|  |  |
| --- | --- |
| CURSO DE CIÊNCIA DA COMPUTAÇÃO – TCC | |
| (   ) PRÉ-PROJETO     (  X   ) PROJETO | ANO/SEMESTRE: 2021/2 |

MIDDLEWARE ESCALÁVEL E RESLIENTE PARA COMUNICAÇÃO ENTRE APLICAÇÕES WEB MÓVEIS EM AMBIENTES DE COMPUTAÇÃO DISTRIBUÍDA

Ariel Adonai Souza

Prof. Francisco Adell Péricas – Orientador

# Introdução

O mundo está cada vez mais rodeado por aplicativos móveis. Estes aplicativos estão se tornando a principal ferramenta de comunicação das pessoas com a Internet, conforme citado por Oehlman e Blanc (2012). Diante desse cenário, é natural que surja uma demanda crescente de ferramentas e tecnologias que embarquem os seus softwares corporativos nos dispositivos móveis. Oehlman e Blanc (2012) também contam que em empresas com aplicações Web, os desenvolvedores reconstroem grandes porções dessas aplicações Web dentro dos aplicativos móveis para cada um dos diferentes dispositivos. “Para algumas companhias que constroem aplicativos móveis, esta é uma metodologia aceitável. É, contudo, uma das menos sustentáveis a longo prazo.” (OEHLMAN; BLANC, 2012, p. 9). Os autores apontam para um futuro em que a demanda por desenvolvedores seria muito grande para conseguir manter todas essas aplicações por conta da grande variedade de dispositivos móveis disponíveis hoje no mercado.

Uma solução mais viável é desenvolver uma aplicação Web para dispositivos móveis. Segundo Oehlman e Blanc (2012), aplicações Web são uma forma de escrever aplicações que, quando feito da forma correta, permite adaptar as aplicações para dispositivos móveis sem a necessidade de reescrever muito código. “Um aplicativo Web para dispositivos móveis é um aplicativo construído com as tecnologias Web centradas no cliente em HTML, CSS e JavaScript, e é especificamente projetado para os dispositivos móveis” (OEHLMAN; BLANC, 2012, p. 9).

Segundo Tanenbaum e Steen (2008), em meados de 1980 ocorreram dois avanços tecnológicos que revolucionariam a computação até os dias atuais. O primeiro foi a evolução de microcontroladores de maior capacidade, que com o passar do tempo chegavam à capacidade de processamento de um *mainframe* por um preço muito menor. O segundo avanço foi o desenvolvimento das redes de computadores e o surgimento das redes locais, as *Local Area Networks* (LANs). As LANs permitiram que centenas de computadores próximos pudessem trocar informações na velocidade de alguns microssegundos. O resultado do surgimento destas duas tecnologias foi o surgimento de um modelo de computação de altíssima capacidade de processamento conectado em uma rede de alta velocidade, os sistemas distribuídos.

A definição de um sistema distribuído é “um conjunto de computadores independentes que se apresenta a seus usuários como um sistema único e coerente.” (TANENBAUM; STEEN, 2008, p. 1). Eles ressaltam aspectos importantes: um sistema distribuído consiste em computadores autônomos e usuários (programas, aplicações ou pessoas) que devem achar que tratam com um sistema único.

Segundo Tanenbaum e Steen (2008), em um ambiente de computação distribuída, por conveniência, é utilizado um modelo de comunicação assíncrona e existem diversos sistemas por trás de uma única aplicação. Diante desse cenário, é possível imaginar que seja complexo uma aplicação Web para dispositivos móveis, que está fora desta rede, se comunicar com algum serviço interno.

Por mais que à primeira vista pareça complexo estabelecer a comunicação entre o dispositivo móvel e os serviços, existe um tipo de software que soluciona esta situação: os *middlewares*. Um *middleware* funciona “[...] de forma essencial como uma camada oculta de tradução, o middleware permite a comunicação e o gerenciamento de dados para aplicativos distribuídos.” (AZURE, 2021). Com a utilização de um *middleware*, a aplicação Web se comunica de forma transparente com os serviços.

Diante do cenário apresentado, levando em consideração o aumento do uso de dispositivos móveis, a facilidade e conveniência em implementar aplicações Web para dispositivos móveis, como seria possível implementar um *middleware* que consiga lidar com alta volumetria, que seja escalável e que garanta a entrega das mensagens?

## OBJETIVOS

O objetivo deste trabalho é apresentar um *middleware* que seja escalável e resiliente para comunicação cliente/servidor entre aplicações Web para dispositivos móveis e serviços em um ambiente de computação distribuída.

Os objetivos específicos deste trabalho são desenvolver um *middleware* para uma arquitetura distribuída que:

1. seja escalável;
2. seja resiliente na entrega de mensagens, ou seja, a mensagem deverá ser enviada até mesmo se o cliente estiver incomunicável no momento do envio (a entrega da mensagem ocorrerá quando o cliente se conectar novamente no *middleware)*;
3. atenda aos protocolos disponíveis em aplicações Web para comunicação;
4. atenda a um contexto de computação distribuída, ou seja, deve abstrair a comunicação das aplicações Web com os diversos serviços que possam existir.

# trabalhos correlatos

Foi encontrado um trabalho que se assemelha bastante com o objetivo deste estudo, o trabalho desenvolvido por Fernandes (2013). Os demais trabalhos não possuem uma relação direta com o objetivo deste estudo, porém, possuem um ou mais aspectos do objetivo deste estudo. O primeiro trabalho foi desenvolvido por Fernandes (2013) e descreve um *framework* para aplicações Web *Real Time*. O segundo trabalho foi desenvolvido por Souza (2016) e é um comparativo de APIs de sockets. O terceiro trabalho é a tese de mestrado desenvolvida por Silva (2017) que apresenta a plataforma EcoCIT, uma plataforma de *middleware* escalável que provê suporte para a integração de dispositivos de IoT à internet.

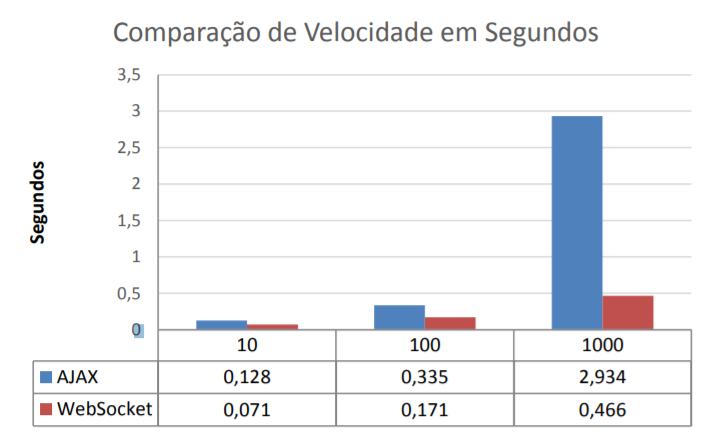
## Desenvolvimento de uma Framework Real Time Web para HTML5

Fernandes (2013) desenvolveu um *framework* para aplicações Web *Real Time*. O objetivo do trabalho é desenvolver o *framework* respondendo algumas perguntas voltadas a entender se é um paradigma a ser considerado na implementação de soluções de internet, como utilizar as tecnologias do momento para melhorar a comunicação de aplicações utilizando paradigma de *publish/subscribe* e avaliar qual a tecnologia mais adequada na utilização de *real time web*.

A metodologia do trabalho constitui em fazer um levantamento da literatura sobre *real time web*, pesquisar trabalhos sobre *publish/subscribe*, estudar os pontos fortes e fracos, levantar os requisitos, definir um *framework* que implemente o paradigma *publish/subscribe*, desenvolver as aplicações para testes e avaliar os resultados.

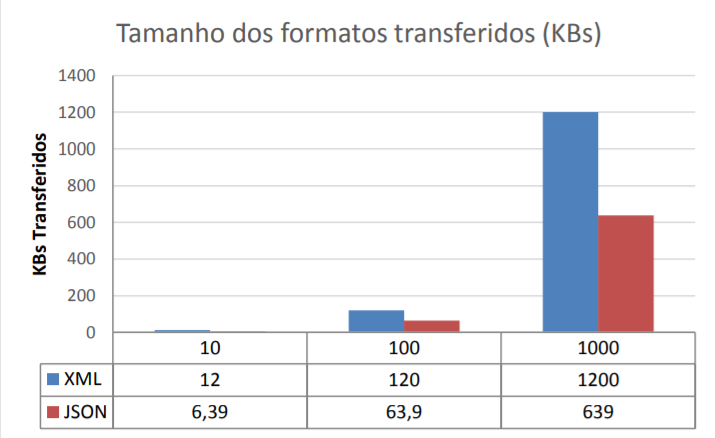
O trabalho comparou algumas tecnologias para descobrir quais performavam melhor dentro do contexto dos testes realizados. Foram realizados comparativos de velocidade entre AJAX e WebSocket, tamanho dos dados em XML e JSON e tamanho dos *headers* utilizando AJAX e WebSocket.

Figura : Comparação de velocidade em Segundos



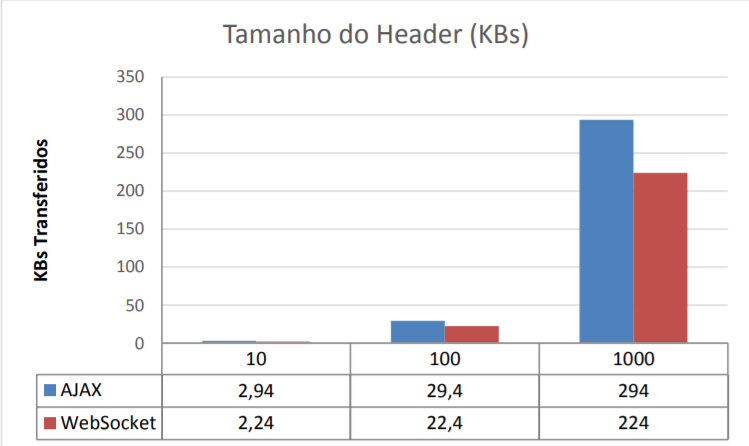
Fonte: Fernandes (2013).

Figura 2: Comparação de tamanho dos formatos transferidos (KBs)



Fonte: Fernandes (2013).

Figura 3: Comparação de tamanho dos headers (KBs)



Fonte: Fernandes (2013).

O resultado do trabalho concluiu que o protocolo WebSocket obteve um melhor resultado em todos os aspectos testados. Aponta também que “[...] WebSockets e o paradigma publisher/subscriber são as tecnologias mais adequadas na implementação de soluções que necessitem de integrar dados em tempo real. Soluções que podem ser bastante diversificadas como já foram referidas: finanças, apostas desportivas, dados, ou mesmo o Big Data.” (FERNANDES, 2013).

## ANÁLISE DE DESEMPENHO DE APIS DE SOCKETS

Souza (2016) desenvolveu um estudo comparativo de APIs *socket* visando medir o desempenho. O trabalho comparou as seguintes APIs: ZeroMQ, NanoMSG e Berkeley Sockets.

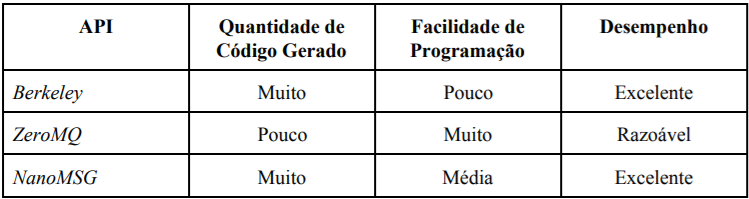
O ZeroMQ foi projetado para facilitar o desenvolvimento de aplicações distribuídas, uma API concebida para ter o mínimo de complexidade. Oferece quatro tecnologias para transporte: TCP, *MULTICAST*, *Instructions per cycle* (IPC) e *Inter-Process Communication* (INPROC).

O NanoMSG é uma concorrente do ZeroMQ. Foi concebida com a intenção de aumentar a rapidez, prover estabilidade da camada de rede e facilitar o desenvolvimento. O NanoMSG provê todos os padrões do ZeroMQ e provê também o uso do protocolo WebSocket.

Berkeley Sockets foram desenvolvidos na década de 1980 na *University of California at Berkeley*. Suporta somente TCP e UDP, e é uma API de mais baixo nível se comparada as demais citadas anteriormente.

Souza (2016) escolheu como métricas de avaliação a vazão e latência. Vazão é a quantidade de mensagens lidas em um determinado tempo. Latência é o tempo gasto para leitura de cada mensagem. Souza (2016) traz um comparativo entre as APIs e expõe alguns pontos relevantes: quantidade de linhas de código que são necessárias para realizar as operações de criar um socket, conectar, enviar dados, receber dados e fechar a conexão; qual a facilidade da programação; qual o desempenho com a utilização de cada uma das aplicações.

Figura 4: Comparativo das APIs



Fonte: Souza (2016).

A conclusão do trabalho, após a análise das métricas obtidas, foi que a NanoMSG obteve o melhor desempenho para os cenários dos experimentos. Souza (2016) enfatiza que esse trabalho pode ser usado como referência pela comunidade para escolha de uma das opções citadas no trabalho.

## EcoCIT: uma plataforma escalável para desenvolvimento de aplicações de IoT

Silva (2017) conta que desenvolver aplicações IoT, à primeira vista, parece ser uma tarefa fácil. Ele estimou que no ano de 2020 haveria mais de 200 bilhões de dispositivos IoT conectados na Internet. Comportar tantos dispositivos conectados ao mesmo tempo, bem como armazenar e processar todos esses dados é uma tarefa difícil. Outro fator que dificulta é que diferentes dispositivos IoT utilizam protocolos diferentes de comunicação. Diante destes cenários, muitas aplicações têm surgido com o propósito de abstrair para os desenvolvedores a utilização dos diferentes protocolos e dispositivos. Estas aplicações têm como objetivo entregar uma interface padronizada para acesso aos dispositivos. O trabalho de Silva (2017) teve como objetivo apresentar “a plataforma EcoCIT, uma plataforma de *middleware* escalável que provê suporte para a integração de dispositivos IoT com requisitos de escalabilidade através do uso de serviços computacionais providos sob demanda por plataformas de computação em nuvem” (SILVA, 2017, p. 8).

Silva (2017) conta que o EcoCIT é uma evolução de uma outra plataforma chamada EcoDiF. O EcoDiF “[...] é uma plataforma de *middleware* que integra dispositivos de IoT e os conecta à internet, fornecendo funcionalidades de controle, visualização e armazenamento de dados em tempo real.” (SILVA, 2017, p. 56). Apesar de diversas funcionalidades e facilidades que a EcoDiF proporciona, ela possui algumas limitações como a escalabilidade e capacidade de gerenciar grandes volumes de dados.

A EcoCIT surge com o objetivo de solucionar as limitações que a EcoDiF possui. A arquitetura é composta de diversos componentes distribuídos para permitir que a EcoCIT possa fazer uso da elasticidade que os ambientes de computação em nuvem podem prover. A arquitetura permite que a aplicação seja executada em máquinas virtuais que trabalham de maneira agrupada e possui um balanceador de carga para distribuir a demanda entre as máquinas virtuais. A aplicação também faz uso de outros recursos escaláveis e distribuídos como os bancos de dados e outras estruturas de armazenamento. Por se tratar de uma evolução de outra ferramenta, o trabalho descreve diversas outras alterações realizadas em muitos módulos da plataforma para que fosse possível tornar a EcoDiF uma plataforma escalável.

Silva (2017) realizou três experimentos em seu trabalho. O primeiro teve como objetivo avaliar a substituição do sistema gerenciador de banco de dados utilizado no gerenciamento de *feeds*. Os *feeds* são basicamente os dados que os dispositivos enviam para o servidor. Os resultados mostraram que a substituição do banco de dados contribuiu para a melhor performance do processo. O autor aponta que o experimento apresentou uma performance ligeiramente menor para a operação de consulta de *feeds*, porém, o novo banco de dados era muito mais escalável, sendo capaz de suportar cargas de trabalho superiores.

O segundo experimento teve como objetivo comparar o tempo de resposta entre a EcoDiF e a EcoCIT em cenários similares, com o mesmo hardware. O experimento foi realizado acrescendo a quantidade de requisições simultâneas em 100 por teste. A EcoDiF foi capaz de aguentar no máximo 400 requisições simultâneas, atingindo um tempo médio de resposta de 4466ms. Já a EcoCIT, foi capaz de aguentar 2800 requisições simultâneas e atingiu um tempo médio de resposta de 517ms. A Figura 2 é um gráfico que apresenta o resultado dos testes realizados.

Figura 2: Resultado dos testes realizados



Fonte: Silva (2017).

O terceiro experimento teve como objetivo avaliar a elasticidade da plataforma. Ou seja, estressar a plataforma ao ponto de disparar os mecanismos de elasticidade, fazendo com que mais instâncias fossem criadas quando a demanda aumentasse e destruindo as instâncias quando a demanda diminuísse. “Esse experimento mostrou que [...] é capaz de monitorar a carga de trabalho nas VMs e tornar as medidas necessárias para evitar que as VMs se tornem sobrecarregadas ou ociosas, ajustando a demanda por recursos computacionais ao uso” (SILVA, 2017, p. 118).

# proposta DO MIDDLEWARE

Esta seção tem como objetivo justificar este trabalho, apresentar os seus motivadores e a metodologia de desenvolvimento que será utilizada.

## JUSTIFICATIVA

Com o aumento do uso dos dispositivos móveis e a quantidade cada vez maior de aplicativos, é natural que alguns problemas comecem a aparecer. Um destes problemas é a comunicação entre os dispositivos e os servidores, principalmente quando se fala de um contexto de computação distribuída.

O Quadro 1 apresenta algumas características que mostram que este trabalho pode contribuir entregando um *middleware* para comunicação de aplicações Web para dispositivos móveis e serviços em um ambiente distribuído. Durante a pesquisa e nos trabalhos correlatos selecionados, nenhum deles apresenta uma proposta de arquitetura de comunicação de aplicações Web para dispositivos móveis com servidores que tenham o objetivo de ser escaláveis e resilientes. Um cenário que necessite de uma solução assim pode ser encontrado facilmente num contexto de dispositivos móveis e servidores utilizando a arquitetura de computação distribuída. Este trabalho pode contribuir com um modelo que simplifique a comunicação, melhore a estabilidade e tire a responsabilidade dos serviços de se preocuparem com a entrega de mensagens para aplicações móveis.

Quadro - Comparativo dos trabalhos correlatos

|  |  |  |  |
| --- | --- | --- | --- |
| Trabalhos Correlatos  Características | Fernandes (2013) | Souza (2016) | Silva (2017) |
| Escalável |  |  | X |
| Resiliente | X | X |  |
| Utilização de alguma ferramenta de mensageria |  |  | X |
| Voltado para comunicação Web | X |  |  |

Fonte: elaborado pelo autor.

No Quadro 1 foram apresentadas algumas características que estre trabalho se propõe a atender em comparação aos trabalhos correlatos. Todos os trabalhos deixam margem para que seja buscada uma solução que possa atender a todas as características citadas. O trabalho desenvolvido por Fernandes (2013) apresenta um *framework* para aplicações *real time web*, que se assemelha a este trabalho por ser voltado para a Web. Entretanto, o trabalho não é apresentado como uma solução escalável e para um contexto distribuído. Já o trabalho de Souza (2016) comparou diversas APIs de sockets, porém, ele trabalhou em uma camada mais baixo nível e não com uma comunicação Web. O trabalho dele pode ser utilizado para a execução deste trabalho na implementação dos sockets de comunicação. O trabalho desenvolvido por Silva (2017) apresenta uma evolução de uma plataforma para abstrair o gerenciamento de dispositivos IoT que atendam o requisito de ser escalável. Porém, não apresenta uma solução resiliente, de comunicação assíncrona e para dispositivos móveis.

## REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

O *middleware* da arquitetura proposta neste trabalho deverá:

1. ser possível comportar uma volumetria cada vez maior (escalável) (RF);
2. armazenar as mensagens até conseguir entregá-las ao destino (RF);
3. garantir a entrega das mensagens pendentes (ainda não enviadas para o destinatário) assim que os dispositivos conectarem novamente (RF);
4. possibilitar que os dispositivos móveis enviem mensagens quando o serviço estiver fora do ar (RF);
5. possibilitar que os serviços enviem mensagens quando o dispositivo móvel estiver fora do ar (RF);
6. fazer com que todas as mensagens sejam assíncronas, ou seja, a aplicação que enviar não deve esperar um retorno de confirmação da outra aplicação (RF);
7. utilizar um banco de dados não relacional para armazenar as mensagens temporariamente (Requisito Não Funcional - RNF);
8. utilizar uma aplicação de mensageria para troca de mensagens internas utilizando o protocolo AMQT (RNF);
9. possibilitar a comunicação dos dispositivos móveis através do protocolo WebSocket (RNF);
10. possibilitar a comunicação dos serviços através do protocolo AMQT(RNF).

## METODOLOGIA

Tendo base no problema, objetivos e requisitos apresentados, será adotada uma estratégia para montar uma arquitetura que possa resolver este problema. O trabalho será desenvolvido observando as seguintes etapas:

1. estudar as tecnologias e ferramentas disponíveis no mercado;
2. especificar um diagrama com os componentes internos do *middleware*;
3. desenvolver o *middleware* proposto;
4. desenvolver uma aplicação Web para dispositivos móveis e dois serviços para testar o *middleware*;
5. testar a aplicação e avaliar os resultados;
6. escrever os resultados do trabalho.

As etapas serão realizadas nos períodos relacionados no Quadro 2.

Quadro - Cronograma

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 2022 | | | | | | | | | |
|  | fev. | | mar. | | abr. | | maio | | jun. | |
| etapas / quinzenas | 1 | 2 | 1 | 2 | 1 | 2 | 1 | 2 | 1 | 2 |
| estudar as tecnologias e ferramentas disponíveis no mercado | X | X |  |  |  |  |  |  |  |  |
| especificar um diagrama com os componentes internos do *middleware* |  | X | X |  |  |  |  |  |  |  |
| desenvolver o *middleware* proposto |  |  |  | X | X | X |  |  |  |  |
| desenvolver uma aplicação Web para dispositivos móveis e dois serviços para testar o *middleware* |  |  |  |  |  |  | X | X |  |  |
| testar a aplicação e avaliar os resultados |  |  |  |  |  |  |  |  | X |  |
| escrever os resultados do trabalho |  | X | X |  |  |  |  | X | X | X |

Fonte: elaborado pelo autor.

# REVISÃO BIBLIOGRÁFICA

Esta seção descreve brevemente os assuntos que fundamentarão o estudo a ser realizado para construir o *middleware* que possa resolver o problema exposto anteriormente: apresentar um *middleware* que seja escalável e resiliente para comunicação cliente/servidor entre aplicações Web para dispositivos móveis e serviços em um ambiente de computação distribuída.

Um sistema distribuído é definido por “[...] um conjunto de computadores independentes que se apresenta a seus usuários como um sistema único e coerente” (TANENBAUM; STEEN, 2008, p. 1). Os autores também esclarecem sobre a estrutura e funcionamento do modelo de comunicação por mensagens.

O protocolo de comunicação Web que permite uma comunicação constante e bidirecional é definida pela WebSocket API, conforme é descrito por Lombardi (2015). Este protocolo será utilizado para comunicação dos dispositivos móveis e o servidor.

Aplicações móveis são aplicações escritas para dispositivos móveis e têm se tornado a principal ferramenta de comunicação das pessoas com a Internet. Desenvolver aplicações Web para dispositivos móveis pode ser uma forma mais viável para atender diferentes dispositivos sem a necessidade de reescrever muitas partes de código, conforme a explicação dada por Oehlman e Blanc (2012).

Referências

FERNANDES, Pedro Manuel da Conceição. **Desenvolvimento de uma Framework Real Time Web para HTML5**. 2013. TCC (Graduação) - Curso de Licenciatura em Sistemas e Tecnologia da Informação, Universidade Atlântica, Barbarena, 2013.

LOMBARDI, Andrew. **WebSocket:** LIGHTWEIGHT CLIENT SERVER COMMUNICATIONS. 1. ed. Sebastopol, CA: O’Reilly Media, Inc., 2015.

MICROSOFT AZURE. **O que é middleware?** Disponível em: https://azure.microsoft.com/pt-br/overview/what-is-middleware. Acesso em: 20 de novembro de 2021.

OEHLMAN, Damon; BLANC, Sébastien. **Aplicativos Web Pro Android:** Desenvolvimento Pro Android Usando HTML5, CSS3 & JavaScript. 1. ed. Rio de Janeiro: Ciência Moderna, 2012.

SILVA, Jorge Pereira da. **EcoCIT:** uma plataforma escalável para desenvolvimento de aplicações de IoT. 2017. 138f. Dissertação (Mestrado em Sistemas e Computação) - Centro de Ciências Exatas e da Terra, Universidade Federal do Rio Grande do Norte, Natal, 2017.

SOUZA, Samuel Pereira de. **ANÁLISE DE DESEMPENHO DE APIS DE *SOCKETS***. 2016. TTC (Graduação) - Curso de Graduação em Redes de Computadores, Universidade Federal do Ceará, Quixadá, 2016.

TANENBAUM, Andrew S.; STEEN, Maarten Van. **SISTEMAS DISTRIBUIDOS:** princípios e paradigmas. 2. ed. São Saulo: Pearson, 2008.

FORMULÁRIO DE avaliação BCC – PROFESSOR TCC I

Avaliador(a): Dalton Solano dos Reis

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ASPECTOS AVALIADOS | | atende | atende parcialmente | não atende |
| ASPECTOS TÉCNICOS | 1. INTRODUÇÃO   O tema de pesquisa está devidamente contextualizado/delimitado? | X |  |  |
| O problema está claramente formulado? | X |  |  |
| 1. OBJETIVOS   O objetivo principal está claramente definido e é passível de ser alcançado? | X |  |  |
| Os objetivos específicos são coerentes com o objetivo principal? | X |  |  |
| 1. JUSTIFICATIVA   São apresentados argumentos científicos, técnicos ou metodológicos que justificam a proposta? | X |  |  |
| São apresentadas as contribuições teóricas, práticas ou sociais que justificam a proposta? | X |  |  |
| 1. METODOLOGIA   Foram relacionadas todas as etapas necessárias para o desenvolvimento do TCC? | X |  |  |
| Os métodos, recursos e o cronograma estão devidamente apresentados? | X |  |  |
| 1. REVISÃO BIBLIOGRÁFICA (atenção para a diferença de conteúdo entre projeto e pré-projeto)   Os assuntos apresentados são suficientes e têm relação com o tema do TCC? |  | X |  |
| ASPECTOS METODOLÓGICOS | 1. LINGUAGEM USADA (redação)   O texto completo é coerente e redigido corretamente em língua portuguesa, usando linguagem formal/científica? | X |  |  |
| A exposição do assunto é ordenada (as ideias estão bem encadeadas e a linguagem utilizada é clara)? | X |  |  |
| 1. ORGANIZAÇÃO E APRESENTAÇÃO GRÁFICA DO TEXTO   A organização e apresentação dos capítulos, seções, subseções e parágrafos estão de acordo com o modelo estabelecido? | X |  |  |
| 1. ILUSTRAÇÕES (figuras, quadros, tabelas)   As ilustrações são legíveis e obedecem às normas da ABNT? |  | X |  |
| 1. REFERÊNCIAS E CITAÇÕES   As referências obedecem às normas da ABNT? | X |  |  |
| As citações obedecem às normas da ABNT? | X |  |  |
| Todos os documentos citados foram referenciados e vice-versa, isto é, as citações e referências são consistentes? | X |  |  |

|  |  |  |
| --- | --- | --- |
| O projeto de TCC será reprovado se:   * qualquer um dos itens tiver resposta NÃO ATENDE; * pelo menos **4 (quatro)** itens dos **ASPECTOS TÉCNICOS** tiverem resposta ATENDE PARCIALMENTE; ou * pelo menos **4 (quatro)** itens dos **ASPECTOS METODOLÓGICOS** tiverem resposta ATENDE PARCIALMENTE. | | |
| **PARECER**: | ( X ) APROVADO | ( ) REPROVADO |