.

Capítulo 6

Conclusões e Trabalhos Futuros

Este trabalho propôs uma solução de monitoramento e gestão remota a servidores que não possuem esses recursos integrados. A solução se baseia no uso de tecnologias de desenvolvimento web, sistemas operacionais, arquitetura de computadores e microcontroladores, sendo um projeto que explorou diferentes conhecimentos e tecnologias.

A solução é composta por dois módulos principais que juntos viabilizam o funcionamento do sistema e fornecem ao usuário final um conjunto de ferramentas para a administração de servidores de um *data center*. O servidor central é o módulo responsável por fornecer a interface de usuário, armazenar as informações sobre o *data center* e se comunicar com o módulo embarcado de gerência. Esse último localiza-se nos servidores a serem gerenciados, executando as solicitações do servidor central e coletando informações.

A instalação da solução é fácil e pode ser adaptada para funcionar com diferentes computadores. Seu custo de implementação é baixo devido ao uso de componentes simples e de fácil acesso. Os testes executados no trabalho indicam as especificações mínimas necessárias dos componentes e de rede, além de mostrar o comportamento esperado do sistema em diferentes situações.

Porém, durante o desenvolvimento, alguns pontos foram identificados como possíveis melhorias ao sistema. Inicialmente, é possível mencionar a necessidade da implementação da integração do mouse remoto, o que pode facilitar algumas operações e melhorar a usabilidade do sistema. Como direção futura, pretende-se também avaliar a possibilidade da implementação do módulo embarcado de gerência

que utiliza o mínimo de portas USB dos servidores no seu funcionamento para que seja possível adicionar mais servidores gerenciados na plataforma, já que o número de portas USB do servidor central é limitado, e simplificar a instalação.

Referências Bibliográficas

[1] HCC-EMBEDDED, "Host Controllers", **https://www.hcc-embedded.com/products/usb-overview/host-controllers**, Acessado em: 28/03/2021.

[2] OPENBMC, "OpenBMC", **https://www.openbmc.org/**, Acessado em: 25/04/2021.

[3] OPENBMC, "A do everything Redfish, KVM, GUI, and DBus webserver for OpenBMC.", **https://github.com/openbmc/bmcweb**, Acessado em: 13/06/2021.

[4] PIBMC, "Build A Raspberry Pi BMC", **https://www.boniface.me/post/a-raspberry-pi-bmc/**, Acessado em: 25/04/2021.

[5] SILVEIRA, C. B., "Desvendando a Comunicação RS232", **https://www.citisystems.com.br/rs232/**, Acessado em: 14/07/2021.

[6] TLDP, "Optionally configure the BIOS", **https://tldp.org/HOWTO/Remote-Serial-Console-HOWTO/bios.html**, Acessado em: 13/06/2021.

[7] MDEVAEV, "Pi-KVM", **https://github.com/pikvm/ustreamer**, Acessado em: 25/04/2021.

[8] KVM-SWITCHES-ONLINE, "KVM-Switches", **https://www.kvm-switches-online.com/kvm-over-ip-guide.html**, Acessado em: 13/06/2021.

[9] KEYCLOAK, "Keycloack Docs", **https://www.keycloak.org/docs/**, Acessado em: 30/03/2021.

[10] PROGRAMMERSOUGHT, "Introduction to OHCI, UHCI, EHCI and XHCI Controllers", **https://www.programmersought.com/article/1457246744/**, Acessado em: 28/03/2021.

[11] TOSHIBA, "TC358840 Camera Serial Interface Converter Chipset (4K HDMI to MIPI CSI -2)", **https://toshiba.semicon-storage.com/content/dam/toshiba-ss/ncsa/en_us/docs/product-brief/assp/14A04_TC358840_ProdBrief.pdf**, Acessado em: 21/02/2021.

[12] FARNELL, "Arduino Pro Micro", **http://www.farnell.com/datasheets/1685581.pdf**, Acessado em: 21/03/2021.

[13] SILVA, E. V. D., "Um sistema de gestão remota para servidores em data centers", **https://github.com/ewertonvsilva/dcm**, Acessado em: 28/06/2021.

[14] LUBUNTU, "Lubuntu Docs", **https://lubuntu.net/**, Acessado em: 23/03/2021.

[15] MDEVAEV, "µStreamer", **https://github.com/pikvm/ustreamer**, Acessado em: 7/04/2021.

[16] PALLETSPROJECTS, "Flask", **https://flask.palletsprojects.com/en/2.0.x/**, Acessado em: 26/06/2021.

[17] USB.ORG, "USB (Universal serial bus)", **http://www.usb.org/**, Acessado em: 02/03/2021.

[18] LEFEBVRE, W., "About top", **http://www.unixtop.org/License.shtml**, Acessado em: 03/03/2021.

[19] GUINET, A., "usbtop", **https://github.com/aguinet/usbtop**, Acessado em: 03/03/2021.

Apêndice A

Ferramentas para o monitoramento do sistema do servidor central

Para a elaboração de alguns dos testes com o objetivo de determinar as especificações do servidor central, foi necessário realizar o monitoramento de diferentes indicadores do sistema como o uso e ocupação dos controladores USB, uso de CPU e memória RAM do sistema. Para a realização dessas medidas, foi necessário instalar algumas ferramentas. Este apêndice tem o objetivo de explicitar o procedimento de instalação e operação dessas ferramentas.

A.1 Instalação do *usbtop*

O usbtop é um utilitário semelhante ao top [18], que mostra uma largura de banda instantânea estimada em barramentos e dispositivos USB [19]. O procedimento abaixo considera o sistema operacional utilizado no desenvolvimento do projeto, o *Lubuntu 18.04*.

Instalar as dependências:

```
$ sudo apt update
$ sudo apt install cmake git libboost-dev libpcap-dev \\
libboost-thread-dev libboost-system-dev
```

Copiar o reposirório para a máquina local:

```
$ git clone https://github.com/aguinet/usbtop.git
```

Compilar o código fonte na máquina local:

```
$ cd usbtop
$ mkdir _build && cd _build
$ cmake -DCMAKE_BUILD_TYPE=Release ..
$ make
$ sudo make install
```

Após compilar, carregar o móludo usbmon no sistema:

```
$ sudo modprobe usbmon
```

Para iniciar a aplicação e monitorar a ocupação de banda dos dispositivos USB, execute o seguinte comando:

```
$ sudo usbtop
```

Saída do comando:

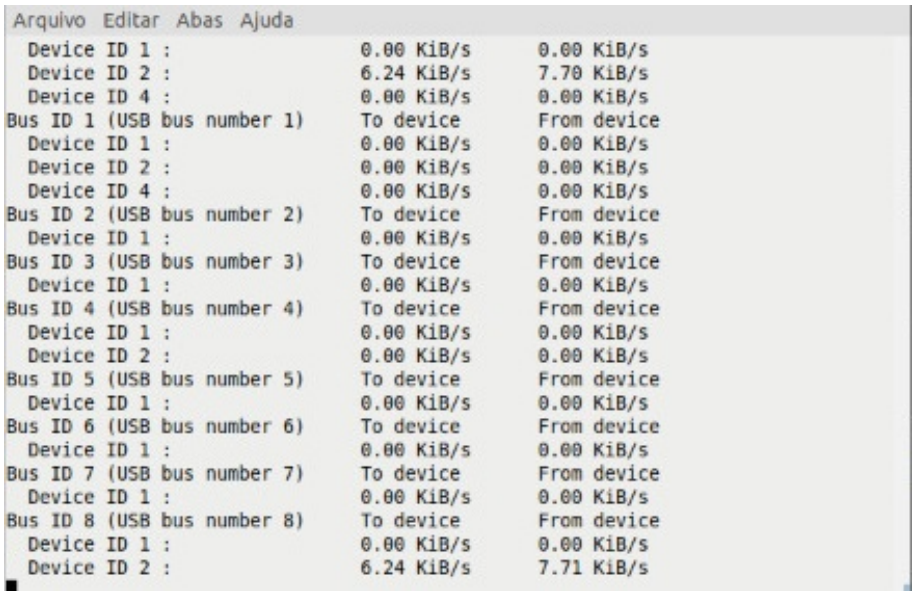


Figura A.1: Ferramenta *usbtop*

A.2 Como instalar o *usbview*

Baixar pacote com aplicação:

```
$ wget  http://www.kroah.com/linux-usb/usbview-2.0.tar.gz
```

Extrair pacote:

```
$ tar - xzvf usbview-2.0.tar.gz
```

Preparar ambiente e compilar aplicação:

```
$ cd usbview-2.0
$ sudo ./configure
$ sudo make
$ sudo make check
$ sudo make install
```

Executar aplicação:

```
$ sudo ./usbview
```

Funcionamento esperado:

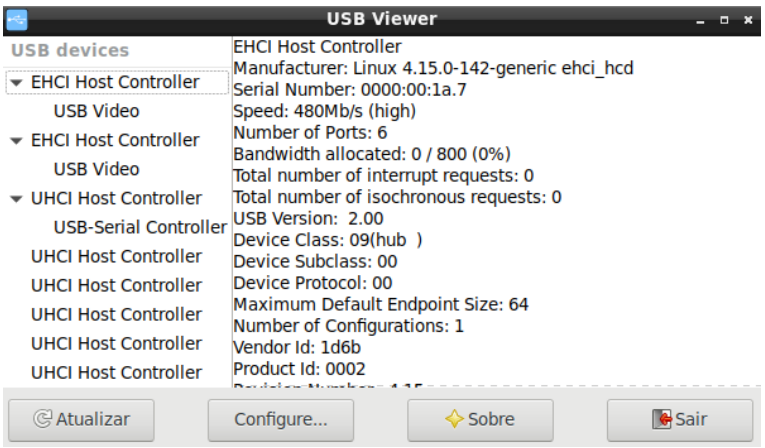


Figura A.2: Ferramenta *usbview*