CENTRO UNIVERSITÁRIO UNIFACVEST
CURSO CIÊNCIA DA COMPUTAÇÃO
ALÉFF MOURA DA SILVA

WOLF MONITOR

LAGES
2020

ALÉFF MOURA DA SILVA

SISTEMA DE MONITORAMENTO DE SERVIDORES

Trabalho de Conclusão de Curso de Ciência da Computação apresentado ao Centro Universitário UNIFACVEST como parte dos requisitos para obtenção do título de bacharel em Ciência da Computação.

Orientador: Cassandro Devenz, MSc.

Co-orientador: Igor Muzeka, MSc.

LAGES
2020

ALÉFF MOURA DA SILVA

WOLF MONITOR

Trabalho de Conclusão de Curso de Ciência da Computação apresentado ao Centro Universitário UNIFACVEST como parte dos requisitos para obtenção do título de bacharel em Ciência da Computação.

Orientador: Cassandro Devenz, MSc.

Co-orientador: Igor Muzeka, MSc.

Lages, SC__/__/2020.

Nota _____

_____

Msc. Márcio José Sembay

Coordenador do Curso de Graduação de Ciência da Computação

LAGES
2020

RESUMO

No presente trabalho, objetiva-se apresentar um projeto para automatização em migrações e monitoramento de servidores, sejam eles com sistema Linux ou Windows, a fim de melhorar a forma com que as empresas lidam com serviços e com arquivos de configurações de programas computacionais em seus servidores. Desta forma, não se fazendo necessário efetuar uma conexão direta no servidor, sendo feito todo o controle em um sistema.

O sistema é dividido em três fases, a primeira etapa, um serviço em execução no servidor da empresa, a segunda etapa API da Web (sistema que pode receber informações via internet) que receberá as informações enviadas por esse servidor e por último um sistema/aplicativo para computadores que será responsável por exibir as informações transmitidas em telas amigáveis para o usuário.

O aplicativo de exibição poderá enviar informações ao serviço para que possa ser executada determinadas funções. Algumas funções permitidas são, reiniciar serviços e alterar valores em arquivos de configuração, desta forma auxiliando uma migração de servidores.

Palavras-chave: Monitoramento de servidores; Serviço de sistema; Automatização;

ABSTRACT

In the present work, the objective is to present a project for automation in migrations and monitoring of servers, be they with Linux or Windows system, how to improve the way companies deal with services and configuration files of computer programs on their servers. This way, it is not necessary to make a direct connection to the server, with all the control being done on a system. The system is divided into three stages, one, a service running on the company's server, the other a Web API (system that can receive information over the Internet) that will receive the information sent by that server and a system / application for computers that will be responsible For displaying the information transmitted on easy-to-use screens.

The display application can send information to the service so that certain functions can be performed. Some functions allowed are, restarting services and changing values in configuration files, thus assisting a server migration.

Keywords: Server monitoring; System service; Automated;

LISTA DE FIGURAS

Figura 1: Projetos de micro serviços do Sistema. 30
Figura 2: Bancos de dados utilizados no Sistema 31
Figura 3: Exemplo de resposta JSON na solicitação de serviços monitorados 32
Figura 4: Listagem no sistema desktop dos serviços monitorados 33
Figura 5: Resumo do Agent cadastrado no sistema 33
Figura 6: Detalhamento do Agent cadastrado 34
Figura 7: Cadastro de agent 35
Figura 8: Arquivo de configuração do agent 35
Figura 9: JSON de configurações do agent 36

SUMÁRIO

1 INTRODUÇÃO 9
1.1 OBJETIVO GERAL 10
1.2 OBJETIVOS ESPECÍFICOS 10
1.3 JUSTIFICATIVA 11
2 SUPORTE TEÓRICO 12
2.1 SISTEMAS OPERACIONAIS 12
2.2 SISTEMA OPERACIONAL WINDOWS 13
2.3 SISTEMA OPERACIONAL LINUX 13
2.4 Linux Vs Windows 14
2.5 TCP/IP 15
2.6 HTTP 16
2.7 SERVIÇOS 18
2.8 SERVIÇOS DO WINDOWS 18
2.9 SERVIÇOS DO LINUX 19
2.10 WEB API 19
2.11 .NET CORE 20
2.12 WPF 20
2.13 PROGRAMAÇÃO ORIENTADA A OBJETOS 21
2.14 INTEROPERABILIDADE 21
2.15 JSON 22
2.16 BANCO DE DADOS 22
2.17 SQL SERVER 22
2.18 ENTITY FRAMEWORK 23
2.19 DDD 24
2.20 DOCKER 24
2.21 MICRO SERVIÇO 25
2.22 PADRÃO DE DESENVOLVIMENTO GATEWAY 26
2.23 OCELOT 26
2.24 RABBITMQ 27
2.25 AUTOMAPPER 28
2.26 IDENTITY SERVER 28

2.27 GERENCIAMENTO DE REDES 29
3 DESCRIÇÃO DA IMPLEMENTAÇÃO. 30
3.1 WEB API 30
3.2 BANCO DE DADOS 31
3.3 DESENVOLVIMENTO DESKTOP 32
3.4 Agent – Serviço do Linux/Windows 34
4 CONCLUSÃO 37
5 REFERÊNCIAS BIBLIOGRÁFICAS 38

1 INTRODUÇÃO

No Governo de Castelo Branco (1964-1967) “iniciou-se um processo de fortalecimento e modernização de instituições e de instrumentos para fomentar a pesquisa e a evolução técnica” (MOTOYAMA, 2004, p.324). Vemos essa evolução tecnológica bastante nos dias atuais, onde quase todos podem ter acesso a tecnologia de ponta; bons computadores, bons celulares entre outros, oque tornou tudo um pouco mais acessível, desde fazer compras a falar com um parente distante.

> Foi criado a partir do Banco Nacional de Desenvolvimento Econômico (BNDE) o Fundo de Desenvolvimento Cientifico e Tecnológico (FUNTEC), com o intuito de financiar profissionais e pesquisadores da mais alta qualificação, adornando as empresas para pesquisa e desenvolvimento (P&D). (MOTOYAMA, 2004). O desenvolvimento tecnológico tem proporcionado a criação de novos produtos e serviços, bem como a alteração da estrutura produtiva e o surgimento de novos modelos de negócios (PEREZ, FREEMAN,1988).

Com base nisso, nota-se perante as empresas de desenvolvimento e tecnologia uma constante preocupação na melhoria de sistema e na evolução dos mesmos, para que a cada dia se torne possível realizar e criar sistemas que a alguns anos não tinha tecnologia suficiente.

Com base no contexto de Freeman, torna-se possível a existência deste presente trabalho, que se constitui em um novo modelo para migrações e auxilio em servidores, para que desta maneira, as empresas de tecnologia possam ter uma maior eficiência ao efetuarem migrações em seus computadores de redes e possam ter uma maior praticidade no dia a dia.

Corroborando com essa ideia, o presente trabalho visa tornar mais prática a ação de desenvolvedores e analistas de redes no momento de efetuar uma migração de servidor ou somente uma visualização de forma rápida do estado de um determinado serviço ou de algum arquivo que configure alguma aplicação. Será possível, na tela do sistema visualizar essas informações, de uma forma simples e amigável, onde qualquer usuário poderá saber a situação atual do conteúdo monitorado.

1.1 OBJETIVO GERAL

Desenvolver um sistema capaz de auxiliar programadores e analistas de sistemas a implantar migrações e realizar monitoramento completo de serviços e arquivos de configurações em servidores por uma interface gráfica amigável e simples de utilizar.

1.2 OBJETIVOS ESPECÍFICOS

- Desenvolver um Serviço para Windows e Linux para monitorar servidores com esses sistemas operacionais;
- Desenvolver uma aplicação na internet que disponibilize informações em forma de dados de texto;
- Criar uma tela de interface gráfica onde serão exibidas as informações salvas na aplicação da internet;
- Implementar os itens acima citados em um servidor;
- Analisar e implementar estratégias para melhorias e problemas futuros;

1.3 JUSTIFICATIVA

O presente projeto visa criar um conjunto de sistemas que trará um auxílio e um conforto para desenvolvedores e analistas de redes, onde com base nesse conjunto de sistemas, será possível o monitoramento de uma série de servidores de computadores e *datacenters* ou popularmente conhecidos como nuvem. As empresas possuirão acesso a todo esse conjunto para que possam monitorar seus servidores e desta forma ter um maior controle sobre sua rede.

> Uma pesquisa recente, realizada pelo The Boston Consulting Group (BCG), foi identificado que as pequenas e médias empresas que adotam a infraestrutura de TI em nuvem ganham competitividade em comparação às concorrentes no mercado. No Brasil, a receita anual destas empresas é aumentada em 16%, e a criação de novos empregos ocorre 11% mais rápido em comparação com as de menor adoção da computação em nuvem. Isto, deve-se ao processo de automação para pequenas e médias empresas. Ao invés de gerenciar as operações individualmente, a gestão da empresa tem a possibilidade de controlar as informações de maneira centralizadas, a partir de aplicações. Gerando maior rapidez e eficiência. (SKYONE, 2018)

Com base nestas informações o presente trabalho torna-se justificável pelo auxílio na forma que estas empresas irão monitorar seus *datacenters*, onde com o Wolf Monitor será possível efetuar migrações de forma rápida, sem que os servidores passem muito tempo parados, tornando-as mais simples e rápidas ou até mesmo melhorando a forma de visualização do que ocorre em sua rede de computadores, e quanto menos tempo os servidores passarem inacessíveis, melhor será para a empresa.

Partindo destas primícias o presente trabalho procura desenvolver um serviço para servidores Linux e Windows sendo que o foco principal será em Linux por toda sua praticidade ao qual será informada em tópicos futuros, um sistema para guardar as informações e repassá-las de maneira simples e prática e um aplicativo para computadores comuns onde qualquer usuário que possua uma conta válida poderá visualizar as informações dos servidores de sua empresa.

2 SUPORTE TEÓRICO

Neste capítulo será abordado e apresentado os estudos com intuito de oferecer uma base de conhecimento para os tópicos de desenvolvimento do projeto e citando também alguns problemas no setor computacional de servidores.

2.1 SISTEMAS OPERACIONAIS

Pode-se definir, toda a interface de um computador/celular ou outro dispositivo ao qual seja possível uma interação com o usuário, como um sistema operacional, ao qual sobre ele, será executado diversos programas e aplicativos para que de tal forma, aquele que estiver utilizando-o consiga usufruir de seus benefícios.

Um sistema operacional é onde tudo acontece no computador, ele é responsável por gerenciar os mais variados programas, como por exemplo um navegador de internet, sem um sistema operacional não seria possível a utilização do mesmo. “Podemos dizer que um computador não possui nenhuma utilidade prática sem pelo menos um sistema operacional instalado.” (Richard Batista Silveira).

Felizmente na atualidade temos computadores bem mais robustos, com telas gráficas e funcionando de forma simples, possuímos diversos sistemas operacionais, onde apresentam as mais diversas interfaces amigáveis e que muitas pessoas utilizam, sejam em celulares ou computadores, dois dos sistemas operacionais mais conhecidos para computadores são Windows e Linux.

2.2 SISTEMA OPERACIONAL WINDOWS

O Windows é um sistema operacional que utiliza o conceito de visualização, onde é possível com cliques de mouse e utilização de teclado, interagir com todo o sistema operacional, sendo possível o acesso a internet, utilização de jogos, criação e visualização de vários tipos de dados e informações.

> Com a chegada da interface gráfica, tudo mudou. O Windows não foi inovador, foi uma cópia de um sistema desenvolvido pela Apple, que, por sua vez, foi uma melhoria do sistema da Xerox. A dupla Steve Jobs e Steve Wozniak criaram o Lisa, sistema operacional baseado na interface gráfica da Xerox. (História do Windows, INFOESCOLA, 2006).

Embora o Windows não tenha sido o primeiro na criação de sistemas operacionais com interfaces gráficas, o mesmo se tornou o mais popular no mercado, por sua fácil utilização, o primeiro sistema operacional Windows era uma mistura entre o sistema MS-DOS com um ambiente gráfico.

2.3 SISTEMA OPERACIONAL LINUX

O Linux é um sistema operacional de código fonte aberto e gratuito, o qual possui diversas distribuições, cada distribuição é mantida por uma organização sem fins lucrativos, ou uma comunidade de usuários, para um melhor entendimento do Linux, primeiro faz-se necessário compreender, o que é um Kernel? Kernel nada mais é que um núcleo do sistema operacional, onde o sistema operacional é carregado após a inicialização do mesmo.

> Cabe ao Kernel fazer a intermediação entre o hardware e os programas executados pelo computador. Isso significa que a junção do Kernel mais os softwares que tornam o computador usável (drivers, protocolos de comunicação, entre outros), de acordo com a sua aplicação, é que formam o sistema operacional em si. (INFOWESTER, 2011).

Uma boa analogia para melhor compreensão seria, comparar o Kernel com um chassi de moto, dependendo da empresa que adquira um chassi, a mesma poderá construir modelos diferentes de motos, cada moto com o mesmo chassi, o Kernel seria o chassi, e o sistema operacional a moto montada completamente.

O Linux é um sistema operacional, muito robusto, e atualmente utilizado em várias plataformas, tanto em servidores onde salvam arquivos e mantem *sites onlines* como também em celulares *smartphones* aos quais utilizam *android*, que é um sistema operacional Linux.

2.4 Linux Vs Windows

Nessa briga entre sistemas operacionais, é comum ler sobre qual deles é o melhor, porém a realidade é que ambos possuem usabilidades diferentes, enquanto o Windows está mais preocupado em ser utilizado por usuários comuns o Linux é um sistema operacional mais voltado a desempenho e utilização em servidores, ou por desenvolvedores de sistemas aos quais possuem bastante conhecimento, entretanto o Linux é gratuito o que torna sua utilização bem mais significativa para desenvolvimento de servidores e áreas de TI's (Tecnologia da informação) afins.

A economia não se resume aos sistemas operacionais, o Linux tem muitos aplicativos que são grátis e principalmente de boa qualidade. Se considerarmos somente o preço do sistema operacional Windows 10 Home, o mesmo está saindo no valor de R$ 230,00 conforme venda no site rupave.com, enquanto o Linux não custa valor algum para sua instalação e usufruir de todos os seus benefícios.

O presente trabalho irá utilizar-se de desenvolvimento voltado ao Linux, devido ser um sistema de código fonte aberto, ou seja, gratuito, o Windows é um sistema um tanto quanto caro, e para tudo se faz necessário compra-lo, e como o intuito é monitoramento de servidores, será voltado para o Linux o monitoramento, apenas o desenvolvimento do sistema gráfico será desenvolvido para o sistema Windows, tendo em vista que o mesmo será voltado para usuários finais.

2.5 TCP/IP

De uma forma simplificada, o protocolo TCP/IP é o principal protocolo de comunicação, envio e recebimento de informações e dados pela internet. TCP tem como significado *Transmission Control Protocol* (Protocolo de Controle de Transmissão) e o IP, *Internet Protocol* (Protocolo de Internet).

O protocolo de transmissão TCP é um dos protocolos utilizados para comunicação entre computadores, para que os mesmos se intendam, e desta maneira seja possível a troca de informações.

Segundo Santos (2014), protocolo é uma "Linguagem" utilizada pelos dispositivos em uma rede de computadores de forma que os mesmos consigam se entender, ou seja, trocar informações entre eles, todos deverão estar usando a mesma linguagem, isto é, o mesmo protocolo.

O TCP/IP é na realidade uma pilha de protocolos que em conjunto efetuam vários procedimentos, e este grupo é dividido em quatro camadas distintas: a camada de aplicação, camada de transporte, uma camada de rede e a camada de interface, estas divisões de camadas são um meio de prevenir e garantir a integridade das informações e dados que passam pela rede de computadores.

a. **Aplicação:** Esta camada é utilizada pelos sistemas e programas para que possa ser enviada e recebida informações de outros programas pela rede de computadores. É nela que se encontra os protocolos como SMTP (protocolo para envio de e-mail), FTP (protocolo responsável pela transferência de arquivos entre os computadores através da rede) e o mais famoso de todos o HTTP (para navegações pela internet). Após os dados passarem por esta camada e tenham sido processados eles são enviados para a divisão seguinte.
b. **Transporte:** A camada de transporte e rede, fica responsável por receber todos os dados que são enviados pela camada de Aplicação, ela verifica a integridade dos dados e divide-os em pacotes. Logo após as informações são enviadas para a camada de Rede, que fica loco abaixo dela.
c. **Rede:** Nesta camada, os dados que foram empacotados são recebidos e em seguida anexados a um endereço virtual (IP) de origem e de destino. Desta forma, os pacotes estão prontos para serem enviados pela internet.

d. **Interface:** É a responsável por receber os pacotes prontos para que assim possam ser enviados pela internet, os protocolos nesta camada dependem bastante do tipo de rede ao qual está sendo utilizada durante a requisição. Na atualidade o protocolo mais utilizado é o Ethernet.

Desta forma o protocolo TCP/IP é o conjunto de protocolos acima citados, que trabalhando em etapas alcançam um objetivo a comunicação entre computadores na rede mundial de computadores, ou até mesmo uma rede local.

2.6 HTTP

O HTTP (*Hyper Transmission Protocol*) tornou-se o protocolo da computação mais utilizado na internet, devido as páginas de redes sociais, sites de notícias, entre outros. Com este, é possível obter dados de um servidor na internet, ao qual o navegador de internet interpreta os dados recebidos, e mostra ao usuário em uma forma mais visível e de simples visualização.

> HTTP é um protocolo (protocol) que permite a obtenção de recursos, tais como documentos HTML. É a base de qualquer troca de dados na Web e um protocolo cliente-servidor, o que significa que as requisições são iniciadas pelo destinatário, geralmente um navegador da Web. Um documento completo é reconstruído a partir dos diferentes sub documentos obtidos, como por exemplo texto, descrição do layout, imagens, vídeos, scripts e muito mais. (MOZILLA).

Sendo assim, o usuário ao qual por meio de um navegador de internet, consegue acessar de forma simples e prática apenas com um programa de computador, desfrutar das inúmeros recursos ao qual a internet disponibiliza, como livros para estudos, vídeos aulas ou até mesmo cursar uma faculdade a distância, e tudo isso só se faz possível graças ao protocolo HTTP.

Existem diversos métodos de requisições dentro do HTTP, esses métodos são responsáveis por obter ou passar valores via chamada pelo protocolo de transmissão, vulgo HTTP, os métodos são:

- GET: É o método mais comumente utilizado na internet, tendo em vista que solicita algum determinado recurso seja ele, um arquivo, um script ou qualquer outro valor identificado na URL (endereço da internet acessado no navegador) da chamada por meio do protocolo HTTP. Este método é o método que acessamos quando entramos em algum site da internet, como por exemplo o *Facebook*, ele é o responsável por nos retornar o conteúdo que o navegador irá nos exibir;
- HEAD: Semelhante ao GET, entretanto, não é retornado recurso algum para o solicitante. É utilizado para obtenção de meta-informações por meio de cabeçalho de resposta, sem ter que recuperar todo o conteúdo da chamada;
- POST: É o responsável por enviar os dados e informações a serem processados pelo servidor, como por exemplo, um formulário de cadastro em algum site, esse formulário, quando enviado para o servidor, utiliza-se do método POST. Os dados a serem enviados são incluídos no corpo do comando e passados pela requisição;
- PUT: Semelhante ao POST, porém mais utilizado para edição de algum valor existente no banco de dados do servidor;
- PATCH: Idêntico ao PUT, com uma pequena diferença em sua regra de negócio, ele é utilizado quando se precisa alterar apenas uma parte da informação;
- DELETE: deleta determinado dado ou recurso do servidor;
- TRACE: Ecoa a solicitação de uma maneira ao qual o cliente fique ciente o que os servidores intermediários estão mudando em seu pedido;
- OPTIONS: Recupera todos os métodos/tipos de solicitações aos quais o servidor aceita;
- CONNECT: Utiliza-se esse método para o uso com um proxy ao qual possa se tornar um túnel SLL (um túnel é algo utilizado para efetuar uma conexão segura com determinado servidor).

No referido trabalho aqui presente será feita a utilização dos principais métodos HTTP, sendo eles GET, POST, PUT, PATCH, DELETE, como o Wolf Monitor se utiliza de informações obtidas e enviadas pela internet, faz-se necessário a utilização de todos os métodos acima informados para uma Web API mais completa e robusta.

2.7 SERVIÇOS

Serviços em sistemas operacionais são programas previamente criados para realizar determinada função ou ficar em execução efetuando determinado procedimento, sem que usuário saiba o que ocorre internamente no sistema operacional.

> Um serviço é um programa que roda em segundo plano fora do controle interativo dos usuários do sistema, por falta de uma interface. Isso é feito para garantir ainda mais segurança, pois alguns desses serviços são cruciais para o funcionamento do sistema operacional. (HOSTINGER, 2019).

Desta forma, um serviço muitas das vezes é utilizado para fazer algo sem que seja necessário um usuário para sua execução, executando determinada função, como por exemplo, um serviço do SQLServer, onde é responsável por execuções e conexões com o banco de dados Microsoft SQLServer.

2.8 SERVIÇOS DO WINDOWS

Serviços Windows, são programas que ficam em execução em segundo plano, ou seja, sem interação do seu usuário, serviços são necessários para manter a execução de forma simples sem precisar que seu usuário tenha alguma preocupação em ter que iniciar. São muito utilizados por seu próprio sistema operacional, ou para auxilio em algum programa.

> Os serviços do Microsoft Windows, anteriormente conhecidos como serviços do NT, permitem que você crie aplicativos executáveis de longa execução que são executados em suas próprias sessões do Windows. Esses serviços podem ser iniciados automaticamente quando o computador é inicializado, podem ser colocados em pausa e reiniciados e não exibem nenhuma interface do usuário. (MICROSOFT CORPORATION, 2007).

De tal forma, podem ser criados a partir de qualquer linguagem .NET, tendo em vista que é a biblioteca de classe .NET Framework ou .NET Core que contém todos os mecanismos que possibilitam criar, instalar e implementar podendo gerenciar serviços Windows.

2.9 SERVIÇOS DO LINUX

Assim como o Windows, os serviços no Linux funcionam da mesma forma, ou seja, um sistema que fica em execução em segundo plano, fazendo determinado procedimento sem a interferência de seu usuário.

Serviços nos sistemas operacionais Linux ou Unix, são conhecidos também como *daemons*. Normalmente os nomes desses serviços/*daemons* são terminados com a letra d. Como por exemplo, *mysqld* é o nome do serviço ao qual é responsável por controlar o *MySql*.

Na utilização do presente projeto, será utilizado da linguagem de programação C# para monitorar os serviços existentes nos servidores, o C# com a utilização do .NET Core torna possível a utilização no Linux, desta forma não se faz necessário distinção entre os sistemas operacionais e a própria linguagem será responsável por faze-lo.

2.10 WEB API

Para o desenvolvimento do projeto foi escolhido o framework de Web API 2 da Microsoft, ao qual é desenvolvido com ASP.NET. Uma api se constitui em uma fonte de informações utilizando o padrão HTTP, onde é possível diversos aplicativos e sistemas acessarem suas informações, mesmo não sendo desenvolvidas na mesma linguagem de programação.

A Web API se torna uma opção viável por sua Interoperabilidade, com ela é possível que seja desenvolvida em uma linguagem, e qualquer outra aplicação criada, não importando sua linguagem de desenvolvimento, poderá acessar essas informações e utilizá-las para seu devido propósito. Como esta tecnologia é voltado a sua praticidade, ela utiliza do padrão de comunicação HTTP para que possa servir as informações para a mesma.

> HTTP não serve apenas para servir páginas da Web. O HTTP também é uma plataforma poderosa para a criação de APIs que expõem serviços e dados. O HTTP é simples, flexível e onipresente. Quase todas as plataformas que você pode considerar têm uma biblioteca HTTP, portanto, os serviços HTTP podem alcançar uma ampla variedade de clientes, incluindo navegadores, dispositivos móveis e aplicativos de desktops tradicionais. (MIKE WASSON, 2017).

Desta forma, por ser feita uma comunicação HTTP, qualquer sistema em qualquer parte do mundo poderá acessa os dados e informações do sistema, acarretando assim em uma maior compatibilidade de linguagens de programação e tornando a comunicação mais fácil.

2.11 .NET CORE

O C# é uma linguagem de programação muito bem difundida nos dias atuais, com base em .NET. A mesma foi escolhida para o desenvolvimento deste trabalho de conclusão de curso, devido a sua praticidade no desenvolvimento e sua ampla comunidade de suporte na internet, ao qual é possível a obtenção de quaisquer informações sobre a mesma, tornando assim o desenvolvimento rápido e prático.

> O .NET Core é uma plataforma de desenvolvimento de código aberto de uso geral mantida pela Microsoft e pela comunidade .NET no GitHub. É uma plataforma cruzada (compatível com Windows, macOS e Linux) que pode ser usada no desenvolvimento de dispositivos, na nuvem e em aplicativos de *IoT* (Internet e outras coisas). (GUIA DO .NET CORE, MICROSOFT).

Por ser uma linguagem multissistemas operacionais de código fonte aberto, ao qual é possível a utilização em qualquer dos sistemas operacionais mais conhecidos e utilizados, a mesma se tornou a escolha mais eficiente para o desenvolvimento deste projeto.

2.12 WPF

WPF é uma sigla dada a um framework da Microsoft, com ele é possível a criação de telas de forma simples e pratica, com ele é possível criar sistemas inteiros com interação com o usuário de Windows utilizando uma linguagem simples e pratica.

Segundo a Microsoft, “O *Windows Presentation Foundation* (WPF) fornece aos desenvolvedores um modelo de programação unificado para a criação de aplicativos de área de trabalho de linha de negócios no Windows ”. Desta forma o WPF se torna uma das melhores opções para desenvolvimento *desktop* para se trabalhar com o sistema operacional Windows.

O WPF torna possível a criação de aplicativos/programas para clientes que utilizam a área de trabalho do sistema operacional Windows, com um ambiente visualmente simples e elegante, e com bastante eficiência e velocidade.

> O núcleo do WPF é um mecanismo de renderização que não depende da resolução e baseado em vetor, criado para tirar proveito de hardwares gráficos modernos. O WPF estende o núcleo com um conjunto abrangente de funcionalidades de desenvolvimento de aplicativos que incluem linguagem XAML, controles, associação de dados, layout, elementos gráficos 2D e 3D, animação, estilos, modelos, documentos, mídia, texto e tipografia. O WPF faz parte do .NET; portanto, você pode criar aplicativos que incorporam outros elementos da API .NET. (Windows Presentation Foundation, MICROSOFT).

A utilização de WPF para este presente trabalho se dá ao fato do mesmo utilizar a linguagem XAML, a qual é bastante utilizada em desenvolvimento de aplicativos para celular, tornando assim possível futuramente uma migração mais fácil para um ambiente *mobile* (aparelho celular).

2.13 PROGRAMAÇÃO ORIENTADA A OBJETOS

A programação utilizando este paradigma revolucionou o setor computacional, utilizando de vários conceitos que aproximam a programação do mundo no qual vivemos, tornando assim o desenvolvimento de sistemas mais ágil, e executando o código da melhor forma possível.

Atualmente quase todas as linguagens de grande porte e de alto nível utilizam de orientação a objetos por sua praticidade, principalmente para escrever códigos, entendê-los e a prática de reutilização.

2.14 INTEROPERABILIDADE

A interoperabilidade em um sistema se constitui na utilização de alguma linguagem de texto, normalmente JSON ou XML para o envio e recebimento de informações, desta forma não importando qual a linguagem de origem nem a linguagem de destino, as duas se entendem como uma linguagem de texto a parte. Desta forma possibilitando que uma informação saia por exemplo de uma WEB API em C# e seja mostrada em um sistema ao qual foi totalmente desenvolvido em Java.

2.15 JSON

Uma das linguagens de texto mais utilizadas em interoperabilidade, o JSON nada mais é que uma forma leve de troca de informações/dados entre determinados sistemas, o seu significado JavaScript Object Notation, por mais que no seu nome leve JavaScript o mesmo não pode ser utilizado apenas nessa linguagem.

O JSON mesmo sendo um formato leve e rápido para troca de informações também é um formato muito simples, fácil e intuitivo de ser lido, diferente do seu principal rival no mercado o XML que mesmo sendo uma linguagem de marcação e de interoperabilidade fácil de ser lida, ainda perde para o JSON se comparado.

2.16 BANCO DE DADOS

O banco de dados nada mais é que um local onde será salva todas as informações. Existem diversos tipos e formatos de banco de dados, porém o mais conhecido na atualidade, são os bancos de dados relacionais, com estrutura de linha e coluna, formando uma tabela de informações, nas palavras de Korth, um banco de dados “é uma coleção de dados inter-relacionados, representando informações sobre um domínio específico”, desta forma, sempre que tiver informações que forem possíveis serem agrupadas e são referentes ao mesmo assunto, pode-se dizer que se possui um banco de dados.

Para o desenvolvimento deste projeto foi utilizado o banco de dados Sql Server, por ser produzido e oferecido suporte pela Microsoft, e com todo o sistema sendo feito na linguagem de programação C# torna-se o casamento perfeito, por ambos serem da Microsoft.

2.17 SQL SERVER

O Sql Server é um SGDB (sistema de gerenciamento de banco de dados) ao qual é um programa capaz de gerenciar informações/dados, se utilizando de uma linguagem para o acesso dessas informações (SQL).

> O SQL Server é um SGBD da Microsoft, criado em parceria com a Sybase, em 1988, inicialmente como um complementar do Windows NT, sendo que depois passou a ser aperfeiçoado e vendido separadamente. A parceria com a Sybase terminou em 1994, e a Microsoft continuou a melhorar o programa após isto. (INFOESCOLA, 2009).

Com o SQL Server é possível salvar informações de forma prática, onde todas essas informações são salvas com um padrão de relacionamento de tabelas, com colunas e linhas, esse padrão é conhecido como banco de dados relacional, é a maneira mais antiga de se salvar informações em um banco de dados e bastante utilizada nos dias de hoje por sua praticidade e de forma coesa.

2.18 ENTITY FRAMEWORK

O *Entity framework* é um conjunto de tecnologias para .NET que dão suporte ao desenvolvimento de softwares orientados a objetos, ele possui os mecanismos para a conexão com bancos de dados sem precisar fazer codificação no mesmo.

Com *Entity Framework* é possível criar sua aplicação, fazendo todo o desenvolvimento da mesma sem se preocupar com o relacionamento do banco de dados, tendo em vista que ele emprega o padrão Code-First, onde primeiro se desenvolve o código do sistema, e o banco de dados se molda ao mesmo.

> O *Entity Framework* permite que os desenvolvedores trabalhem com dados na forma de objetos e propriedades específicos de domínio, como clientes e endereços de clientes, sem precisar se preocupar com as tabelas e colunas de banco de dados subjacentes em que esses dados são armazena. Com este *Framework*, os desenvolvedores podem trabalhar em um nível mais alto de abstração ao lidar com dados e podem criar e manter aplicativos orientados a dados com menos códigos do que em aplicativos tradicionais. Como o mesmo é um componente do .NET Framework, os aplicativos *Entity Framework* podem ser executados em qualquer computador no qual o .NET Framework a partir da versão 3,5 SP1 esteja instalado. (MICROSOFT, 2018).

O *Entity Framework* em poucas palavras, faz o intermédio, da codificação que no caso desse projeto foi realizada em C# com o banco de dados ao qual está sendo utilizado, desta forma o desenvolvedor não necessita criar códigos para o banco de dados, fazendo tudo pelo código do sistema.

2.19 DDD

O DDD (*Domain-Driven Design)* é o conjunto de princípios, fundamentos e práticas que surgiu com o intuito de diminuir problemas que sempre atormentaram os desenvolvedores de software. Se trata de um tipo de guia para aplicações de diversas práticas existentes com o intuito de melhorar a totalidade do processo de desenvolvimento de software.

O mesmo visa ajudar as equipes com os padrões de projetos para que todos tenham um melhor entendimento do contexto de todos os projetos, permitindo desta forma ser utilizado tal conhecimento para criar um produto final com uma maior qualidade e satisfação para o cliente.

Segundo a DevMedia (2009), "A principal ideia do DDD é a de que o mais importante em um software não é o seu código, nem sua arquitetura, nem a tecnologia sobre a qual foi desenvolvido, mas sim o problema que o mesmo se propõe a resolver, ou em outras palavras, a regra de negócio ". Este padrão de projeto é atualmente bem consolidado por sua forma prática, limpa e de fácil reuso de código, tendo em vista que as classes desenvolvidas dentro do sistema ficam o mais limpa possível, onde cada uma é responsável por algo, e nunca assumindo mais que uma responsabilidade.

2.20 DOCKER

O Docker é um sistema semelhante a maquinas virtuais, com ele é possível pegar o "estado" de uma máquina previamente configurada e executar em qualquer ambiente ao qual o tenha instalado, funciona de forma semelhante a java *virtual machine.* Com o docker é possível efetuar procedimentos de atualizações de forma prática e rápida, tendo em vista que todos os recursos ficam instalados no mesmo, sem precisar ser instalado no servidor.

> De forma resumida, o Docker é uma plataforma de código aberto, desenvolvido na linguagem Go e criada pelo próprio Google. Por ser de alto desempenho, o software garante maior facilidade na criação e administração de ambientes isolados, garantindo a rápida disponibilização de programas para o usuário final. (GOMES, Pedro César Tabaldi, 2018).

Utilizando o sistema de "*containers*" aos quais são a própria máquina por assim dizer, onde pode ser um container executando um sistema Linux ou Windows, dependendo da preferência do usuário, o container ele isola o servidor das aplicações as quais estão em execução no docker.

2.21 MICRO SERVIÇO

O conceito de micro serviços se constitui em uma forma de simplificar o trabalho dos desenvolvedores de sistemas, em sua procura pela melhor forma de criar um produto a seus clientes. Os micros serviços podem ser aplicados em uma estrutura de negócio, visando aprimorar as atividades, gerando uma maior flexibilidade atuando na divisão de processos e fluxo de forma rápida e integrando informações, o que faz com que esta arquitetura crie uma estrutura de software como conjunto de serviços diretamente ligados.

Um Micro serviço pode ser aplicado em diversas situações, as mais comuns são utilizando Web API, já que esse tipo de arquitetura funciona muito bem no sentido de enviar informações onde é dividida as requisições por esse padrão de projeto.

> Ao mesmo tempo, um micro serviço deve ser considerado de tamanho reduzido. Para que seja possível realizar a alteração arquitetônica desejada, serão avaliadas diferentes abordagens para a conversão de uma arquitetura monolítica para uma arquitetura de micro serviços. (AMARAL, José Diogo Coelho, 2018).

Conforme Amaral, a estrutura do micro serviço deve conter sempre uma divisão de forma correta, para que todas as requisições que o sistema possua não seja sobrecarregado em um único serviço, ou no caso de uma Web API, um único local de busca e envio de informações. Micro serviços formam um par perfeito com o padrão de projetos de Gateway ao qual visa sempre enviar informações prontas para os clientes, evitando assim, várias requisições no servidor de dados.

2.22 PADRÃO DE DESENVOLVIMENTO GATEWAY

Esse padrão de desenvolvimento consiste em criar um serviço ao qual serve como intermediário das solicitações realizadas na Web API onde redirecionará as solicitações que ocorrem no servidor para seu devido local, tendo em vista que é um padrão voltado para micro serviços e cada serviço ficara muitas das vezes com padrão de requisição diferente, o gateway um local de endereço IP fixo na rede ao qual encaminha a requisição para o micro serviço correto.

> O API Gateway é uma camada intermediária por meio da qual a interface do utilizador pode interagir com os micros serviços. Este também fornece uma interface mais simples e simplifica o processo de consumir esses serviços. Fornece um nível diferente de granularidade para clientes diferentes, conforme necessário. (AMARAL, José Diogo Coelho, 2018).

Esta camada será fundamental neste projeto, devido, que conterá vários “end-points” (locais que serão proveniente de informações) e cada um deles será dividido em um micro serviço diferente, assim não ocorre um gargalo de chamadas no sistema, evitando congestionamento das informações.

2.23 OCELOT

O Ocelot é um Framework desenvolvido em dotnet para auxiliar na criação de Gateway, este framework funciona também em todas as aplicações que utilizem protocolos de comunicação HTTP. Existe também mais uma funcionalidade deste que possibilita de forma fácil a configuração de um sistema de *circut breaker* (Circuitos eletrônicos).

> O Ocelot recebe um objeto, validando a existência do pedido no ficheiro de configuração e posteriormente, quando acedido, a ferramenta cria um objeto *HttpRequestMessage*, que é usado para fazer uma solicitação para um serviço de recebimento de dados. Existe também um middleware que mapeia o HttpResponseMessage para o objeto HttpResponse e que é retornado ao cliente. (AMARAL, José Diogo Coelho, 2018).

Conforme Amaral descreveu, esta ferramenta de desenvolvimento serve como intermediário entre o cliente solicitante da requisição HTTP e os micros serviços do sistema, onde o usuário fara a requisição apenas em um ponto na internet, e essa requisição será redirecionada

internamente para seu devido micro serviço, ao qual irá prover as informações solicitadas pelo cliente.

Desta forma, não importando quantos serviços possua, basta acessar um único local por chamada HTTP, e informar o micro serviço ao qual se quer utilizar, e o Ocelot se encarregará de redirecionar a chamada e retornará os dados corretamente, ou passará as informações, de acordo com o tipo da requisição.

2.24 RABBITMQ

O RabbitMQ é um sistema de mensageria voltado para o desenvolvimento de micro serviços com sistemas de middleware, onde com ele é possível enviar mensagem de um serviço para outro sem nenhuma ligação entre os mesmos. De tal forma o mesmo gerencia filas de mensagens onde direciona cada mensagem para algum sistema que esteja “inscrito” na fila para receber essas mensagens.

Segundo o *RabbitMQ* (2016), o *RabbitMQ* foi desenvolvido na linguagem de programação Erlang, tendo seu desenvolvimento iniciado em 2006, e a primeira versão liberada foi em 2007 através de uma licença *Mozilla Public License* (MPL).

O *RabbitMQ* suporta o envio de mensagens através de filas e através de esquema de inscrição e publicação, desta forma, o *RabbitMQ* é composto por uma servidora, clientes para a comunicação com o serviço de mensagens, aos quais estão disponíveis nas mais diversas linguagens de programação como por exemplo: Java, C# e Python.

Existem diversos tipos de serviços de mensageria, porém, o *RabbitMQ* foi o escolhido para desenvolvimento nesse projeto, devido a sua praticidade e forma simples de utilização, tendo em vista que o mesmo será utilizado para funcionalidades simples no sistema.

2.25 AUTOMAPPER

É um framework que serve para mapeamento de forma automática de um tipo de valor para outro. Normalmente utilizado para modificar um objeto de origem e transforma-lo para um objeto de destino, dependendo da sua regra de negócios, tornando assim mais prática a conversão que anteriormente, seria feita de forma manual.

De acordo com a organização *AutoMapper*, esta ferramenta de trabalho *é* um mapeador de objeto-objeto. O mapeamento objeto-objeto funciona transformando um objeto de entrada de um tipo em um objeto de saída de um tipo diferente. O que torna o *AutoMapper* interessante é que ele fornece algumas convenções interessantes para descobrir como mapear o tipo A para o tipo B. Desde que o tipo B siga a convenção estabelecida do *AutoMapper*, é necessária uma configuração quase zero para mapear dois tipos. (AUTOMAPPER, ORG 2017).

Por ser um framework gratuito, o mesmo foi o escolhido para a atualização de mapeamentos automáticos para o sistema deste trabalho, o mesmo está localizado no servidor de frameworks da Microsoft, tornando-o assim mais seguro e confiável para sua utilização.

2.26 IDENTITY SERVER

É uma ferramenta para gerenciamento de login no sistema, onde é possível manter usuário conectados com tokens e um Sistema para atualização automática, com ele é possível diversos tipos de autenticação de usuários e de sistemas, sendo possível autorizar aplicativos a poderem utilizar determinado recurso ou não, dependendo de sua forma de conexão e permissões.

Baseado no *OpenID Connect*, e garante que diferentes tipos de aplicações, em diferentes tecnologias, possam fazer uso do servidor de identidade rodando o *IdentityServer*. Com ele pode-se oferecer informações de identidade através de *OAuth*, oferecer segurança para *APIs*, autenticação de aplicações móveis, etc.

O sistema de autenticação escolhido para o desenvolvimento nesse projeto, foi o sistema baseado em tokens, onde o usuário se conecta no servidor de identidade e recebe um código onde

em cada requisição na API de dados é enviado o token junto e validado em cada requisição, desta forma toda vez que o usuário solicitar algo, será validada a requisição, e não sendo necessário uma conexão mantida com servidor, evitando tráfego na rede do sistema.

2.27 GERENCIAMENTO DE REDES

Redes de computadores são fundamentalmente importantes na atualidade, tendo em vista que estamos conectados com a internet, que é nada além de uma enorme rede de computadores conectados por cabo e transmitindo informações constantemente, e o número de computadores não para de crescer.

“Foram vendidas 1,448 milhão de máquinas no intervalo analisado em 2019. Assim, foi gerado uma receita de R$ 4,1 bilhão, 12% de aumento em relação ao segundo trimestre de 2018.” [TECMUNDO, 2019]. Com base nesses dados, é notável o crescimento anual da venda de computadores no Brasil, e consequentemente no mundo, a tecnologia já é parte fundamental de nossas vidas.

Tais computadores são vendidos para as mais diversas finalidades, e uma delas é a utilização nas empresas, onde cada dia se faz necessário ter um maior controle no desenvolvimento de atividades de forma ágil, e para que isso aconteça, é preciso que todos os setores de uma companhia estejam interligados

Exatamente nessa necessidade que entra o conceito de redes, onde faz-se uma ligação de diversos pontos isolados ao qual no final compartilham seus dados independentemente da localização física que os compõe. “A rede também aumenta a confiabilidade do sistema, pois tem fontes alternativas de fornecimento” (TANENBAUM, 1997, P. 3). Desta forma, uma rede de computadores nada mais é que um mecanismo de extrema importância em um setor ao qual trabalhe com informações, pois facilita a comunicação de diferentes pontos da empresa compartilhando seus dados.

3 DESCRIÇÃO DA IMPLEMENTAÇÃO.

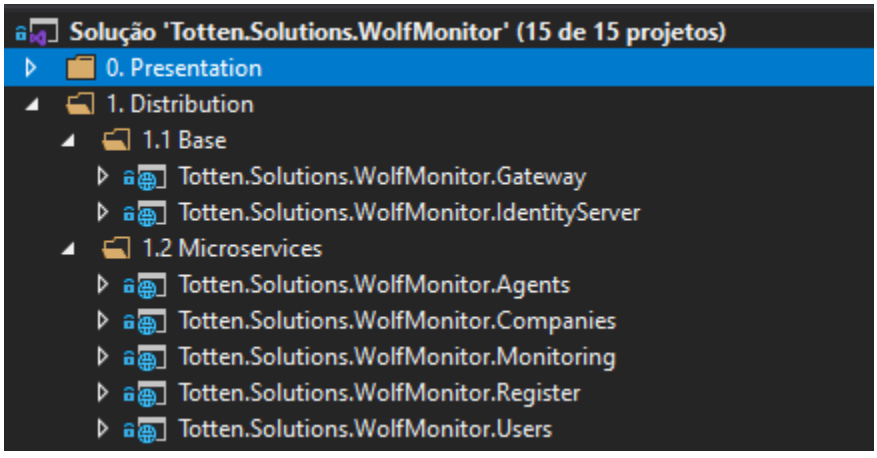
3.1 WEB API

A implementação do projeto se inicia com o desenvolvimento da Web API, onde tanto o Serviço do Windows, quando as interfaces gráficas utilizaram para o envio e recebimento de informações. Por ser um sistema que utiliza a interoperabilidade, toda comunicação que será feita por texto em formato JSON que trafega pela internet, e ambos faram requisições para obter e enviar essas informações.

Foi escolhida a linguagem ASP.NET para criação da Web API, pela praticidade que o .NET oferece no desenvolvimento de suas aplicações e pela vasta gama de conteúdo na internet ensinando e auxiliando no desenvolvimento de uma API robusta que suporte a quantidade de requisições que o sistema possuirá ao ser finalizado.

Utilizando os padrões de Micro serviços para criação da API e end-points devido que todo o sistema está nas conformidades de API Gateway, onde a mesma é desenvolvida pensando no front-end, para que assim não seja realizada consultas desnecessárias e diminua a quantidade de requisições no servidor e até mesmo uma internet com menor velocidade conseguirá acessar os dados.

Figura 1: Projetos de micro serviços do Sistema.



FONTE: Próprio autor

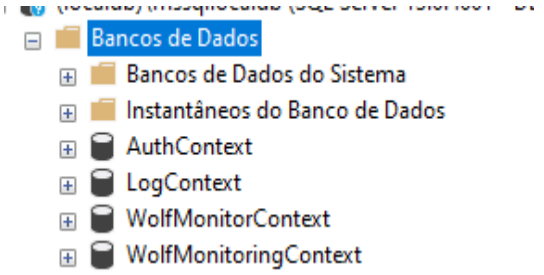
Cada projeto listado na figura 1, desde a pasta 1.1 a pasta 1.2 são micro serviços criados para servirem de provedores e receptores de informações para o sistema, cada um atuará em uma porta no servidor, fazendo com que o tráfego de informações não acarretem em congestionamento ou demora no sistema, ao todo serão sete projetos para este intuito.

3.2 BANCO DE DADOS

O Banco de dados escolhido para este projeto, foi o banco de dados do SQL Server da Microsoft, ao qual possui versões gratuitas para desenvolvimento e estudo, o SQL Server é um banco de dados relacional produzido e com suporte pela Microsoft, um banco de dados rápido e simples de utilização.

Como o presente trabalho utiliza de micros serviços, faz-se válido uma diferenciação no banco de dados, o mesmo foi separado em quatro partes, para que dessa forma o fluxo de dados seja rápido e não ocorra lentidão no sistema nem congestionamento no fluxo de dados.

Figura 2: Bancos de dados utilizados no Sistema



FONTE: Próprio autor

O primeiro é o banco de dados de Autenticação, onde é salvo todos os dados referentes aos usuários do sistema, o segundo é responsável por logs do sistema, onde todo o log da web api é salvo como criação de dados, alterações e exclusões de informações, o terceiro fica a cargo de salvar os dados referentes a empresas e os *agents* (servidores do cliente) cadastrados e o último é o encarregado por salvar todos os itens que serão monitorados pelo *agents* no servidor do cliente, seja este, serviço ou arquivos.

3.3 DESENVOLVIMENTO DESKTOP

O Desenvolvimento para desktop foi feito para a visualização das informações enviadas pelo serviço do sistema Windows/Linux para a Web Api de dados, todo o sistema desktop foi desenvolvido utilizando o *Microsoft Visual Studio Community* é uma ferramenta de desenvolvimento voltada para linguagem C#.

C# é uma linguagem bem popular no mercado por sua forma de programar em alto nível e principalmente pela utilização do Visual Studio que torna a vida do programador mais simples por suas inúmeras ferramentas para auxilio ao desenvolver um sistema.

O sistema *desktop* se conecta com o servidor de informações de dados passando por autenticação no *Identity Server*, após ser autorizado o mesmo pode visualizar todos os dados que seu usuário tenha permissão para acessar.

Esses dados são obtidos pelo aplicativo *desktop* com base em requisições HTTP onde a Web Api retorna esses dados em um JSON, para que o programa possa tratar essas informações e mostrar na tela para o usuário, por exemplo, ao solicitar a lista de serviços monitorados o JSON é recebido conforme a figura 3 abaixo.

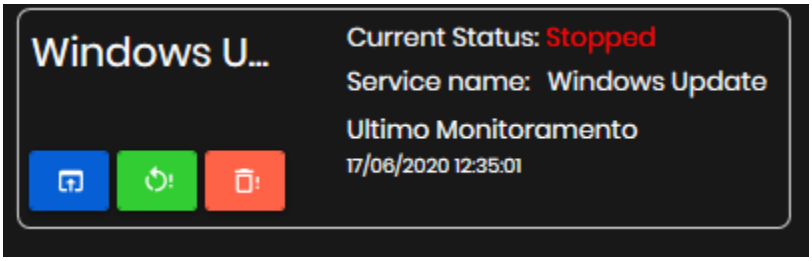
Figura 3: Exemplo de resposta JSON na solicitação de serviços monitorados


```
{
    "id": "ca165926-a490-477e-bbcc-08d80d918359",
    "displayName": "Windows Update",
    "name": "wuauserv",
    "value": "Running",
    "aboutCurrentValue": "ChangeContainsProfile",
    "monitoredAt": "10/06/2020 20:56:28"
},
```


FONTE: Próprio autor

Após receber essas informações em formato JSON o aplicativo faz um tratamento e transforma esses dados em algo mais visual para que o usuário possa verificar essas informações de forma mais simples, conforme a figura 4.

Figura 4: Listagem no sistema desktop dos serviços monitorados



FONTE: Próprio autor

O Programa de visualização de informações trabalha desta forma com todos os dados apresentados na tela, desde as listagens de servidores, usuários, monitoramento de arquivos e autenticação; desta forma a interoperabilidade não é afetada, ou seja, caso haja necessidade futura de desenvolver um sistema para internet, o mesmo poderá buscar as informações no mesmo servidor de dados.

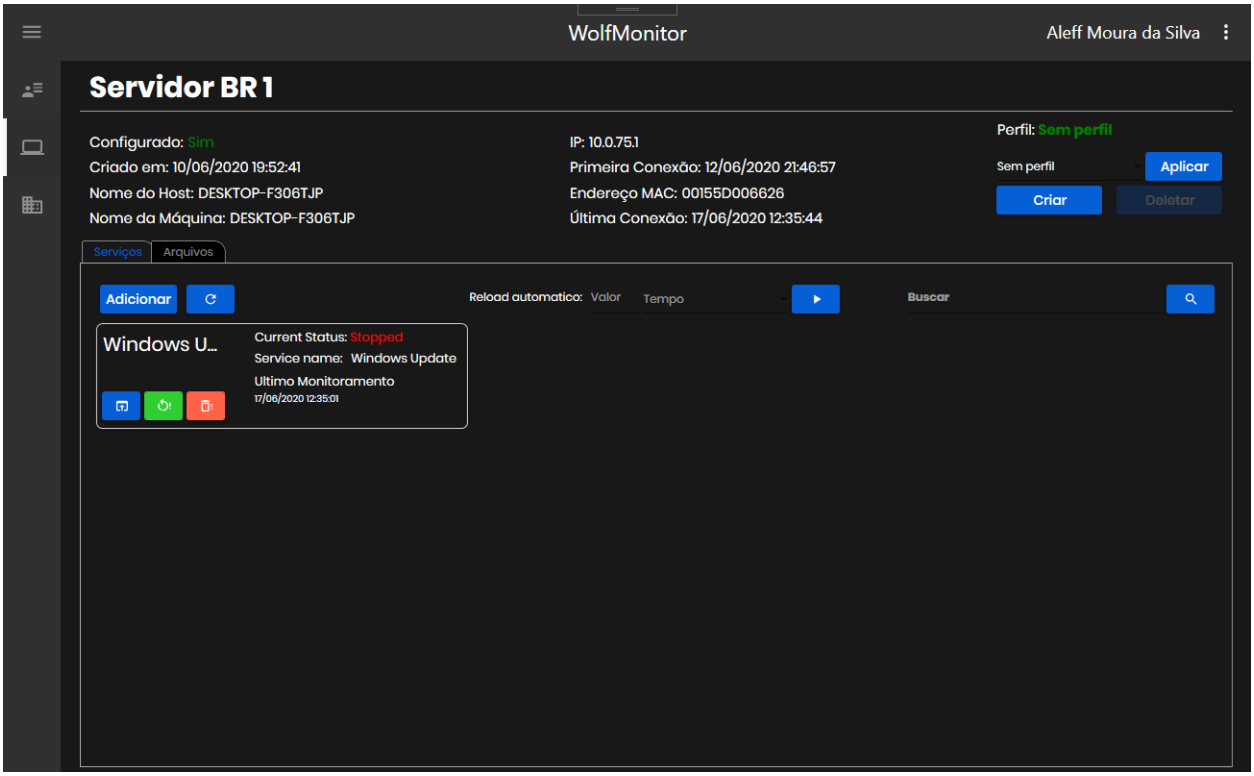
Para utilização do sistema, após obter uma conta de login, o usuário administrador da empresa poderá efetuar cadastros de *agents* aos quais ficarão responsáveis pelo monitoramento, ao ser cadastrado o mesmo aparecerá em uma listagem na tela conforme figura 5.

Figura 5: Resumo do Agent cadastrado no sistema

FONTE: Próprio autor

Após o cadastro o usuário poderá acessar esse *agent* abrindo seu detalhamento, nos detalhes terá as opções de efetuar cadastros, visualizar serviços e arquivos a serem monitorados, O *agent* deverá ser não precisa ser definido como Windows ou Linux ao ser instalado ele faz a diferenciação em tempo de execução no sistema.

Figura 6: Detalhamento do Agent cadastrado



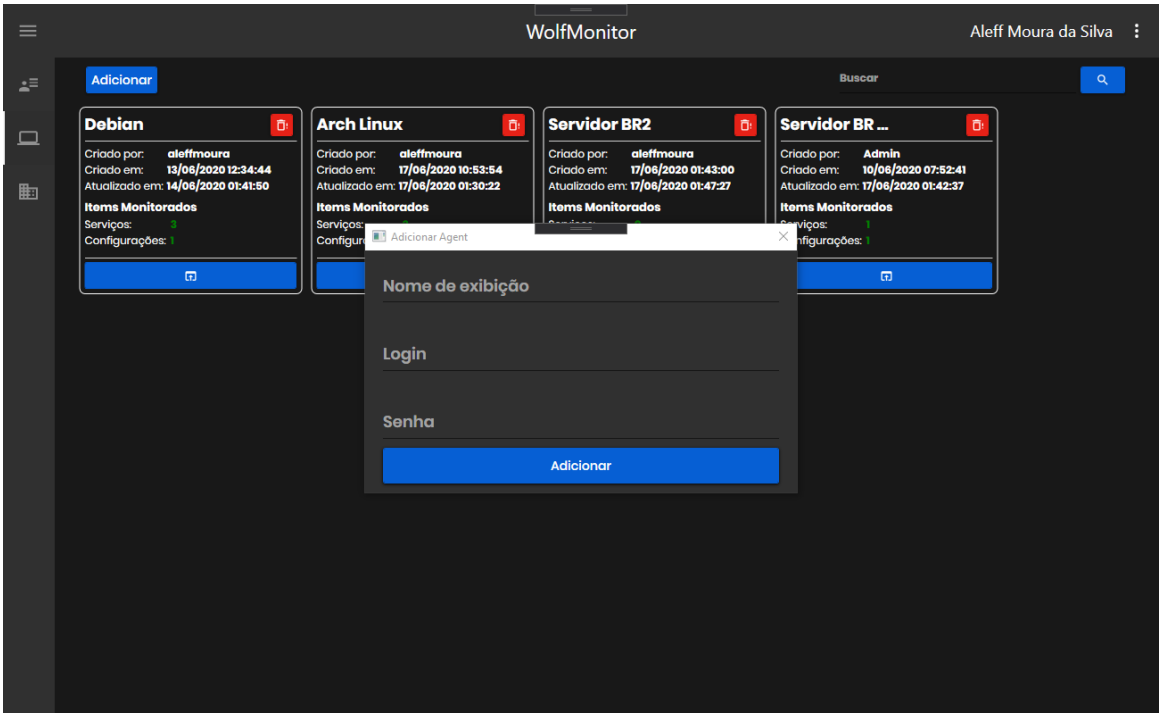
FONTE: Próprio autor

O usuário poderá criar perfis de *agent*, ao efetuar este procedimento, é salvo o estado atual dos serviços e arquivos monitorados, ao ser aplicado um perfil, o serviço sempre forçará os itens a terem seus valores conforme o perfil selecionado, desta maneira por exemplo, caso um serviço crucial pare sua execução em algum momento, o monitoramento faz com que o mesmo fique em execução logo em seguida.

3.4 Agent – Serviço do Linux/Windows

O Serviço tratado no sistema como um *agent*, após ser criado na tela do *front-end* deverá ser instalado no servidor do cliente como um serviço, seja um servidor Windows ou Linux, após isso deve ser modificado suas configurações para conexão com o *agent* criado, com mesmo usuário e senha ao qual foi colocado no sistema, conforme figura 7.

Figura 7: Cadastro de agent



FONTE: Próprio autor

Após esse cadastro o usuário solicitará os arquivos necessários para instalação do *agent* e deverá configurá-lo para utilização, estes arquivos serão as dll's (arquivos de sistema) e executável do programa de monitoramento, ao qual ficará responsável por verificar os dados de todos os itens cadastrados, sempre buscando as informações no servidor de dados da Web API.

Figura 8: Arquivo de configuração do agent

FONTE: Próprio autor

A figura 8 é o arquivo ao qual o usuário deverá atualizar com suas informações, o arquivo é um JSON que contém os dados do *agent*, a empresa do mesmo e algumas informações de pastas para guardar valores de log do sistema em caso de erros, a figura 9 mostra o conteúdo do arquivo com um *agent* configurado.

Figura 9: JSON de configurações do agent

FONTE: Próprio autor

Após essas configurações o usuário poderá iniciar o *agent*, e logo ao ser executado o mesmo fará login no sistema e iniciará os procedimentos de monitoramento, enviando informações diretamente para a web api, onde o usuário buscará essas informações e obtendo dados quase que em tempo real.

4 CONCLUSÃO

Este trabalho apresentou o desenvolvimento de um sistema completo, contendo 3 subsistemas onde o intuito do mesmo é capturar informações de um servidor seja ele Windows ou Linux a fim de auxiliar empresas em seus trabalhos do dia-a-dia de uma forma simples para observar os dados de seus servidores. O estudo busca ser uma solução para empresas que possuem muitos servidores e diversos serviços instalados no mesmo; onde fazer migrações de sistemas e aplicar correções de bugs leva bastante tempo, devido a quantidade de serviços a serem desligados/ligados.

Apesar do presente sistema ser voltado a monitoramento apenas de arquivos e serviços, o mesmo abrange outras possibilidades como: banco de dados, serviços de IIS (Provedor de dados Windows), entre toda e qualquer informação que seja fundamental para o funcionamento correto dos servidores, para que assim seja evitado conexões e demoras para obter determinada informação. O presente estudo visa mostrar as empresas que suas atividades corriqueiras do cotidiano podem ser melhoradas fazendo com que não fique exaustivo efetuar os mesmos procedimentos manuais, sendo que é possível otimizá-los.

A utilização deste sistema irá promover uma melhoria significativa na produção das empresas, para que sejam feitos procedimentos de forma rápida e com qualidade, garantindo a funcionabilidade correta dos servidores, mesmo após um serviço de migração ao qual tenha sido necessário o desligamento de todos os serviços.

5 REFERÊNCIAS BIBLIOGRÁFICAS

MICROSOFT, **Introdução a aplicativos do Serviço Windows 2017**. Disponível em: <https://docs.microsoft.com/pt-br/dotnet/framework/windows-services/introduction-to-windows-service-applications>. Acesso em: 23 fev. 2020.

TANENBAUM, Andrew S. Redes de computadores. 3. Ed. Rio de Janeiro: Campus, 1997.

Wasson Mike, **Introdução ao ASP.NET Web API 2** Disponível em: <https://docs.microsoft.com/pt-br/aspnet/web-api/overview/getting-started-with-aspnet-web-api/tutorial-your-first-web-api>. Acesso em: 23 fev. 2020.

AMBLER, Scott W. Analise e projeto orientado a objeto. Rio de Janeiro: Infobook, 1998

KORTH, H.F. e SILBERSCHATZ, A.; Sistemas de Bancos de Dados, Makron Books, 2a. edição revisada, 1994.

SQL Server. Disponível em: <https://www.infoescola.com/informatica/sql-server/>. Acesso em: 23 fev. 2020

W3SCHOOLS. **What is JSON?** Disponível em <https://www.w3schools.com/whatis/whatis_json.asp/>. Acesso em: 23 fev. 2020.

Visão Geral Entity Framework. Disponível em: <https://docs.microsoft.com/pt-br/dotnet/framework/data/adonet/ef/overview >. Acesso em: 23 fev. 2020.

DDD – Domain-Driven Design com .NET. Disponível em: <https://www.devmedia.com.br/ddd-domain-driven-design-com-net/14416>. Acesso em: 23 fev. 2020.

MOTOYAMA, S. (org) **Prelúdio para uma história**: ciência e tecnologia no Brasil. São Paulo: EDUSP, 2004.

PEREZ, C.; FREEMAN, C. Structural crises of adjustment, business cycles and investment behavior, In: DOSI, G. et al. (Eds.). Technical change and economic theory. Londres: Pinter Publishers, 1988. p. 38-66.

Richard Batista Silveira, **HISTÓRIA DO MICROSOFT® WINDOWS®.** Disponível em: <http://ccp.uenp.edu.br/centros/d_matematica/jcoelho/txt/ap-jc01-windows.pdf>. Acesso em: 04 mar. 2020.

INFOESCOLA, **História Windows.** Disponível em: <https://www.infoescola.com/informatica/historia-do-windows/ >. Acesso em: 04 mar. 2020.

INFOWESTER, **História do Linux.** Disponível em: <https://www.infowester.com/historia_linux.php>. Acesso em: 04 mar. 2020.

TECMUNDO, **Mercado de computadores tem leve crescimento em 2019, aponta IDC**. Disponível em: <https://www.tecmundo.com.br/mercado/146050-mercado-computadores-tem-leve-crescimento-2019-aponta-idc.htm>. Acesso em: 04 mar. 2020

HOSTINGER, **Como Gerenciar e Listar Serviços no Linux.** Disponível em: <https://www.hostinger.com.br/tutoriais/listar-servicos-linux/>. Acesso em: 04 mar. 2020.

MICROSOFT, **Guia do .NET CORE.** Disponível em: <https://docs.microsoft.com/pt-br/dotnet/core/>. Acesso em: 04 mar. 2020.

MICROSOFT, **WINDOWS PRESENTATION FOUNDATION.** Disponível em: <https://docs.microsoft.com/pt-br/dotnet/framework/wpf//>. Acesso em: 04 mar. 2020.

SANTOS, Luiz C. **Firewall Linux IPTABLES**. Rio de Janeiro, 2004, Disponível em: <http://www.clubedasredes.eti.br/rede0023.htm>. Acesso em: 11 de mar. 2020.

SKYONE, **A importância da nuvem para o crescimento de pequenas e médias empresas.** Disponível em: <https://skyone.solutions/pb/importancia-da-nuvem/>. Acesso em: 11 de mar. 2020.

Amaral, José Diogo Coelho, **A evolução das arquiteturas monolíticas para as arquiteturas baseadas em micro serviços.** Disponível em: <https://recipp.ipp.pt/handle/10400.22/11920/>. Acesso em: 11 de mar. 2020.

Getting Started Guide — AutoMapper documentation. Disponível: <http://docs.automapper.org/en/stable/Getting-started.html>. Acesso em: 6-abr-2020.

RABBITMQ (n.d.) 2016 Disponível em: <http://www.rabbitmq.com/documentation.html>. Acesso em: 29 mar. 2016

Gomes, Pedro César Tabaldi, **AFINAL, O QUE É DOCKER?** Disponível em: <https://recipp.ipp.pt/handle/10400.22/11920/>. Acesso em: 11 de mar. 2020.