# Plano de Trabalho de Conclusão de Curso
# Análise e caracterização de arquiteturas de microserviços empregados a jogos MMORPG voltada a otimização do uso de recursos de gerenciamento de mundos virtuais

Marlon Henry Schweigert – marlon.henry@magrathealabs.com
Charles Christian Miers – charles.miers@udesc.br *(orientador)*

Turma 2018/1 – Joinville/SC

1 de Fevereiro de 2018

**Resumo**

A crescente onda de jogos massivos tem requirindo de novas arquiteturas a fim de suprir a demanda de usuários com menor custo de recursos computacionais. Projetar essas arquiteturas, do ponto de vista da rede, é algo extremamente importante e impactante para o sucesso desses jogos. O objetivo deste trabalho é propor uma análise voltada a otimização do recurso computacional consumido pelas arquiteturas descritas na literatura. Esse objetivo será atingido após efetuar uma análise das arquiteturas descritas na literatura, utilizando simulações sobre uma nuvem computacional, identificando gargalos das arquiteturas descritas e provendo soluções viáveis a esses problemas. Os resultados obtidos não são importantes somente a jogos massivos, mas a qualquer serviço que demande de escalabilidade, flexibilidade, gerenciamento e desempenho.

**Palavras-chave:** *arquitetura de microserviços, desenvolvimento de jogos, rede de jogos, jogos massivos, otimização de recursos, nuvens computacionais*

## 1 Introdução e Justificativa

Os avanços tecnológicos de sistemas distribuídos estão permitindo que as pessoas utilizem serviços com volumes massivos de dados para aplicações sensíveis a latência. Essa situação é propícia a área de jogos massivos e tem atraído pesquisadores para essa área de estudo [Kim, Kim e Park 2008]. O principal objetivo das pesquisas é reduzir a carga e o impacto a latência para o usuário final nesses serviços [Huang, Ye e Cheng 2004] [Saldana et al. 2012] [Barri, Gine e Roig 2011]. Reduzir a carga e impacto da latência em serviços para jogos massivos resultam em uma melhor experiência de jogabilidade aos usuários finais[Huang, Ye e Cheng 2004].

Jogos Massively Multiplayer Online Role-Playing Game (MMORPG) são utilizados como negócio viável e lucrativo, a qual a experiência de uso do usuário final é um fator crítico para o sucesso. O mercado de jogos massivos multijogadores de interpretação (*MMORPG*) vem crescendo desde 2012 [Bilton 2011], sendo no ano de 2016 um dos mais lucrativos[Statista 2016]. A sua projeção para

2018 é que sejam arrecadados mais de 30 bilhões de dólares americanos com esta categoria de jogos [Statista 2017], um aumento de 20% a mais sobre o ano de 2016.

MMORPG são jogos de interpretação de papéis massivos. A principal característica desse estilo de jogo é a comunicação e representação virtual de um mundo fantasia na qual cada jogador pode interagir com objetos virtuais compartilhados ou tomar ações sobre outros de jogadores em tempo real, tendo como principais objetivos a resolução de problemas conforme a sua regra de *design*, o desenvolvimento do personagem e a interação entre os jogadores[Hanna 2015].

Um jogo MMORPG é arquitetado em 2 partes [Kim, Kim e Park 2008]:

- **Serviço**: É o macroserviço que implementa as regras de negócio e requisitos do jogo. O serviço disponibiliza uma interface com ações possíveis ao cliente sobre algum protocolo de rede.
- **Cliente**: Cliente é a aplicação que realizará requisições com a interface do macroserviço, exibindo o estado de jogo de forma imersiva.

A maioria dos jogos MMORPG disponíveis no mercado estão implementados sobre uma arquitetura que executa sobre diversos servidores, nos quais o desempenho destes servidores influencia tanto na experiência de jogabilidade do usuário final, quanto no custo de manutenção destes serviços. Modelar um sistema de alto desempenho torna-se um trabalho essencial para a satisfação do usuário final[Huang, Ye e Cheng 2004]. As ocorrências geradas por um sistema de baixo desempenho podem acarretar em frustração do usuário com o serviço e/ou aumento dos gastos com recurso computacional para manter o serviço.

## 1.1 Ocorrências em serviços massivos

Uma métrica popular para mensurar o desempenho de um serviço MMORPG é o número de conexões[Huang, Ye e Cheng 2004]. Em geral, caso o serviço ultrapasse o limite para o qual este foi projetado, diversas falhas de conexão, problemas de lentidão ou dessincronização com o cliente podem ocorrer. As ocorrências mais comuns são[Huang, Ye e Cheng 2004]:

- **Longo tempo de resposta aos clientes**: implica em uma qualidade insatisfatória de jogabilidade ao usuário ou até mesmo impossibilitando o uso do serviço.
- **Dessincronização com os clientes**: realiza reversão na aplicação. Reversão é definida pela situação onde uma requisição é solicitada ao servidor, um pré-processamento aparente é executado e essa requisição é negada, sendo necessário desfazer o pré-processamento aparente realizado ao cliente.
- **Problemas internos ao serviço**: pode estar relacionada a diversos outros erros internos de implementação ou capacidade de recurso computacional (*e.g.,* sobrecarga no banco de dados, gerenciamento lento do espaço ou inconsistências dentro do jogo perante a regra de negócios).
- **Falha de conexão entre o cliente e os microserviços**: causa a negação de serviço ao usuário final.

Existem algumas causas comuns para essas as ocorrências descritas[Huang, Ye e Cheng 2004], na qual são apresentadas a seguir:

- **Baixo poder computacional do servidor**: poder computacional baixo para a qualidade de experiência de jogabilidade do usuário final desejada.
- **Complexidade de algoritmos**: o serviço usa algoritmos de alta complexidade ou regras de negócios que demandam por um algoritmo complexo.
- **Limitado pela própria arquitetura**: está limitado diretamente pelo número de conexões, não suportando a carga recebida.

A área de desenvolvimento web compartilha várias ocorrências comuns geradas por sobrecarga do serviço[Khazaei et al. 2016]. Em desenvolvimento web é comum utilizar a abordagem de microserviços para resolver o problema de sobrecarga, modularizando seu funcionamento em pedaços menores. Da mesma forma, faz sentido modularizar um serviço MMORPG em microserviços para suportar cargas maiores e diminuir o custo de manutenção [Villamizar et al. 2016].

## 1.2 Arquiteturas de microserviços

Entende-se por microserviço as aplicações que executam operações menores de um macroserviço, da melhor forma possível[Willson 2017]. Microserviços devem funcionar separadamente e de forma autônoma. Seu funcionamento deve ser desenhado para permitir alinhamentos de alta coesão e baixo acoplamento entre os demais microserviços existentes em um macroserviço[Acevedo, Jorge e Patiño 2017].

Essas arquiteturas iniciam uma nova linha de desenvolvimento de aplicações preparadas para executar sobre nuvens computacionais, promovendo maior flexibilidade, escalabilidade, gerenciamento e desempenho, sendo a principal escolha de arquitetura de grandes empresas como Amazon, Netflix e LinkedIn[Khazaei et al. 2016][Villamizar et al. 2016].

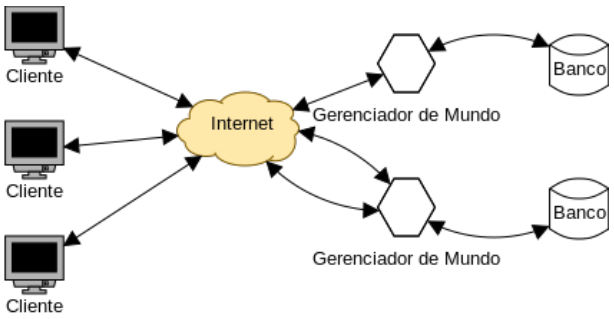
Um microserviço é definido pelas seguintes características[Acevedo, Jorge e Patiño 2017]:

- Deve posibilitar a implementação como uma peça individual do macroserviço.
- Deve funcionar individualmente.
- Deve estar disponível na rede.
- Cada serviço deve ter uma interface. Essa interface deve ser o suficiente para utilizar o microserviço.
- O serviço pode ser utilizado por qualquer linguagem de programação e/ou plataforma.
- O serviço deve executar com as dependências mínimas.
- Ao agregar vários microserviços, o macroserviço resultante poderá prover funcionalidades complexas.

Alguns exemplos de arquitetura de microserviços para jogos MMORPG são as arquiteturas apresentadas por Rudy (Figura 1), Salz (Figura 2) e a arquitetura escrita por Knowles (Figura 3).

A arquitetura Rudy (Figura 1) é formada por um sistema cliente-servidor monolítico, na qual cada microserviço individual gerencia um mundo mútuo dos demais gerenciadores de mundo [Ruddy 2011]. Essa arquitetura dificulta a escalabilidade, modificações e manutenção[Acevedo, Jorge e Patiño 2017], além de segregar a comunidade de jogadores em servidores menores[Ruddy 2011]. Inicialmente essa arquitetura foi pensada para ser um sistema Cliente-Servidor monolítico. A arquitetura Rudy é uma arquitetura de microserviços adaptada de um serviço cliente-servidor[Ruddy 2011]. O jogo Tibia, operante sobre essa arquitetura, possui 68 mundos oficiais (Sendo 2 servidores de teste) [Ruddy 2011], com capacidade para 2.000 jogadores em cada servidor.

**Figura 1:** Arquitetura Ruy, utilizada no jogo Tibia [Ruddy 2011].



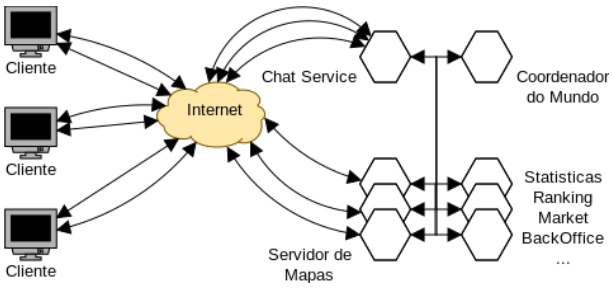

Adaptado de: [Ruddy 2011]

A arquitetura Salz (Figura 2) é formada por diversos microserviços [Salz 2016]. O principal objetivo dessa arquitetura é modularizar o serviço visando melhorar a escalabilidade. A arquitetura é planejada para funcionar conforme a seguinte especificação[Salz 2016]:

- O mundo é distribuído sobre os vários servidores de mapas. Cada microserviço gerencia uma região do mundo, denominado *chunk*.
- Jogadores mudam a conexão com os microserviços quando estão posicionados na borda de um *chunk*.
- A autorização de acesso aos microserviços é obtido pelo banco de dados.
- O coordenador do mundo é responsável por tudo que seja de escopo global (*e. g.,* Grupos, chat global, guildas, *etc.*).

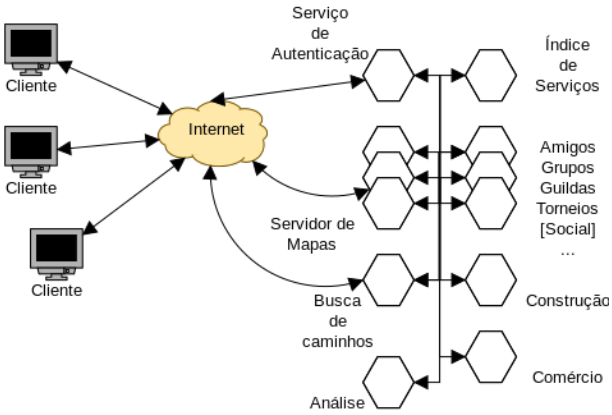
A arquitetura Knowles (Figura 3) é distribuida em diversos microserviços assim como a arquitetura Salz (Figura 2). A grande diferença em comparação a arquitetura Salz (Figura 2) é a maior decomposição da arquitetura para outros microserviços e a conexão direta entre esses microserviços e o cliente. O principal objetivo dessa arquitetura é facilitar a manutenção e desempenho de reinicialização do macroserviço[Willson 2017]. Guild Wars 2 é um jogo que executa sobre a arquitetura Knowles. Ele é popularmente conhecido por ter seus serviços sempre ativos, visto que a arquitetura

**Figura 2:** Arquitetura Salz, utilizada no jogo Albion [Salz 2016].




Adaptado de: [Salz 2016]

**Figura 3:** Arquitetura Knowles, utilizada no jogo Guild Wars 2 [Willson 2017].




Adaptado de: [Willson 2017]

possibilita desativar pequenos pedaços do serviço para manutenções básicas e a sua reinicialização é rápida para manutenções críticas.

Conforme as arquiteturas de microserviços dos jogos MMORPG aumentam para atender novas demandas do mercado, o seu custo computacional e complexidade de nós na nuvem computacional aumentam. Por esse motivo a análise e otimização dessas arquiteturas é importante para impactar em um melhor desempenho tanto ao usuário final, quanto a otimização de lucros das empresas que disponibilizam esses serviços.

## 1.3 Justificativa

A proposta de otimização das análises realizadas sobre as arquiteturas de microserviços para jogos massivos focada ao gerenciamento de mundos virtuais, proposta pela literatura, traz impacto direto ao usuário final e a empresa que mantem esse sistema[Huang, Ye e Cheng 2004].

A proposta contida no atual trabalho é otimizar tais arquiteturas visando melhorar algumas características específicas:

- Recursos utilizados
- Escalabilidade
- Disponibilidade

- Flexibilidade

## 2 Objetivos

Análisar e caracterizar arquiteturas de microserviços empregados a jogos MMORPG voltada a otimização do uso de recursos (*e.g., banda, memória e processamento* [Huang, Ye e Cheng 2004]) de gerenciamento de mundos virtuais. Propor soluções tangíveis (*e.g.,* protocolos, compressão, paralelismo, *etc.* [Huang, Ye e Cheng 2004]) para otimizar os recursos utilizados pelas arquiteturas analisadas.

Os objetivos específicos são:

- Identificar e caracterizar arquiteturas empregadas na categoria de jogos do presente trabalho.
- Identificar e caracterizar os protocolos utilizados nessas arquiteturas.
- Identificar e caracterizar os microserviços dessas arquiteturas.
- Identificar e analisar ferramentas de análise de recursos para definir métricas as arquiteturas identificadas e caracterizadas.
- Especificar requisitos específicos para a arquitetura estudada.
- Aplicar as arquiteturas descritas na literatura em uma nuvem de computadores *OpenStack*.
- Analisar o comportamento das arquiteturas aplicadas, levantando questões de desempenho e recursos utilizados.
- Propor alternativas de otimização para os problemas encontrados nas devidas arquiteturas.

## 3 Metodologia

Para que seja possível atingir os objetivos, serão utilizados dois métodos: pesquisa referenciada, desenvolvida durante o Trabalho de Conclusão de Curso I, e pesquisa aplicada, desenvolvida durante o Trabalho de Conclusão de Curso II.

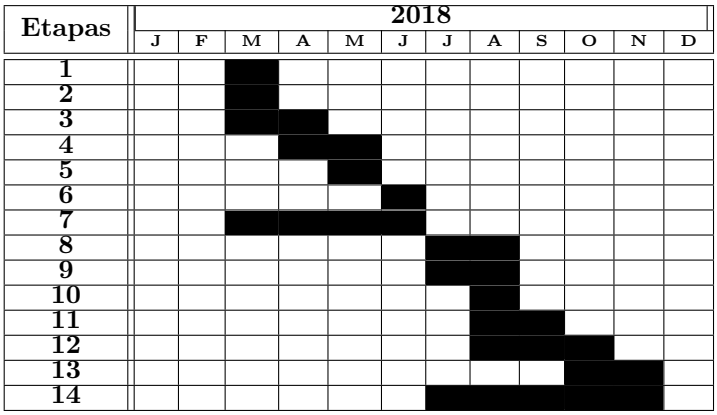
Na pequisa referenciada serão levantadas arquiteturas da literatura, buscando as mais adequadas ao escopo deste trabalho. Será dividido em três situações: levantamento de arquiteturas de microserviços descritas na literatura e caracterização dessas arquiteturas. Por fim, o levantamento e caracterização de possíveis simulações para efetuar testes durante a pesquisa aplicada.

Na pesquisa aplicada, o resultado a ser obtido é a análise das arquiteturas de microserviços caracterizadas, visando uma profunda análise sobre os recursos computacionais consumidos e identificação de seus gargalos. Divide-se em três situações: aplicação das arquiteturas descritas e selecionadas, realização dos testes utilizando a simulação descrita na pesquisa referenciada e análise dos dados coletados.

Para que os resultados sejam alcançados, são definidas as seguintes etapas:

1. **Levantamento e fichamento das referências:** Pesquisa de fontes para embasamento teórico do trabalho, com base nos objetivos específicos;
2. **Consolidação das referências:** Compreensão e seleção de artefatos literários que permitam atingir o objetivo do Trabalho de Conclusão de Curso I;
3. **Identificação e caracterização de arquiteturas descritas na literatura:** Enumeração e caracterização das arquiteturas de microserviços descritas na literatura, bem como os seus objetivos;
4. **Especificação das arquiteturas selecionadas:** Especificar o funcionamento das arquiteturas selecionadas.
5. **Identificação e caracterização de simulações aplicáveis ao teste:** Eleger e caracterizar a simulação a ser aplicada nos testes;
6. **Especificação da simulação elegida:** Especificar o funcionamento específico da simulação elegida para testes;
7. **Escrita Trabalho de Conclusão de Curso I**;
8. **Desenvolvimento da simulação:** Desenvolvimento da simulação para interagir com as arquiteturas de microserviços;
9. **Desenvolvimento da arquitetura:** Desenvolvimento da arquitetura para executar os testes;
10. **Aplicação das arquiteturas selecionadas na pesquisa referênciada:** Aplicação das arquiteturas descritas sobre uma nuvem computacional;
11. **Realização dos testes utilizando a simulação descrita na pesquisa referênciada:** Execução de testes da arquitetura desenvolvida sobre a nuvem computacional;
12. **Análise das arquiteturas testadas:** Analisar as métricas obtidas dos testes e descrever resultados, identificando possíveis gargalos nas arquiteturas;
13. **Otimização para melhorar as métricas obtidas:** Analisar pontos de gargalo nos microserviços analisados e propor soluções viáveis para aumentar o desempenho desses sistemas.
14. **Escrita Trabalho de Conclusão de Curso II**;

# 4 Cronograma proposto




# 5 Linha de Pesquisa

Este trabalho será desenvolvido junto ao Grupo de Redes e Aplicações Distribuídas (GRADIS) e ao Laboratório de Processamento Paralelo e Distribuído (LabP2D). Esta pesquisa abrange as áreas de Redes de Computadores, Sistemas Distribuídos, Segurança em Redes de Computadores, Processamento Paralelo e Engenharia de Software.

# 6 Forma de Acompanhamento/Orientação

O acompanhamento será realizado principalmente através de reuniões semanais ou quinzenais, com duração máxima de 1 (uma) hora. Eventualmente as reuniões poderam ser trocadas por vídeo-conferência, troca de menssagens de correio eletrônico ou telefone. O controle das tarefas a fazer serão feitas baseadas em uma ata gerada a cada reunião. Metas semanais ou quinzenais serão atribuídas para melhor acompanhamento.

# Referências

[Acevedo, Jorge e Patiño 2017]ACEVEDO, C. A. J.; JORGE, J. P. G. y; PATIñO, I. R. Methodology to transform a monolithic software into a microservice architecture. In: *2017 6th International Conference on Software Process Improvement (CIMPS)*. [S.l.: s.n.], 2017. p. 1–6.

[Barri, Gine e Roig 2011]BARRI, I.; GINE, F.; ROIG, C. A hybrid p2p system to support mmorpg playability. In: *2011 IEEE International Conference on High Performance Computing and Communications*. [S.l.: s.n.], 2011. p. 569–574.

[Bilton 2011]BILTON, N. *Search Bits SEARCH Video Game Industry Continues Major Growth, Gartner Says*. 2011. Acessado em: 19/01/2018. Disponível em: <https://bits.blogs.nytimes.com/2011/07/05/video-game-industry-continues-major-growth-gartner-says/>.

[Hanna 2015]HANNA, P. *Video Game Technologies*. 2015. Acessado em: 19/01/2018. Disponível em: <https://www.di.ubi.pt/ agomes/tjv/teoricas/01-genres.pdf>.

[Huang, Ye e Cheng 2004]HUANG, G.; YE, M.; CHENG, L. Modeling system performance in mmorpg. In: *IEEE Global Telecommunications Conference Workshops, 2004. GlobeCom Workshops 2004.* [S.l.: s.n.], 2004. p. 512–518.

[Khazaei et al. 2016]KHAZAEI, H. et al. Efficiency analysis of provisioning microservices. In: *2016 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*. [S.l.: s.n.], 2016. p. 261–268.

[Kim, Kim e Park 2008]KIM, J. Y.; KIM, J. R.; PARK, C. J. Methodology for verifying the load limit point and bottle-neck of a game server using the large scale virtual clients. In: *2008 10th International Conference on Advanced Communication Technology*. [S.l.: s.n.], 2008. v. 1, p. 382–386. ISSN 1738-9445.

[Ruddy 2011]RUDDY, M. *Inside Tibia, The Technical Infrastructure of an MMORPG*. 2011. Disponível em: <http://twvideo01.ubm-us.net/o1/vault/gdceurope2011/slides/Matthias_Rudy_ProgrammingTrack_InsideTibiaArchitecture.pdf>.

[Saldana et al. 2012]SALDANA, J. et al. Traffic optimization for tcp-based massive multiplayer online games. In: *2012 International Symposium on Performance Evaluation of Computer Telecommunication Systems (SPECTS)*. [S.l.: s.n.], 2012. p. 1–8.

[Salz 2016]SALZ, D. *Albion Online - A Cross-Platform MMO (Unite Europe 2016, Amsterdam)*. 2016. Disponível em: <https://www.slideshare.net/davidsalz54/albion-online-a-crossplatform-mmo-unite-europe-2016-amsterdam>.

[Statista 2016]STATISTA. *Statistics and Facts on MMO/MMORPG gaming*. 2016. Acessado em: 19/01/2018. Disponível em: <https://www.statista.com/topics/2290/mmo-gaming/>.

[Statista 2017]STATISTA. *Games market revenue worldwide in 2015, 2016 and 2018, by segment and screen (in billion U.S. dollars)*. 2017. Acessado em: 19/01/2018. Disponível em: <https://www.statista.com/statistics/278181/video-games-revenue-worldwide-from-2012-to-2015-by-source/>.

[Villamizar et al. 2016]VILLAMIZAR, M. et al. Infrastructure cost comparison of running web applications in the cloud using aws lambda and monolithic and microservice architectures. In: *2016 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*. [S.l.: s.n.], 2016. p. 179–182.

[Willson 2017]WILLSON, S. C. *Guild Wars Microservices and 24/7 Uptime*. 2017. Disponível em: <http://twvideo01.ubm-us.net/o1/vault/gdc2017/Presentations/Clarke-Willson_Guild Wars 2 microservices.pdf>.

***Charles Christian Miers***

***Marlon Henry Schweigert***

***Rafael Rodrigues Obelheiro***
*(Coordenador do GRADIS)*