UNIVERSIDADE FEDERAL DO MARANHÃO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

THIAGO DA SILVA NASCIMENTO

AVALIAÇÃO DE DESEMPENHO DE RENDERIZAÇÃO DE PÁGINAS WEB: UM ESTUDO DE CASO COM TECNOLOGIA JAVASCRIPT

SÃO LUÍS-MA
2018

Thiago da Silva Nascimento

AVALIAÇÃO DE DESEMPENHO DE RENDERIZAÇÃO DE PÁGINAS WEB: UM ESTUDO DE CASO COM TECNOLOGIA JAVASCRIPT

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Mário Antonio Meireles Teixeira

São Luís-MA
2018

Ficha gerada por meio do SIGAA/Biblioteca com dados fornecidos pelo(a) autor(a).
Núcleo Integrado de Bibliotecas/UFMA

Nascimento, Thiago da Silva.
Avaliação de desempenho de renderização de páginas web: um estudo de caso com tecnologia javascript / Thiago da Silva Nascimento. – 2018.
50 f.

Orientador(a): Mário Antonio Meireles Teixeira.
Monografia (Graduação) – Curso de Ciência da Computação, Universidade Federal do Maranhão, São Luís, 2018.

1. Avaliação de Desempenho. 2. Lado Cliente. 3. Lado Servidor. 4. Renderização de páginas Web. 5. Tecnologia Javascript. I. Teixeira, Mário Antonio Meireles. II. Título.

Thiago da Silva Nascimento

AVALIAÇÃO DE DESEMPENHO DE RENDERIZAÇÃO DE PÁGINAS WEB: UM ESTUDO DE CASO COM TECNOLOGIA JAVASCRIPT

Monografia apresentada como requisito parcial para obtenção do grau de Bacharel em Curso de Ciência da Computação da Universidade Federal do Maranhão - UFMA, Centro de Ciências Exatas e Tecnologia.

Data de Aprovação: 12/07/2018

Banca Examinadora

Prof. Dr. Mário Antonio Meireles Teixeira
Universidade Federal de Maranhão
Orientador

Profª. Msc. Alana Oliveira Meireles Teixeira
Universidade Federal de Maranhão

Prof. Dr. Geraldo Braz Júnior
Universidade Federal de Maranhão

A Larissa, a meus pais Francisco e Neide e a minha descendência que há de vir.

AGRADECIMENTOS

Primeiramente à Deus, criador, autor da minha fé. Agradeço pela força, sabedoria e fôlego de vida. Nunca me desamparou e nunca me desamparará mesmo eu sendo um fraco pecador.

Agradeço a Universidade Federal do Maranhão pela oportunidade de concluir minha formação no curso de Ciência da Computação.

Agradeço a minha amada Larissa. Muito obrigado pelo apoio, compreensão e por nunca deixar eu desistir. Eu te amo.

Agradeço a minha família, dona Neide e seu Francisco. Agradeço por sonharem este sonho junto comigo.

Agradeço ao meu orientador Mário Meireles, pela grande ajuda, pela paciência e pela experiência compartilhada na construção deste trabalho.

A todos os companheiros do curso, em especial Diego Fontenelle, Júlio Filho e Felipe Aragão, por todo conhecimento que podemos compartilhar ao longo da vida acadêmica e ambiente profissional. Também agradeço pelo apoio e motivação.

Ao departamento de informática que continuamente visou oferecer um ensino de qualidade a todos os alunos. E a todos que direta ou indiretamente contribuíram para a realização desse trabalho. Sou muito agradecido.

"Aprender uma lição sem dor, não tem significado, isso porque as pessoas não conseguem obter nada sem sacrificar alguma coisa mas quando elas superaram as dificuldades e conseguem o que querem, as pessoas conquistam um coração forte que não perde pra nada, é um coração forte como aço."

Hiromu Arakawa

RESUMO

Em um momento que compreende uma constante evolução de tecnologias e uma demanda por aplicações com tempo de resposta otimizado, surge a necessidade de selecionar um modelo para desenvolvimento que atenda as necessidades de desempenho para apresentação de informação em páginas para Web. Existem duas abordagens para apresentação destas informações: A renderização no lado Servidor e a renderização no lado Cliente. Este trabalho realiza um estudo que envolve uma análise no desempenho destas abordagens através de implementações utilizando tecnologia Javascript. Para essa análise é utilizado o Google Developer Tools - Performance para mensurar o tempo de renderização em cenários variados. Os experimentos realizados demonstram que a abordagem de renderização de páginas do lado cliente foi superior em todos os cenários realizados, onde foram consideradas variações na velocidade da rede de acesso, da CPU e na quantidade de objetos recuperados.

Palavras-chave: Avaliação de Desempenho, Tecnologia Javascript, Renderização de páginas Web, Lado Cliente, Lado Servidor.

ABSTRACT

At a time that includes a constant evolution of technologies and a demand for applications with optimized response time, there is a need to select a development model that meets the performance requirements for presenting information in Web pages. There are two approaches to present of this information: Server-side Rendering and Client-side Rendering. This work carries out a study that involves a performance analysis of these approaches through implementations using Javascript technology. For this analysis, Google Developer Tools - Performance is used to measure the rendering time in a variety of scenarios. The experiments carried out demonstrate that client-side page rendering was superior in all studied scenarios, where variations were introduced in access network bandwidth, CPU speed, and number of objects retrieved.

Keywords: Performance Analysis, Javascript Technology, Web page rendering, Client Side, Server Side.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
ARPAnet	Advanced Research Projects Agency Network
BaaS	Backend-as-a-Service
CERN	Conseil Européen pour la Recherche Nucléair
CSR	Client-Side Rendering
C/S	Cliente/Servidor
EJS	Embedded JavaScript templates
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
MPA	Multiple-Page Application
SPA	Single-Page Application
SSR	Server-Side Rendering
TI	Tecnologia da Informação
URI	Uniform Resource Identifier
WWW	Word Wide Web
W3C	World Wide Web Consortium
XML	eXtensible Markup Language

SUMÁRIO

1 INTRODUÇÃO

O conceito primário da Internet, a ARPAnet (Advanced Research Projects Agency Network), surge em 1969 como parte do projeto do Governo Norte Americano, com propósito militar (ALMEIDA, 2005). Entretanto, somente em 1989 o cientista da CERN Tim Berners-Lee, propos o projeto *WWW* (Word Wide Web) (BERNERS-LEE, 2006) como um canal colaborativo para troca de documentos científicos e direcionado a uma pequena rede de cientistas.

A Web é um sistema em expansão, que incorpora novos componentes e serviços em um ritmo acelerado. Aplicações como comércio eletrônico, bibliotecas digitais, vídeo por demanda e aprendizado à distância aumentam o tráfego da Web em ritmos ainda maiores. As lojas virtuais na Web permitem a compra de carros, livros, computadores e muito mais produtos e serviços. Muitas agências do governo estão usando websites para disseminar documentos e formulários para indivíduos, grupos de interesse público, empresas privadas e outras agências do governo. As interações entre agências do governo e cidadãos estão sendo facilitadas por meio de serviços baseados na Web. Audiências virtuais sobre políticas públicas podem ser realizadas através da Web (MENASCÉ; ALMEIDA, 2003).

Para Emery (2016), a ascenção da Internet está diretamente relacionada a utilização de meios gráficos, pois eles facilitam o uso das funcionalidades dos computadores. De fato, para que a maioria dos usuários pudessem utilizar os recursos da Web era necessário que essa troca de informação na internet evoluísse, de modo que usuários deixassem de ser apenas consumidores de informação, mas se tornassem provedores também.

Alguns websites populares recebem milhões de solicitações por dia. Não é raro que esses sites apresentem tempos de resposta por vezes altos. Isso tem sido uma fonte de frustação para muitos usuários Web e um problema para o gerenciamento de muitos websites.

Um dos problemas mais difíceis para os administradores de websites é o dimensionamento adequado de sua infra-estrutura de Tecnologia de Informação (TI), de modo que possam oferecer a qualidade de serviço exigida por seus usuários. Esse desafio requer a possibilidade de monitorar o desempenho de websites e de seus serviços. É necessário monitorar a intensidade da carga de trabalho, detectar gargalos, prever limitações de capacidade futuras e determinar o modo mais econômico de se fazer o upgrade de sistemas para a Web. Isso ajuda a contornar problemas de desempenho e enfrentar as demandas de carga de trabalho cada vez maiores (MENASCÉ; ALMEIDA, 2003).

A necessidade de desenvolver aplicações para Web é resultado da expansão e populariza-

ção da Internet. O cenário atual é caracterizado pela diversidade de linguagens, abordagens e metodologias de desenvolvimento. Vega (2017) caracteriza os sites atuais como "falsos sites", pois grande parte possuem uma série de comportamentos que os tornam aplicações para Web. Os sites modernos permitem envio de mensagens, atualizações de informações pessoais, compras e negociações. Isto exige que desenvolvedores busquem soluções na criação de novos produtos para a Internet.

As atuais abordagens utilizadas para exibir o conteúdo de um site são a SSR (Server-Side Rendering) e a CSR (Client-Side Rendering), que possuem fluxo de trabalho diferente uma da outra. Com isso surge a necessidade de selecionar uma abordagem que apresente um melhor desempenho na construção de aplicações.

1.1 Objetivo

1.1.1 Objetivo Geral

Este trabalho tem como objetivo realizar uma avaliação de desempenho das abordagens SSR e CSR, através do estudo de duas tecnologias de desenvolvimento populares, Embedded JavaScript templates e a biblioteca de interfaces React.

1.1.2 Objetivo Específico

- Medir o tempo de renderização de páginas no lado cliente e no lado servidor;
- Identificar o fluxo de trabalho das abordagens CSR e SSR;
- Apresentar cenários variados de utilização das abordagens.

1.2 Organização do trabalho

O presente documento está organizado da seguinte forma: O Capítulo 2 apresenta os conceitos básicos de Sistemas Web, compreendendo uma descrição sobre arquitetura Cliente/Servidor, abordando também a arquitetura de aplicações web tradicionais e aplicações web modernas.

O Capítulo 3 apresenta fundamentação teórica sobre Avaliação de Desempenho, descrevendo as técnicas e as etapas necessárias para realizar uma avaliação.

O estudo de caso deste trabalho será apresentado no capítulo 4, em que foi realizado um experimento através da aferição de tempos médios para cada abordagem em diferentes cenários.

Finalmente, o Capítulo 5 apresenta e discute os resultados obtidos.

2 SISTEMAS WEB

Ao processo de criação de sistemas, aplicações e infraestrutura de TI, as organizações dedicam grande parte de seu tempo. Os sistemas críticos são criados e ajustados para executar os principais processos de negócios, com propósito de operarem com alta disponibilidade. Em muitos casos, os aplicativos de missão crítica são executados em sistemas legados e não há justificativa convincente para mover os aplicativos para servidores da Web (LINDGREN, 2001).

M'baya et al. (2017) afirmam que a modernização de aplicativos legados é um desafio significativo tanto em nível técnico quanto em nível de negócios. Sendo assim, os processos de negócio precisam ser adaptados de modo a atender evoluções. A modernização para aplicações web leva as organizações a lidar com as necessidades emergentes do mercado. Entretanto, as organizações que desejam aproveitar os recursos da Web devem encontrar maneiras de integrar o novo com o antigo (LINDGREN, 2001).

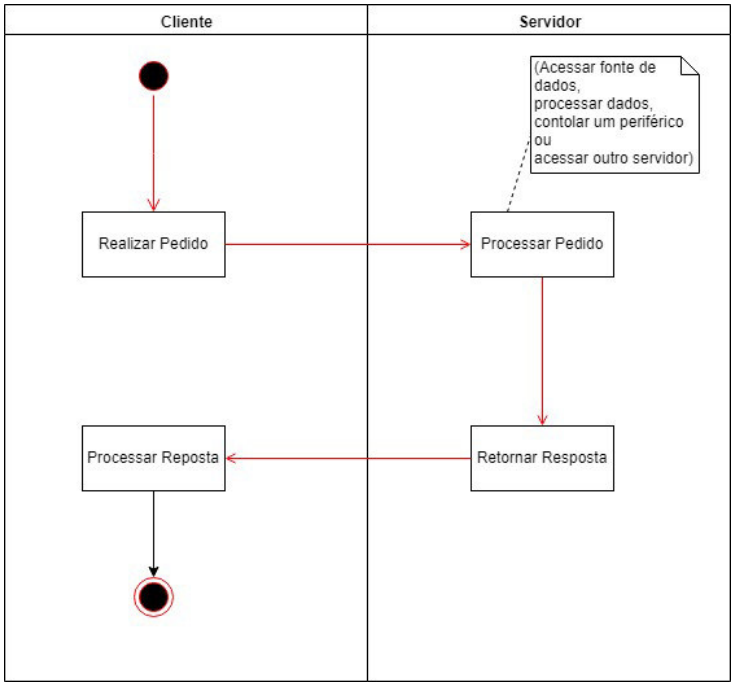
Neste capítulo são abordados os principais conceitos teóricos, relativos a Sistemas Web. Inicialmente, uma breve explicação sobre arquitetura Cliente/Servidor. Em seguida é abordada a aplicação do conceito de Cliente/Servidor na construção de sistemas, com uma subdivisão de Sistemas em Back-end e Front-end. Por fim, as duas últimas sessões falam, respectivamente, sobre aplicações de múltiplas páginas e aplicações de uma página.

2.1 Arquitetura Cliente/Servidor

A arquitetura Cliente/Servidor (C/S) consiste em um modelo de software, que segundo Hill et al. (2017) e Gallaugher e Ramanathan (1996) é um paradigma computacional distribuído, que em sua forma mais fundamental envolve uma entidade de software chamada de cliente fazendo uma solicitação específica, que é atendida por outra entidade de software chamada de servidor, a qual é provedora de serviços. Em outras palavras "Um servidor executa "Serviços"[...] e os clientes utilizam tais "Serviços"através da rede"(MIURA, 1994, página 127, tradução nossa).

Na arquitetura C/S, a comunicação é feita através de troca de mensagens (FILETO, 2016). A Figura 1 demonstra o procedimento de comunicação C/S. O processo cliente envia uma solicitação ao servidor, que por sua vez interpreta a mensagem e, em seguida, prossegue em atender a solicitação. Para atender à solicitação, o servidor pode ter que se acessar uma fonte de dados (banco de dados), processar dados (realizar cálculos), controlar um periférico ou fazer uma solicitação adicional a outro servidor (GALLAUGHER; RAMANATHAN, 1996).

Figura 1 – Troca de mensagens Cliente/Servidor



Sobre a comunicação entre cliente e servidor:

> É importante entender que o relacionamento entre cliente e servidor é um relacionamento de comando/controle. Em qualquer troca, o cliente inicia a solicitação e o servidor responde de acordo. Um servidor não pode iniciar o diálogo com os clientes... (GALLAUGHER; RAMANATHAN, 1996, Página 8, tradução nossa)

O entendimento deste paradigma é de fundamental importância, pois constitui a base da maioria da comunicação de aplicativos e protocolos. Através da arquitetura C/S, é possível entender o funcionamento da computação distribuída. (COMER, 2017)

2.2 Arquitetura Cliente/Servidor na construção de sistemas web

Gallaugher e Ramanathan (1996) ressaltam o papel do arquiteto de software na concepção de sistemas. O arquiteto é o profissional responsável por considerar os fatores necessários para a construção de um sistema. Alguns fatores se destacam como:

- A complexidade da aplicação;
- O nível de integração e de interface requisitado;
- A quantidade e localização dos usuários;
- O tipo de rede e

- As necessidades gerais de transação da aplicação.

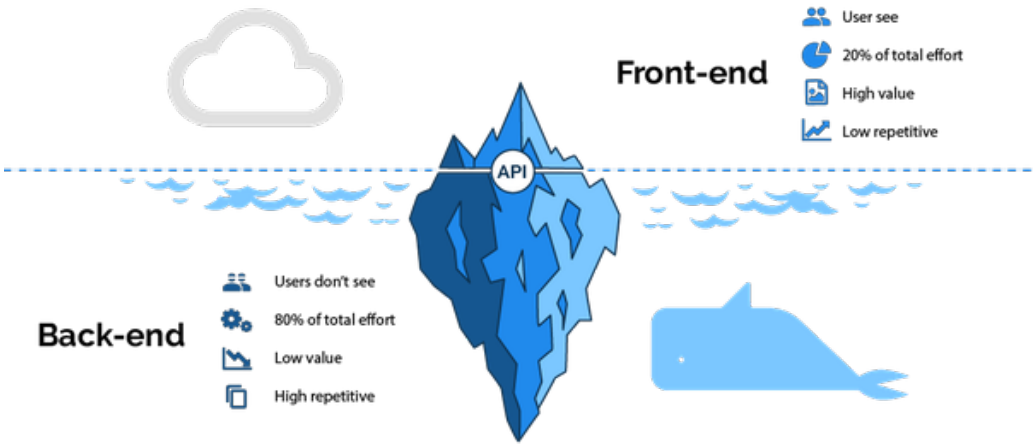
Sendo assim o arquiteto precisa organizar o projeto de modo a separar os pápeis de cada nível da aplicação. Isto está diretamente relacionado com a adoção do modelo C/S na construção de sistemas. Ao construir sistemas, uma parte da aplicação estará responsável por desempenhar o papel das funções de Servidor e outra parte estará responsável por desempenhar o papel das funções de Cliente.

A parte Cliente do sistema, sempre fornecerá a interface do sistema de usuários. É através do Cliente que os usuários poderão fazer solicitações de serviços (BRYAN, 1995). O lado Cliente de um sistema é fundamental por apresentar aos usuários uma interface amigável e de fácil navegabilidade.

A parte Servidor, fornece os serviços necessários lidando com funções centrais necessárias para todos os usuários do sistema. O servidor, que fornece esses serviços compartilhados, deve possuir um hardware maior, com mais poder de processamento, mais memória, maior capacidade de armazenamento (BRYAN, 1995).

Ao falar de Sistemas para Web, é prático dividi-los em duas seções: o *Front-end* e o *Back-end* (STRICKLAND, 2008). Fileto (2016), relaciona esta divisão de Sistemas Web com a arquitetura C/S. O Front-end é equivalente à parte Cliente e O Back-end é o equivalente ao Servidor. Os usuários não precisam conhecer a divisão arquitetural de sistemas para os utilizar. Também, esta divisão não deve impactar na experiência final do usuário ao utilizar sistemas neste modelo de arquitetura. A Figura 2 representa a divisão de sistemas web em front-end e back-end.

Figura 2 – Divisão de Sistemas em Back-end e Front-end



Fonte: Disponível em: <https://www.quora.com/What-is-backend-as-a-service-BaaS>. Acesso em: 09 de jun. de 2018.

Essa forma conveniente de divisão de Sistemas tornou-se mais comum nos últimos 10 a 15 anos com a evolução do JavaScript (WODEHOUSE, 2017a), que atualmente está presente não só nos navegadores mas também nos servidores.

Ambos Front-end e Back-end podem fazer parte da mesma aplicação ou ser aplicações desacopladas. Sobre isso, Hill et al. (2017) afirmam que clientes e servidores normalmente estarão em hardware separado, entretanto isso não impede que em algumas implementações de sistemas o Cliente e o Servidor possam residir no mesmo sistema físico.

2.3 Back-end

Em Academy (2018), o Back-end é toda a tecnologia necessária, tanto de linguagem de programação quanto arquitetural, para processamento de solicitações recebidas e envio de respostas ao cliente. O Back-end é composto usualmente por servidor, bancos de dados, Interfaces de Programação de Aplicações (APIs) e sistemas operacionais, entretanto, a estrutura do Back-end pode variar de aplicação para aplicação, seja através do uso de servidores em nuvem, contêineres de serviço, provedores de Backend-as-a-Service (BaaS) ou APIs para substituir processamento mais complexo (WODEHOUSE, 2017a).

Para simplificar o entendimento da estrutura de um Back-end, Academy (2018) divide o servidor em três componentes principais: o servidor, o banco de dados e a aplicação.

2.3.1 Servidor

Segundo Mitchell (2018), um servidor é um computador projetado para processar solicitações e fornecer dados para outro computador através de uma rede, seja ela local ou na rede mundial de computadores. É nele que sistemas são armazenados e executados. Embora qualquer computador executando um sistema possa funcionar como um servidor (ACADEMY, 2018), o uso mais típico da palavra faz referência a máquinas de alta potência e muito maiores (WODEHOUSE, 2017a).

2.3.2 Banco de Dados

Um banco de dados é uma coleção de dados relacionados, que são fatos conhecidos que podem ser traduzidos em registros e que têm seu significado implícito (ELMASRI, 2008).

No contexto de um sistema, o banco de dados é o conjunto de dados que tornam os sites dinâmicos. O dinamismo dos sitemas web é resultado da utilização dos bancos de dados.

Quando, por exemplo, um usuário efetua uma busca em um sistema, o sistema transforma aquela solicitação em uma consulta ao banco de dados. Outro exemplo é quando o usuário do sistema cria algum conteúdo novo em um sistema, estes dados são armazenados no banco de dados (WODEHOUSE, 2017a).

2.3.3 Aplicação

Uma aplicação, no contexto de back-end, é um software que funciona como a camada lógica necessária para atender várias solicitações através do método HTTP e da URI (Uniform Resource Identifier). Um método HTTP associado a uma URI é chamado de rota da aplicação (ACADEMY, 2018).

2.4 Front-end

O Front-end, também chamado de *client-side*, refere-se a tudo que acontece no navegador. É tudo o que os usuários conseguem ver e interagir diretamente (WODEHOUSE, 2017b). "O Front-End é a primeira camada com a qual nos deparamos ao acessarmos um site, uma intranet ou mesmo um sistema web. O front-end é onde encontramos a interface de navegação [...]"(CITRUS7, 2014).

O front-end também é fundamental por desempenhar as funções de validação da entrada de dados dos usuários, validação das respostas de solicitações feitas ao back-end e manter a consistência dos dados quando houver replicação local (dados armazedos no cache local) (FILETO, 2016).

2.5 Padrões de Design de Sistemas Web

Os sistemas web estão substituindo involuntariamente os sistemas para computador em ritmo acelerado (JAMES, 2017). Sobre isso, Neoteric (2016) afirma que sistemas para Web possuem algumas vantagens sobre sistemas para computador: são mais convenientes de usar, são fáceis de atualizar e não estão vinculados a um único dispositivo ou sistema operacional.

Atualmente, os sistemas web podem ser tão complexos quanto necessário (TARNOVSKIY, 2015). Com isso a demanda por aplicações complexas e mais sofisticadas tem crescido significativamente devido à migração de sistemas legados para sistemas web (NEOTERIC, 2016). Para Tarnovskiy (2015) praticamente qualquer tipo de sistema pode ser implementado como aplicativo da web. Mas a funcionalidade avançada requer arquitetura e design de aplicativos

avançados. Para isso, o arquiteto deve se perguntar que tipo de padrão de design de sistemas web deverá utilizar em seu projeto de software (JAMES, 2017).

Segundo Neoteric (2016) existem dois padrões principais de design para sistemas Web: aplicações de múltiplas páginas e aplicações de uma página. Ambos os modelos possuem vantagens e desvantagens descritas a seguir.

2.5.1 Aplicações de múltiplas páginas

Aplicações de múltiplas páginas (MPA) são o modo tradicional e mais comum de criação de sistemas web. Sobre MPAs Neoteric (2016) e Tarnovskiy (2015) ressaltam que a cada interação do usuário com a aplicação, seja ela simples como exibir dados ou complexa como envio de dados, o navegador irá solicitar uma nova página do servidor que será então apresentada para exibição pelo navegador. Para James (2017), usar essa abordagem para sistemas simples é aceitável, entretanto, decidir usar este modelo para sistemas com uma interface mais arrojada e complexa é inadequado. Isso levará a uma degradação na experiência do usuário, pois, a cada solicitação, o servidor precisa gerar uma nova página e transferi-la para o usuário final pela internet, onde, uma vez recebida, ela deve ser renderizada no navegador.

O uso de MPAs ganhou um aprimoramento no início dos anos 2000 com o surgimento do Asynchronous JavaScript and XML (AJAX). Através do AJAX, as aplicações no modelo MPA poderiam atualizar pequenos pedaços do sistema e carregar dados em partes menores. Isso representa um ganho na experiência do usuário, mas também adicionou mais complexidade na criação de aplicações (JAMES, 2017).

MPAs estão diretamente relacionadas com renderização no lado do servidor, que será abordado na seção 2.7.1.

2.5.2 Aplicações de uma página

Aplicações de uma página (SPA) são aplicações que funcionam dentro de um navegador e não requerem recarregamento de páginas durante o uso (NEOTERIC, 2016). São responsáveis por processar os dados recebidos do servidor e transformá-los em informação visual. SPAs surgem em 2009, originando-se da combinação de MPA com AJAX. Após a inicialização do SPA, somente os dados são passados do servidor (JAMES, 2017).

As SPAs devem estar diretamente relacionadas com as boas práticas de experiência de usuário (UX). Através de SPAs os usuários podem dispor de um ambiente que transmita a sensação de fluidez, por não exigir recarregamento de novas páginas (NEOTERIC, 2016).

SPAs estão diretamente relacionadas com renderização no lado do cliente, que será abordado na seção 2.7.2.

2.6 HTML

O HTML (abreviação de Hypertext Markup Language), é uma linguagem de marcação de hypertexto, utilizada para publicação de conteúdo (texto, imagem, vídeo, áudio, por exemplo) para a Web (EIS; FERREIRA, 2012). Surgido nos laboratórios do CERN, é responsável por unificar os documentos na Web, possibilitando que autores não precisassem mais distribuir informação de imagem, sons e textos fragmentados (MUSCIANO; KENNEDY, 2002). Desde o início o HTML foi criado para ser uma linguagem independente de plataforma, browsers e outros meios de acesso (EIS; FERREIRA, 2012).

Silva (2008) e Eis e Ferreira (2012) afirmam que o HTML possuiu até o momento algumas versões: HTML, HTML +, HTML 2.0, HTML 3.0, HTML 3.2, HTML 4.0, HTML 4.01 e HTML 5. Cada versão representa a adição de mudanças para melhorar a linguagem. No decorrer deste processo, o HTML ainda não era tido como um padrão, tornando-se apenas em 1997, instituido pela W3C (World Wide Web Consortium), organização responsável por manter o padrão do código.

Um documento em HTML é composto basicamente por uma tag *html*, que compreende duas outras tags principais o *head* e o *body*. O *head* corresponde ao cabeçalho do documento, em que são definidos parâmetros como o tipo de codificação de texto e o título da página. O *body* corresponde ao corpo do site, em que pode se adicionar uma variedade de tags para representar a informação visível ao acessar uma página na Web. Esta representação básica de um documento em HTML é expressa na Figura 3.

Figura 3 – Exemplo de documento HTML

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <title>Meu Título</title>
</head>
<body>
    Corpo do Site
</body>
</html>
```

2.7 Abordagens de Renderização de Páginas Web

Por muito tempo, utilizando-se da abordagem C/S, o método convencional para apresentação das informações na web era o servidor processar as informações e enviar o documento HTML (este procedimento de geração de documentos HTML é chamado de renderização). Por outro lado, o cliente ficava responsável por simplesmente apresentar as informações providas pelo servidor (VEGA, 2017). Entretanto uma nova maneira de trabalhar documentos HTML no lado Cliente surgiu atualmente. Nas seções a seguir, são descritos os processos de renderização existentes, a renderização no lado do servidor e a renderização no lado do cliente.

2.7.1 Renderização no lado do Servidor

O conceito de Renderização no lado do Servidor se baseia em um navegador operado por um usuário (Cliente), que efetua requisições; e um servidor responsável por analisar e processar requisições. A partir desse processamento, o servidor responde com um documento HTML que será interpretado pelo navegador e apresentado ao usuário solicitante (SCHNEIDER, 2016).

Nesse modelo a construção de documentos HTML é feita através de marcações que funcionam como *placeholders* (espaços reservados) para dados dinâmicos. Estas marcações são traduzidas, em tempo de execução no servidor, para dados voltados para o cliente que disparou a requisição (SCHNEIDER, 2016). A figura 4 demonstra este processo de tradução de dados. Para (VEGA, 2017) este processo pode levar apenas alguns milissegundos, mas dependendo da velocidade de internet, localização do servidor, demanda de usuários utilizando a aplicação e como o site é otimizado, pode levar bem mais tempo que isto.

Figura 4 – Procedimento de tradução de dados dinâmicos para HTML

Apesar de ser a forma tradicional e mais comum de utilização em aplicações para Web, podemos ver em Schneider (2016) e Vega (2017) algumas problemáticas associadas a esta abordagem:

- **Renderização lenta da página**: Uma vez que o servidor processa toda a massa de dados e os traduz em HTML, dependendo da quantidade de dados o tempo para carregamento (renderização) pode ser alto;
- **Excesso de recarregamento**: Cada vez que o usuário necessita navegar em uma outra página da aplicação, uma nova solicitação é feita ao servidor para prover um novo documento HTML;
- **Escalabilidade**: Uma aplicação centrada no servidor é passível a problemas de desempenho. Os usuários poderão encontrar problemas ao interagir com a aplicação se existir uma grande demanda de usuários efetuando chamadas ao servidor, pois o número de sessões armazenadas, pode chegar ao limite suportado;
- **Experiência de usuário**: Uma vez que a cada nova interação do usuário com a aplicação resulta em um novo carregamento, a experiência do usuário é prejudicada pelo recarregamento de toda a página, pois o usuário necessita aguardar o tempo da página ser reconstruída.

2.7.2 Renderização no lado do Cliente

Como visto na seção 2.5.1, o Ajax representou um avanço na construção de páginas para Web, sendo assim, desenvolvedores começaram a construir aplicações inteiramente com essa tecnologia. Disto se origina o conceito de renderização no lado do cliente, em que se têm aplicações sendo renderizadas inteiramente no cliente (SCHNEIDER, 2016). Quando se fala nessa abordagem, está se falando em renderização utilizando JavaScript. Portanto, em vez de obter todo o conteúdo do próprio documento HTML, o usuário obtém um documento HTML básico com um arquivo JavaScript que renderiza o restante do site usando o navegador (VEGA, 2017).

Segundo Vega (2017) a principal diferença desta abordagem, é que se o usuário solicitar carregar mais conteúdo, como navegar para outra página, o navegador não fará outra solicitação de HTML ao servidor. O navegador processará e gerará o novo conteúdo sem o recarregamento total da página. Entretanto, quando o cliente necessita de dados que precisam vir do Servidor, como informações salvas no banco de dados, é feita uma solicitação Ajax ao servidor que retorna estas informações em estruturas mais simples para o Cliente processar e gerar o novo conteúdo.

3 AVALIAÇÃO DE DESEMPENHO

A avaliação de desempenho é um tópico de grande importância em Ciência da Computação. Usuários, administradores e arquitetos de sistemas estão todos interessados na avaliação de desempenho uma vez que seu objetivo é obter ou fornecer o mais alto desempenho pelo menor custo (JAIN, 1990). O campo da avaliação de desempenho envolve tanto a escolha de sistemas, quanto o desenvolvimento de novos sistemas para organizações. Além disso, compreende também a seleção de equipamentos para projetos e análise de sistemas já existentes (JR, 1971).

Um dos grandes desafios no projeto de software é construir soluções de qualidade que atendam a demanda de usuários. Para isso, a avaliação de desempenho é necessária em todos os estágios do ciclo de vida de um sistema. Mesmo que não existam alternativas, a avaliação de desempenho em sistemas existentes ajuda a determinar o desempenho e a necessidade de melhorias (JAIN, 1990).

Jain (1990) afirma que existem diferentes tipos de sistemas com diferentes finalidades o que torna impossível ter uma medida padrão de desempenho, um ambiente de medição padrão ou uma técnica padrão que envolva todos os casos de sistemas. Em Menascé e Almeida (2003) é ressaltada a necessidade de monitorar a intensidade da carga de trabalho, a detecção de gargalos, previsão de limitações de capacidade futuras e determinação do modo mais econômico de se fazer atualizações em sistemas.

A avaliação é geralmente realizada para um propósito específico, e as técnicas empregadas devem ser consideradas à luz dos objetivos do avaliador. Os objetivos da avaliação podem ser maximizar o rendimento do sistema, processar uma determinada carga de trabalho para um custo mínimo ou qualquer número de outras funções objetivo. Essas metas fornecem o ambiente geral para avaliação e determinam qual nível de esforço pode ser dedicado à medição do desempenho (JR, 1971). Nas seções a seguir são explanadas as técnicas de avaliação e o processo de seleção de etapas.

3.1 Técnicas para Avaliação de Desempenho

Através das técnicas de avaliação de desempenho é possível selecionar quais os métodos que melhor exprimem as características do sistema real que se deseja avaliar (ROCHA et al., 2002). Em Jain (1990) existem três técnicas principais para modelagem e previsão de desempenho: modelagem analítica, simulação e medição. Estas são descritas a seguir.

3.1.1 Modelagem Analítica

A modelagem analítica é a técnica que possui o menor custo, pois ela baseia-se principalmente na análise estática (ADHIANTO; CHAPMAN, 2007). Análise estática refere-se a técnicas projetadas para extrair informações de uma versão estática de um sistema. A análise estática é mais eficiente do que as análises realizadas dinamicamente, como o rastreamento de uma execução. Tradicionalmente, as análises estáticas são frequentemente usadas para coletar informações sobre a "modificação, preservação e uso de quantidades de dados" para fins de otimização de código (WANG et al., 2000 apud HECHT, 1977). A modelagem analítica trabalha com suposições do fluxo de controle da aplicação, podendo não levar em consideração o ambiente e tempo de executação (ADHIANTO; CHAPMAN, 2007).

3.1.2 Simulação

Para Bratley et al. (2011) simulação é operar um modelo de um sistema com entradas adequadas e observar as saídas correspondentes. A simulação é uma técnica útil que permite uma abordagem automatizada para avaliar o desempenho de um sistema sob uma variedade de condições (ADHIANTO; CHAPMAN, 2007). Silva et al. (2002) afirmam que simulação tornou-se uma ferramenta muito poderosa para planejamento, projeto e controle de sistemas e é uma metodologia indispensável em projetos de TI. A simulação pode ser empregada tanto em sistemas disponíveis quanto em não disponíveis. Para sistemas não disponíveis, oferece a possibilidade de ter um modelo em que é possível prever o desempenho ou comparar várias alternativas. Para sistemas disponíveis, oferece a possibilidade de comparar alternativas sob uma ampla variedade de cargas de trabalho e ambientes (JAIN, 1990).

3.1.3 Medição

Medição é provavelmente a mais precisa e cara das técnicas. Pode se tornar um método simples ao ser realizada diretamente em um sistema real, no entanto, requer a construção do modelo real ou ao menos a prototipação do modelo a ser avaliado (ROCHA et al., 2002). Em medições, a sobrecarga de instrumentação pode perturbar significativamente os resultados, e grandes arquivos de rastreamento de eventos podem ser gerados. Uma abordagem baseada em um sistema não sofre com esses problemas, mas tem aplicabilidade um tanto limitada (ADHIANTO; CHAPMAN, 2007).

3.2 Etapas da Avaliação de Desempenho

Mesmo que a maioria dos problemas de desempenho sejam únicos e que parâmetros como métricas, a carga de trabalho e as técnicas de avaliação utilizados geralmente não possam ser reaproveitados para um próximo problema, em (JAIN, 1990) são definidas etapas comuns a todos os projetos de avaliação de desempenho, que estão apresentadas nas seções a seguir.

3.2.1 Determinar os Objetivos e Definir o Sistema

A primeira etapa em um projeto de avaliação de desempenho é definir os objetivos do estudo e definir o que constitui o sistema e seus limites. A escolha das fronteiras do sistema afeta as métricas de desempenho, bem como as cargas de trabalho usadas para comparar duas propostas de modelo de sistema.

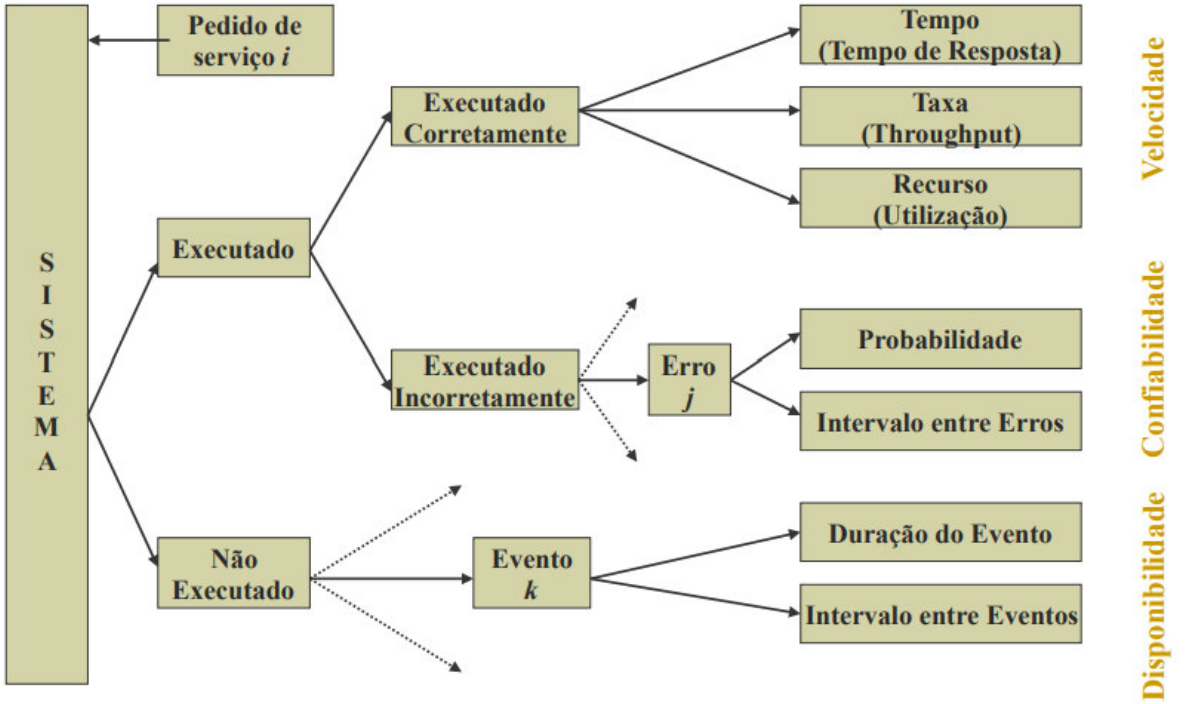
3.2.2 Relação de Serviços e Resultados

Todo sistema fornece ou executa uma lista de serviços. Estes serviços podem prover resultados esperados ou inesperados. Por exemplo, quando um usuário envia um arquivo em um sistema de carregamento de fotos o resultado esperado pode ser o carregamento bem sucedido ou não. Esta etapa visa definir os tipos de serviços oferecidos e prováveis resultados.

3.2.3 Escolha de métricas

O próximo passo é selecionar as métricas para comparar o desempenho. Geralmente, as métricas estão relacionadas à velocidade, precisão e disponibilidade de serviços. Para cada solicitação de serviço feita ao sistema, há vários resultados possíveis. De forma geral, esses resultados podem ser classificados em três categorias: executado corretamente, executado incorretamente e não executado. Estes estados são representados na Figura 5 a seguir.

Figura 5 – Os três possíveis estados de uma solicitação de serviço



Fonte: (JAIN, 1990)

Se o sistema executa o serviço corretamente, seu desempenho é medido pelo tempo gasto para executar o serviço (ou tempo de resposta), a taxa na qual o serviço é executado (ou *throughput*) e os recursos consumidos durante a execução do serviço (utilização).

Se o sistema executar o serviço incorretamente, ocorrerá um erro. É útil classificar os erros e determinar as probabilidades de cada classe de erros.

Se o sistema não executar o serviço, diz-se que ele está inoperante, falhou ou indisponível. Mais uma vez, é útil classificar os modos de falha e determinar as probabilidades de cada classe.

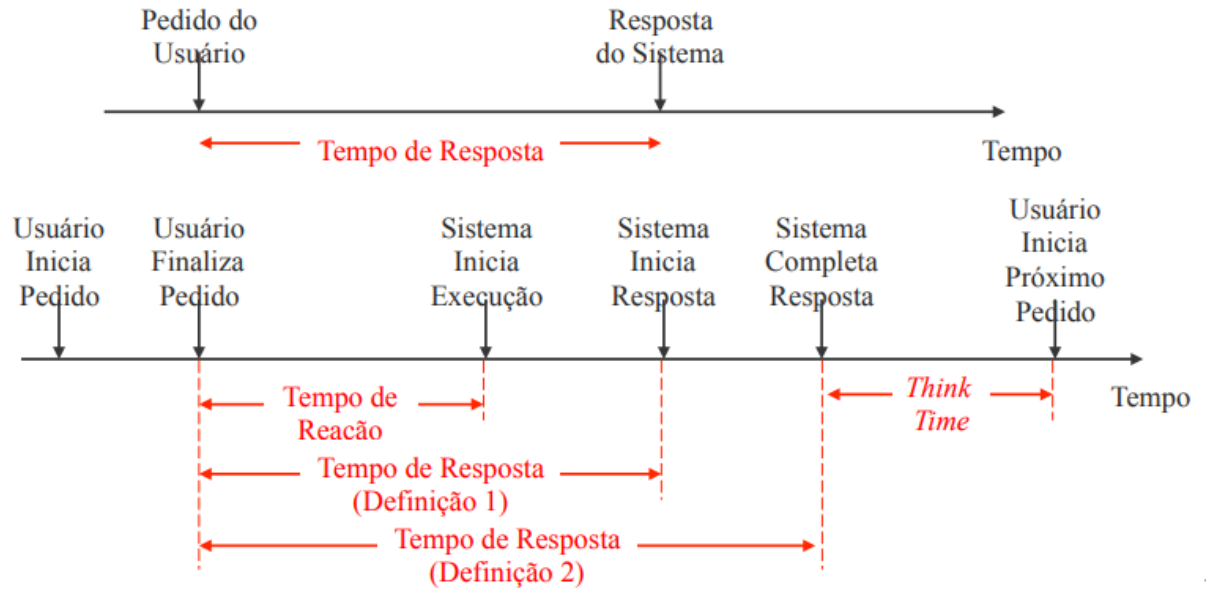
3.2.3.1 Tipos de métricas

Os tipos de métricas são expressos a seguir:

3.2.3.1.1 Tempo de resposta

O tempo de resposta corresponde ao intervalo entre a solicitação do usuário e a resposta do sistema. Este conceito pode ser subdividido como expressa a figura 6. Os usuários levam tempo efetuando a solicitação e o sistema gerando a resposta.

Figura 6 – Representação do fluxo de tempo de resposta



Fonte: (JAIN, 1990)

3.2.3.1.2 Throughput

O throughput é a taxa (solicitações por unidade de tempo) na qual as solicitações podem ser atendidas pelo sistema.

3.2.3.1.3 Eficiência

Eficiência é a razão entre o throughput máximo (capacidade utilizável) e a capacidade nominal do sistema.

3.2.3.1.4 Utilização

É a medida da fração de tempo na qual o recurso está ocupado atendendo uma solicitação.

3.2.3.1.5 Confiabilidade

É normalmente medido pela probabilidade de erros ou pelo tempo médio entre erros.

3.2.3.1.6 Disponibilidade

É a fração de tempo no qual o sistema está disponível para atender as solicitações.

3.2.4 Lista de Parâmetros

Na etapa de listagem de parâmetros é feito um levantamento de todos os parâmetros que afetam o desempenho do sistema. A lista pode ser dividida em parâmetros do sistema e parâmetros de carga de trabalho. Os parâmetros do sistema incluem parâmetros de hardware e software, que geralmente não variam entre as várias instalações do sistema. Os parâmetros de carga de trabalho são característicos das solicitações dos usuários, que variam de uma instalação para outra. A lista de parâmetros pode não estar completa, isso quer dizer que, após o primeiro passo da análise, é possível descobrir que existem parâmetros adicionais que afetam o desempenho.

3.2.5 Seleção de Fatores para Estudo

A lista de parâmetros, levantada no passo anterior, pode ser dividida em duas categorias: os que serão variados durante a avaliação e os que não serão. Os parâmetros a serem variados são chamados de fatores e seus valores são chamados de níveis. Em geral, a lista de fatores e seus possíveis níveis superam a quantidade de recursos disponíveis. Se não fosse assim, a lista continuaria a crescer até se tornar óbvio que não há recursos suficientes para estudar o problema. É preferível iniciar com uma lista pequena de fatores e um número pequeno de níveis para cada fator e estender a lista na próxima fase do projeto, se os recursos permitirem.

Os parâmetros que devem ter um alto impacto no desempenho devem ser preferencialmente selecionados como fatores. Assim como as métricas, um erro comum na seleção dos fatores é que os parâmetros que são fáceis de variar e medir são usados como fatores, enquanto outros parâmetros mais influentes são ignorados simplesmente devido à dificuldade. Ao selecionar fatores, cabe considerar as limitações econômicas, políticas e tecnológicas envolvidas, bem como incluir as limitações impostas pelos controladores de decisão e o tempo disponível para a decisão. Isso aumenta as chances de encontrar uma solução aceitável e implementável.

3.2.6 Seleção de Técnicas de Avaliação

Selecionar uma técnica de avaliação e juntamente com ela selecionar uma métrica são duas etapas principais em toda a avaliação de desempenho de sistemas. A seleção das técnicas, já explanadas em 3.1, exige a observação de alguns critérios discutidos a seguir.

3.2.6.1 Critérios para seleção de técnicas

Há várias considerações que ajudam a decidir a técnica a ser usada. Para isso, há condições a serem consideradas da mais importante para a menos importante, sendo que o fator mais significativo para decidir a técnica a ser empregada é a fase em que está o sistema. Se o fato a ser apurado é um novo conceito, a modelagem analítica e a simulação são as únicas técnicas para escolher, já se o objetivo for uma versão melhorada do sistema a melhor escolha é a medição. Essas considerações estão listadas na Tabela 1.

Tabela 1 – Critérios para selecionar uma técnica de avaliação

	Modelagem Analítica	Simulação	Medição
Etapa	Qualquer	Qualquer	Após prototipação
Tempo necessário	Pouco	Médio	Variável
Ferramentas	Analístas	Linguagens de computação	Instrumentação
Precisão	Pouca	Moderada	Variável
Comparações	Fácil	Moderada	Difícil
Custo	Pequeno	Médio	Alto
Poder de convencimento	Baixa	Média	Alta

A etapa do ciclo de vida em que o sistema se encontra é a principal consideração a ser tomada. A medição somente é possível quando algo similar ao sistema proposto já exista, como quando se deseja melhorar sua versão. Se for um projeto novo, somente se pode escolher modelagem analítica ou simulação. A modelagem analítica e a simulação podem ser usadas para situações em que a medição não é possível. Entretanto, em geral seria mais convincente se a modelagem analítica ou simulação fossem baseada em medições prévias.

Outro fator a ser considerado é o tempo disponível para avaliação. Na maioria das situações, os resultados são necessários para o menor prazo possível. Nestes casos a modelagem analítica é provavelmente a única escolha, pois as simulações levam bastante tempo e as medições, mesmo que geralmente levem menos tempo que as simulações, também demandam uma quantia de tempo superior a modelagem analítica.

Um outro fator é a disponibilidade de ferramentas para análise. As ferramentas incluem habilidades de modelagem, linguagens de simulação e instrumentos de medição.

O nível de precisão desejado é outra consideração importante. Geralmente, a modelagem analítica exige tantas simplificações e suposições que, resultados inesperados podem surpreender os analistas. Simulações frequentemente estão mais próximas da realidade, pois podem agregar mais detalhes exigindo menos suposições do que a modelagem analítica. Medições podem não fornecer resultados precisos, apesar de parecerem dados reais. Isso se dá simplesmente porque muitos dos parâmetros ambientais, como configuração do sistema, tipo de carga de trabalho e tempo de medição, podem ser exclusivos do experimento. Além disso, os parâmetros podem não representar o intervalo de variáveis do mundo real. Assim, a precisão dos resultados pode variar de muito alta a nenhuma.

O propósito de todo estudo de desempenho é comparar diferentes alternativas ou encontrar o valor ideal dos parâmetros. O Modelo analítico geralmente fornece a melhor percepção dos efeitos de vários parâmetros e suas interações. Pelas simulações, pode ser possível procurar o espaço de valores de parâmetros para a combinação ótima, porém muitas vezes não está claro qual é o trade-off entre os diferentes parâmetros. A medição é a técnica menos desejável a esse respeito. Não é fácil saber se o desempenho aprimorado é resultado de alguma alteração aleatória no ambiente ou devido à configuração específica de um parâmetro.

O custo alocado para o projeto também é importante. A medição é a forma mais cara das três técnicas, pois requer equipamentos, instrumentos e tempo real. Em sistemas muito caros a simulação é uma boa alternativa pela facilidade de alteração de configurações. Modelagem analítica somente requer papel, lápis e o tempo do analista, sendo essa a mais barata das alternativas.

O poder de convencimento é provavelmente a justificativa principal quando se considera a despesa e o trabalho das medições. É muito mais fácil convencer pessoas com uma medida real. A maioria das pessoas é pessimista quanto a resultados analíticos simplesmente porque não entendem a técnica ou o resultado final. Na verdade, as pessoas que desenvolvem novas técnicas de modelagem analítica frequentemente as validam usando simulações ou medições reais.

Para alguns casos é útil usar duas ou mais técnicas simultaneamente, pois todos os resultados de uma avaliação são considerados a princípio suspeitos. Isso implica em três regras de validação:

- Validar um modelo de simulação através de modelo analítico ou medição;
- Validar um modelo analítico através de simulação ou medição e
- Validar uma medição através de um modelo analítico ou simulação.

3.2.7 Seleção de Carga de Trabalho

A carga de trabalho consiste em uma lista de solicitações de serviço para o sistema. Dependendo da técnica de avaliação escolhida, pode ser expressa em diferentes formas. Na modelagem analítica, a carga de trabalho é geralmente uma probabilidade de várias solicitações. Na simulação, pode-se usar um rastreamento de solicitações medidas em um sistema real. Na medição, a carga de trabalho consiste em *scripts* de usuário a serem executados no sistema. Em todos os casos, é essencial que a carga de trabalho seja representativa do uso do sistema na vida real.

3.2.8 Projeto de Experimentos

Após ter uma lista de fatores e seus níveis, é necessário decidir os experimentos que oferecem o máximo de informações com o mínimo de esforço. Na prática, é útil realizar um experimento em duas fases: uma com maior número de fatores e com menor número de níveis, outra com o número de fatores reduzido e o número de níveis relevantes aumentado.

3.2.9 Análise e Interpretação dos Dados dos Resultados

É importante reconhecer que os resultados das medições e simulações são valores aleatórios, já que os resultados são diferentes cada vez que o experimento é repetido. Ao comparar duas alternativas, é necessário levar em conta a variabilidade dos resultados. Simplesmente comparar os meios pode levar a conclusões imprecisas, portanto interpretar os resultados de uma análise é fundamental. Análises só produzem resultados e não conclusões. Estes fornecem a base sobre a qual chega-se às conclusões.

3.2.10 Apresentação de Resultados

A etapa final é comunicar o resultados, através destes resultados serão feitas as tomadas de decisão. É importante que os resultados sejam claros e de fácil compreensão. Uma boa forma de apresentação é através de meios gráficos dimensionados adequadamente. Muitas vezes, nesse ponto do projeto, o conhecimento adquirido pelo estudo pode exigir que o analista volte atrás e reconsidere algumas das decisões tomadas nas etapas anteriores como redefinição dos limites do sistema ou inclusão de outros fatores e métricas de desempenho que não foram considerados anteriormente. O projeto completo, portanto, consiste em vários ciclos através das etapas, em vez de um único passo seqüencial.

4 COMPARAÇÃO DAS ABORDAGENS DE RENDERIZAÇÃO

Este capítulo é destinado a apresentar um experimento que compara o desempenho de duas abordagens: uma de renderização na camada de Servidor e outra de renderização na camada de Cliente. Inicialmente são apresentadas as tecnologias envolvidas neste experimento, tais como linguagem de programação, bibliotecas, banco de dados e plataforma de hospedagem. Logo após é realizada uma descrição geral do experimento.

4.1 Tecnologias utilizadas

Nesta seção são apresentadas as tecnologias utilizadas neste experimento.

4.1.1 Javascript

Segundo Flanagan (2007) Javascript é uma linguagem de programação referência da Web atual. Ela é suportada em todos os navegadores modernos e atualizados e está presente na maioria das aplicações para Web. Sendo que atualmente não está limitada apenas ao Cliente, mas hoje já está presente em aplicações para computador, aplicações para dispositivos móveis, jogos para console, processamento de dados no servidor, entre outros. De fato, é uma linguagem fundamental e onipresente em todos os aspectos do mercado de desenvolvimento de softwares atualmente.

4.1.2 Node.js

O Node.js, ou simplesmente Node, é um ambiente de execução de JavaScript no lado do servidor (JOYENT, 2014). É baseado na implementação do mecanismo V8 do Google. O V8 e o Node são geralmente implementados em C e C++, com foco em desempenho e em baixo consumo de memória. O Node tem como objetivo suportar processos de servidor de longa duração. Ao contrário da maioria dos outros ambientes modernos, um processo Node não depende de multithreading para suportar a execução simultânea da lógica de negócios; é baseado em um modelo de eventos de entrada e saída assíncronas (TILKOV; VINOSKI, 2010).

4.1.3 React

O React é uma biblioteca JavaScript de interface de usuário desenvolvida pelo Facebook. Ela trabalha através de manipulação do DOM (Document Object Model) do navegador, organizando o fluxo de dados da aplicação e considerando os elementos da interface do como

componentes individuais. O React, por ser uma biblioteca, não impacta no restante da pilha de tecnologias do Front-end, podendo ser usado isoladamente para apenas um trecho da aplicação (FEDOSEJEV, 2015).

4.1.4 Embedded JavaScript templates

Embedded JavaScript templates, ou simplesmente EJS, é um mecanismo de templates simplificados que permite geração de marcação HTML com JavaScript (RILEY, 2015). Através do EJS o desenvolvedor constrói template de páginas HTML com *placeholders* a serem substituídos por dados. O resultado final é o documento HTML gerado.

4.1.5 MongoDB

Segundo Banker (2011), o MongoDB é um sistema de gerenciamento de banco de dados não-relacional projetado para aplicativos Web e infraestrutura da Internet. O modelo de dados e as estratégias de persistência são criados para alta taxa de transferência de leitura e gravação e a capacidade de escalar facilmente. É indicado tanto para aplicativos que necessitem apenas de uma instância de banco de dados quanto para os que necessitam de várias instâncias.

4.1.6 Mongoose

O Mongoose é um pacote do Node.js utilizado para modelagem de documentos para o MongoDB. Ele fornece uma solução direta e baseada em esquema para modelar os dados de aplicações, incluindo conversão de tipo integrada, validação, construção de consulta, ganchos de lógica de negócios entre outras funcionalidades (MONGOOSE, 2011).

4.1.7 Heroku

O Heroku é uma plataforma em nuvem que permite que empresas e desenvolvedores construam, entreguem, monitorem e dimensionem aplicativos (HEROKU, 2011). Tem por objetivo contornar as problemáticas de implementação de infraestrutura.

4.1.8 Google Developer Tools - Performance

O Google Developer Tools, disponível no navegador Chrome, disponibiliza uma aba para monitorar o desempenho de uma página Web. Além disso, esta ferramenta possibilita realizar testes simulando conexão 3G ou CPU com hardware mais lento que o computador utilizado.

4.2 Descrição do Experimento

A descrição deste experimento está dividida nas seções a seguir. Primeiramente é apresentada a medotologia utilizada para avaliação. Em seguida são abordados aspectos técnicos do experimento referentes ao modelo de banco de dados e estrutura da aplicação construída.

4.2.1 Metodologia de avaliação

O procedimento deste experimento é baseado no modelo proposto por Jain (1990) na seção 3.2 e utiliza a técnica de aferição. Para melhor entendimento, esta seção está divida nos tópicos que se seguem. Primeiramente, uma observação do objetivo do experimento, compreendendo o serviço prestado e o resultado esperado. Em seguida, são apresentadas as métricas a serem consideradas na avaliação, bem como os parâmetros de medição.

4.2.1.1 Objetivo do Experimento

O experimento tem como objetivo avaliar o desempenho de duas abordagens de renderização para Web, através da medição do tempo necessário para carregar uma massa de dados. Estes dados estão salvos em um banco de dados não relacional, contendo informações básicas de usuários. O serviço prestado neste experimento é apresentar uma única página contendo uma massa de dados de usuários, sendo apresentados o nome, o email e uma foto, consequentemente o resultado esperado é a página contendo estas informações.

4.2.1.2 Métricas e parâmetros considerados

Para avaliação das abordagens será utilizada a métrica do tempo necessário para carregamento das informações. Essa métrica se subdivide em tempo para execução dos scripts e tempo para renderização do HTML.

Os principais parâmetros levantados que podem impactar na medição das métricas necessárias para a avaliação, são:

- **Tipo de conexão**: serão considerados dois tipos de conexão: conexão ethernet e conexão 3G;
- **Velocidade da CPU**: serão considerados dois tipos de CPU: CPU rápida e CPU lenta;
- **Quantidade de registros**: serão consideradas duas massas de dados: cinco mil registros e vinte e cinco mil registros;

Como visto na seção 4.1.8 a ferramenta de medição de desempenho do Google Developer Tools possibilita simular uma CPU mais lenta e conexão 3G. É fato que usuários possuem diferentes computadores com muito ou pouco poder computacional e conexão com a Internet variada. De fato, a combinação de parâmetros otimizados com parâmetros não-otimizados, podem gerar resultados distintos neste experimento. Sendo assim, foram desenvolvidos oito cenários com diferentes combinações dos parâmetros de tipo de conexão, velocidade da CPU e massa de dados. Os oito cenários são apresentados na tabela 2:

Tabela 2 – Cenários de teste de desempenho

	Tipo de Conexão	Velocidade de CPU	Quantidade de registros
Cenário 1	Conexão Ethernet	CPU Rápida	5.000
Cenário 2	Conexão Ethernet	CPU Lenta	5.000
Cenário 3	Conexão Ethernet	CPU Rápida	25.000
Cenário 4	Conexão Ethernet	CPU Lenta	25.000
Cenário 5	Conexão 3G	CPU Rápida	5.000
Cenário 6	Conexão 3G	CPU Lenta	5.000
Cenário 7	Conexão 3G	CPU Rápida	25.000
Cenário 8	Conexão 3G	CPU Lenta	25.000

Cada um dos oito cenários de teste será empregado nas duas abordagens deste experimento descritas na seção 4.2.4.

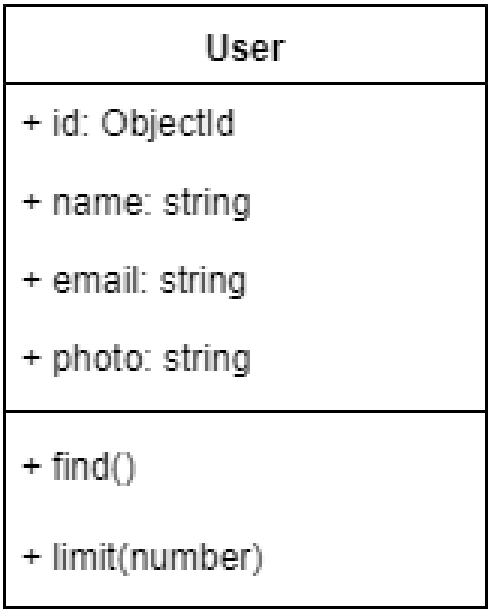
4.2.2 Aspectos técnicos do experimento

Nesta seção são abordados aspectos técnicos do experimento.

4.2.2.1 Modelagem das informações

Para o experimento, foi considerado uma estrutura de usuário contendo os campos: *id*, *name*, *email* e *photo*. Sendo que o campo de *photo* é uma URL para uma imagem externa disponibilizada pelo site *Random User*. O pacote Mongoose, apresentado na seção 4.1.6, disponibiliza alguns métodos para manipulação das consultas ao banco de dados, os métodos utilizados neste trabalho são: *find* e *limit*. Uma representação da estrutura da coleção de usuários pode ser observada na figura 7.

Figura 7 – Representação da coleção de Usuários



O método *find* possibilita encontrar registros que correspondam aos parâmetros solicitados, entretanto para este experimento não se faz necessário estes parâmetros adicionais. Sendo assim, utilizar este método sem parâmetros implica em retornar todos os registros salvos no banco de dados. O método *limit*, possibilita limitar a quantidade de registros retornados pelo método *find*.

Os dados recuperados são postos em uma lista estruturada de objeto JSON. A Figura 8 representa um objeto JSON de um usuário de exemplo.

Figura 8 – Representação do objeto Usuário

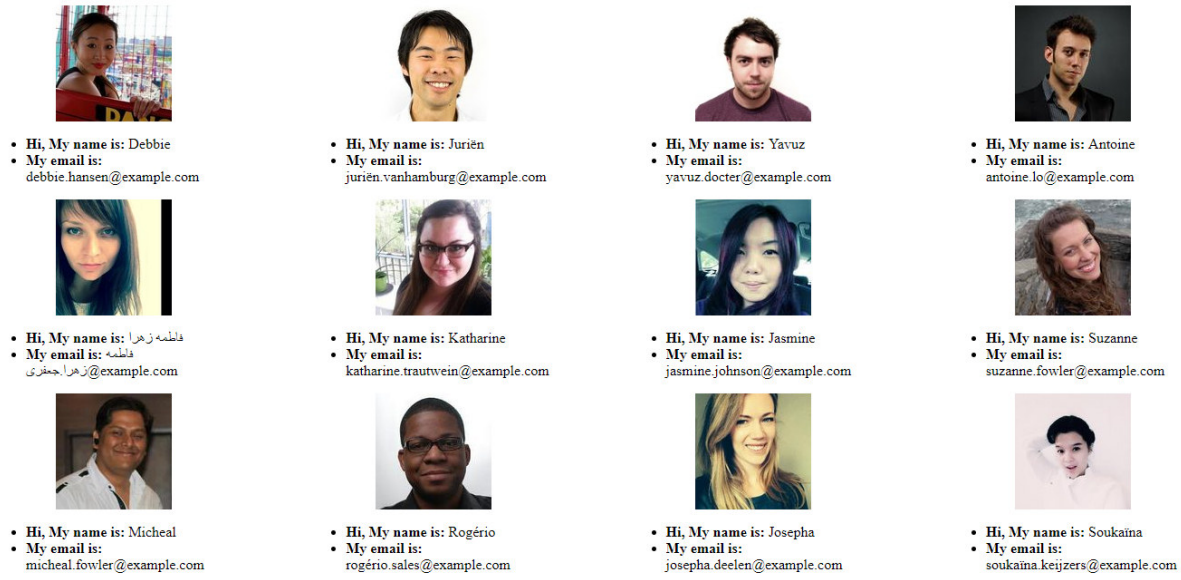

```
{
    "id":"5b2fee08b11f162d64681812",
    "name":"Shayan",
    "email":"shayan.plantinga@example.com",
    "photo":"https://randomuser.me/api/portraits/men/30.jpg"
}
```


4.2.3 Representação das informações

Para apresentação das informações recuperadas da base de dados, foi utilizada uma representação simples de um documento HTML. Para organizar e prover uma visualização

apresentável as informações foram distribuídas em linhas e colunas. Estilizações mais arrojadas foram dispensadas, pois estruturação de páginas não faz parte deste estudo. A apresentação dos dados pode ser observada na figura 9.

Figura 9 – Apresentação estruturada das informações recuperadas



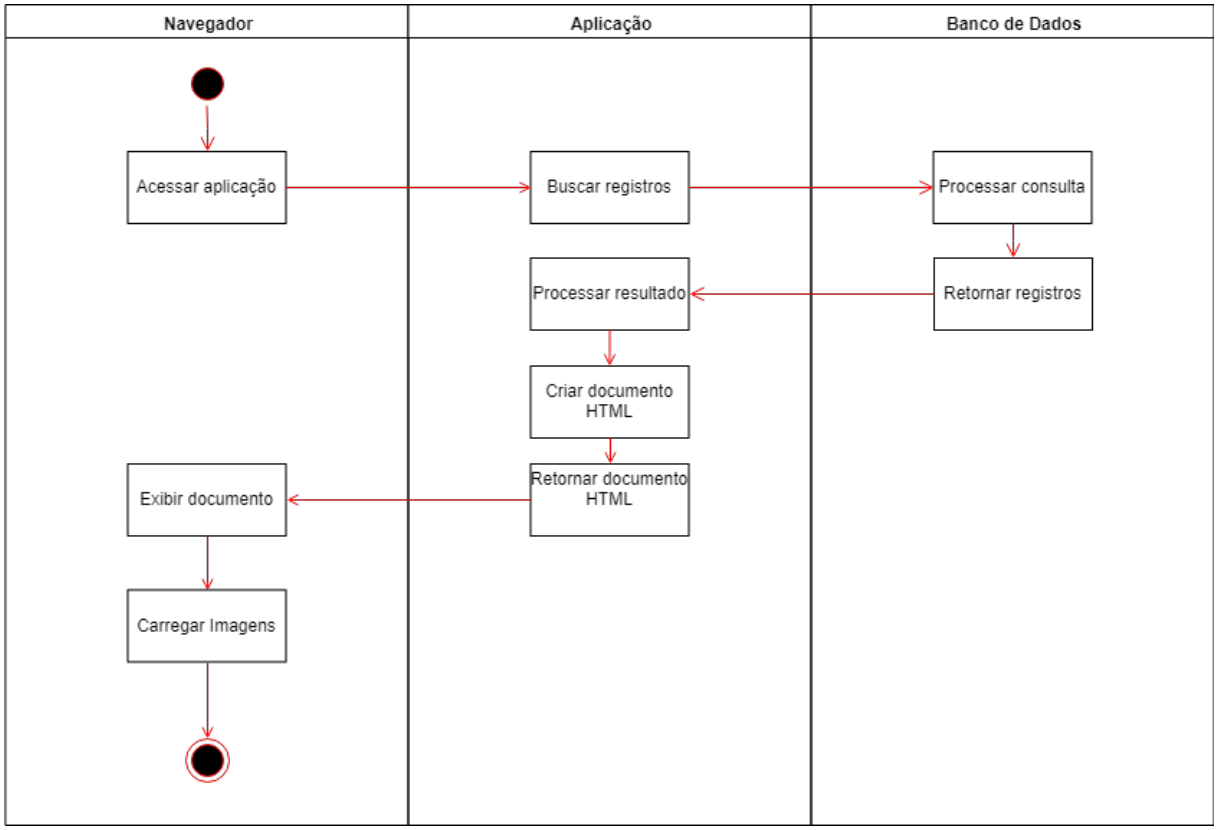
4.2.4 Abordagens

Para realizar este experimento são necessárias duas abordagens de renderização para Web. Nos tópicos que seguem estas abordagens são explanadas.

4.2.4.1 Abordagem 1: Renderização Servidor

Primeiramente o navegador acessa a aplicação, esta por sua vez busca os registros salvos no banco de dados, que os processa e os retorna para a aplicação. A aplicação processa os resultados do banco de dados, cria o documento HTML e o envia como resposta da solicitação do navegador. O navegador, por fim exibe o documento HTML resultante e carrega o conjunto de imagens. Os passos executados são expressos na figura 10:

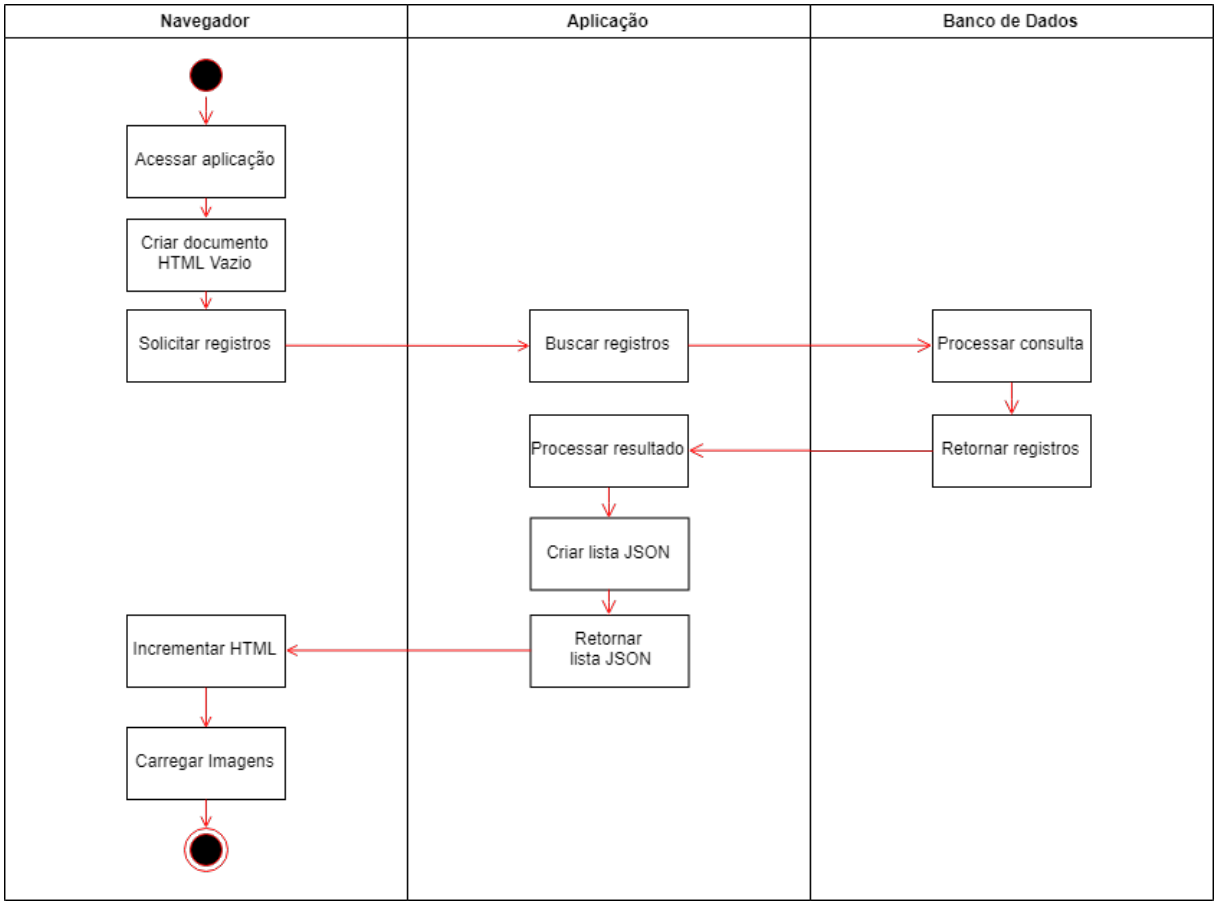
Figura 10 – Diagrama de atividade da abordagem 1



4.2.4.2 Abordagem 2: Renderização Cliente

Primeiramente o navegador acessa a aplicação, que gera um documento HTML mínimo vazio. Em seguida a aplicação solicita, em segundo plano, dados ao servidor, que acessa o banco de dados. O banco de dados processa e retorna os registros para o servidor. O servidor processa os resultados, transforma a lista obtida do banco de dados em uma lista de objetos JSON e os retorna para a aplicação. A aplicação obtém esta lista e incrementa o documento HTML existente. Por fim a aplicação carrega o conjunto de imagens. Os passos executados são expressos na figura 11:

Figura 11 – Diagrama de atividade da abordagem 2



5 RESULTADOS

Este capítulo trata de apresentar os resultados obtidos em cada cenário do experimento proposto no capítulo 4, bem como discutir características pontuais de cada cenário.

5.1 Valores coletados de cada cenário

A tabela 3 apresenta os valores médios de tempo de resposta obtidos ao executar os testes em cada um dos cenários descritos na seção 4.2.1.2 para as abordagens de renderização no lado do servidor (abordagem 1) e renderização no lado do cliente (abordagem 2).

Tabela 3 – Tempos médios coletados para cada cenário por abordagem (em milisegundos)

	Abordagem 1	Abordagem 2
Cenário 1	10.786	6.253
Cenário 2	36.422	26.315
Cenário 3	50.532	24.816
Cenário 4	340.264	116.106
Cenário 5	129.327	62.917
Cenário 6	136.155	89.022
Cenário 7	426.070	108.707
Cenário 8	472.786	116.964

Para o cenário 1, com conexão Ethernet, CPU rápida e 5.000 registros, é possível observar que a abordagem 2 apresentou um menor tempo para carregar os resultados, tendo uma diferença média de 4.533 milisegundos.

No Segundo cenário, caracterizado por conexão Ethernet, CPU lenta e 5.000 registros, a abordagem 2 apresenta menor tempo para apresentação dos resultados obitidos, com uma diferença média de 10.107 milisegundos.

No terceiro cenário, com conexão Ethernet, CPU rápida e uma maior carga de registros (25.000 no total), a abordagem 2 novamente precisou de menos tempo para apresentar os registros. Sendo a diferença entre a abordagem 1 e a abordagem 2 de 25.716 milisegundos.

O cenário 4, com conexão Ethernet, CPU lenta e 25.000 registros. Neste cenário, novamente a abordagem 2 apresenta melhor tempo de carregamento das informações em relação a abordagem 1. A diferença de tempo neste cenário é de 224.158 milisegundos.

Com conexão 3G, CPU rápida e 5.000 registros, o cenário 5, a abordagem 2 mais uma vez apresentou resultados mais satisfatórios em relação a abordagem 1. Neste cenário a diferença média de tempo entre cada abordagem é de 66.410 milisegundos.

O sexto cenário, apresenta conexão 3G, CPU lenta e 5.000 registros. Os valores coletados neste cenário novamente apresentam resultados satisfatórios para a abordagem 2. A diferença média de milisegundos neste cenário é de 47.133.

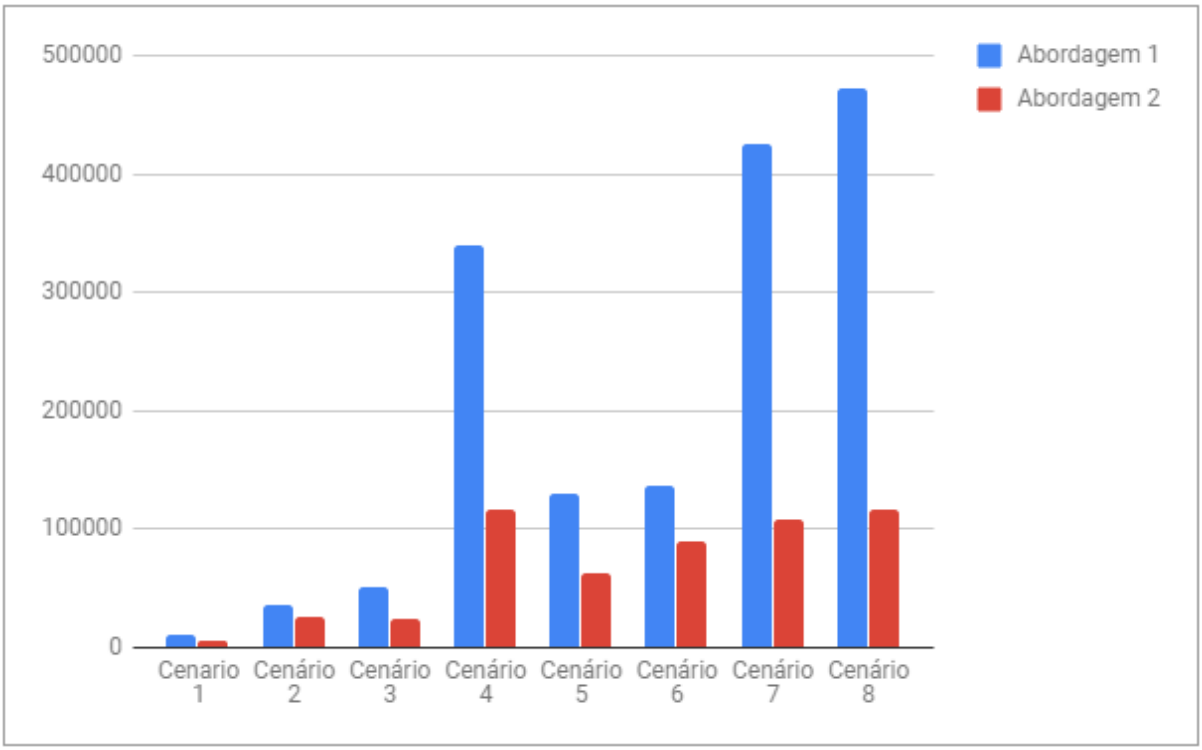
No sétimo e penúltimo cenário, com conexão 3G, CPU rápida e 25.000 registros, a abordagem 2 possui menor tempo em cada um dos testes. A diferença de tempo médio neste cenário é de 317.363 milisegundos.

O oitavo e último cenário, com as características de conexão 3G, CPU lenta e 25.000 registros, a abordagem 2 apresenta mais uma vez o menor tempo de apresentação em todos os testes. A diferença média entre as abordagens neste cenário é de 355.822 milisegundos.

5.2 Discussão dos resultados

Observando o resultado presente em 5.1, é possível constatar que a abordagem 2 apresentou melhores resultados em todos os cenários. A figura 12 apresenta um comparativo geral do experimento. Nesta figura são apresentados lado a lado os valores médios de cada abordagem em cada cenário.

Figura 12 – Valor médio de cada cenário



É interessante observar que no cenário 1, em que ambas abordagens possuem características ótimas a diferença de tempo é mínima. Sendo assim, pode se concluir apesar da leve vantagem

da abordagem 2 que os resultados são equivalentes. Por outro lado, o cenário 8 apresenta o pior cenário e a diferença de tempo entre as abordagens é expressiva.

O cenário 4 apresenta um resultado peculiar. Possuindo duas características de estresse da aplicação, CPU lenta e carga expressiva de registros, apresenta resultados próximos dos cenários 7 e 8, principalmente para a abordagem 2. Apesar de estar com conectividade melhor, na abordagem 2 o que é transportado na rede é um documento JSON e este apresenta um tamanho relativamente pequeno. Sendo assim o que é mais determinante é o poder de processamento da CPU.

Nos cenários 5 e 6, observa-se que para a abordagem 1 os resultados são bem próximos. possuindo diferença média de 6.828 milisegundos. Por outro lado, para a abordagem 2 os resultados apresentam um distanciamento mais significativo, sendo a diferença média de 26.105 milisegundos.

O experimento demonstra que o desempenho da abordagem 1 é fortemente impactado pela quantidade de registros, entretanto os outros fatores também resultam em dados ao desempenho. Por outro lado na abordagem 2, o fator impactante é o poder de processamento da CPU.

É importante frisar que os testes foram realizados em um computador pessoal, sujeito a picos de processamento de CPU por processos em segundo plano. Além disso, a conexão com a internet pode variar no decorrer do dia e depende diretamente das condições de infraestrutura da rede. Isso quer dizer que dependendo das condições dos cenários de teste os tempos de renderização para cada abordagem podem sofrer alterações para mais ou para menos.

6 CONCLUSÃO

Neste trabalho é feito um comparativo de abordagens funcionais e com características diferentes. Contudo, este trabalho não pretende descartar a utilização da abordagem de renderização do lado servidor. Pois esta se demonstra útil, com grande aderência por parte dos desenvolvedores e utilizada por grandes corporações. Entretanto a abordagem de renderização do lado cliente apresentou-se mais eficaz para apresentação de páginas Web. Esta eficiência é resultado da utilização de documentos JSON e com o melhor da tecnologia Javascript.

Por meio deste estudo fica claro que para a abordagem de renderização do lado do cliente é importante que o desenvolvedor considere o hardware utilizado pelos usuários. Em situações em que há significativa carga de registros a ser recuperada, o desempenho é prejudicado.

Através do Google Developer Tools foi possível realizar uma simulação de um computador com CPU inferior e conexão 3G. Sendo assim, para trabalhos futuros, poderia ser utilizado um segundo computador com CPU mais lenta. Outra situação possível seria efetuar os testes em uma rede 3G verdadeira. Por fim, outra possibilidade de estudo seria realizar testes simultâneos com vários computadores com diferentes velocidades de CPU.

REFERÊNCIAS

ACADEMY, C. **Back-End Web Architecture**. 2018. Acesso em 05/05/2018. Disponível em: <https://www.codecademy.com/articles/back-end-architecture>.

ADHIANTO, L.; CHAPMAN, B. Performance modeling of communication and computation in hybrid mpi and openmp applications. **Simulation Modelling Practice and Theory**, Elsevier, v. 15, n. 4, p. 481–491, 2007.

ALMEIDA, J. M. F. d. Breve história da internet. Universidade do Minho. Departamento de Sistemas de Informação, 2005.

BANKER, K. **MongoDB in action**. [S.l.]: Manning Publications Co., 2011.

BERNERS-LEE, T. Longer biography. **Dostupné z: http://www. w3. org/People/Berners-Lee/Longer. html**, 2006.

BRATLEY, P.; FOX, B. L.; SCHRAGE, L. E. **A guide to simulation**. [S.l.]: Springer Science & Business Media, 2011.

BRYAN, G. E. The full cost of client/server computing. In: IEEE. **Aerospace Applications Conference, 1995. Proceedings., 1995 IEEE**. [S.l.], 1995. v. 1, p. 137–152.

CITRUS7. **O QUE É FRONT-END E BACK-END?** 2014. Acesso em 05/05/2018. Disponível em: <https://citrus7.com.br/artigo/o-que-e-front-end-e-back-end/>.

COMER, D. **Interligação de Redes com TCP/IP–Vol. 1: Princípios, Protocolos e Arquitetura**. [S.l.]: Elsevier Brasil, 2017. v. 6.

EIS, D.; FERREIRA, E. **HTML5 e CSS3 com farinha e pimenta**. [S.l.]: Lulu. com, 2012.

ELMASRI, R. **Fundamentals of database systems**. [S.l.]: Pearson Education India, 2008.

EMERY, C. **A Brief History of Web Development**. 2016. <https://www.techopedia.com/2/31579/networks/a-brief-history-of-web-development>. Acesso em 05/05/2018.

FEDOSEJEV, A. **React. js Essentials**. [S.l.]: Packt Publishing Ltd, 2015.

FILETO, R. **Sistemas Cliente-Servidor**. 2016. Notas de Aula.

FLANAGAN, D. **JavaScript: O guia definitivo**. [S.l.]: Bookman Editora, 2007.

GALLAUGHER, J. M.; RAMANATHAN, S. C. Choosing a client/server architecture. **Information Systems Management**, v. 13, n. 2, p. 7–13, 1996.

HECHT, M. S. **Flow analysis of computer programs**. [S.l.]: Elsevier Science Inc., 1977.

HEROKU, I. What is heroku? **Stand**, v. 30, p. 2011, 2011.

HILL, A. L.; FRISBIE, R. L.; MOORE, E. J.; KING, H. S.; POTTS, D. B. **Integrated software development and deployment architecture and high availability client-server systems generated using the architecture**. [S.l.]: Google Patents, 2017. US Patent 9,817,657.

JAIN, R. **The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling**. [S.l.]: John Wiley & Sons, 1990.

JAMES, C. **Choosing a web application design pattern**. 2017. Acesso em 06/05/2018. Disponível em: <https://www.deloittedigital.com.mt/en/news/2017/7/choosing-a-web-application-design-pattern-414>.

JOYENT, I. **About node. js**. [S.l.]: NodeJS, 2014.

JR, H. L. Performance evaluation and monitoring. **ACM Computing Surveys (CSUR)**, ACM, v. 3, n. 3, p. 79–91, 1971.

LINDGREN, L. E. **Application Servers for E-business**. [S.l.]: CRC Press, 2001.

M'BAYA, A.; LAVAL, J.; MOALLA, N. An assessment conceptual framework for the modernization of legacy systems. In: IEEE. **Software, Knowledge, Information Management and Applications (SKIMA), 2017 11th International Conference on**. [S.l.], 2017. p. 1–11.

MENASCÉ, D. A.; ALMEIDA, V. A. Planejamento de capacidade para serviços na web: Métricas, modelos e métodos. **Rio de Janeiro: Campus**, 2003.

MITCHELL, B. **Servers Are The Heart and Lungs of the Internet**. 2018. Acesso em 05/05/2018. Disponível em: <https://www.lifewire.com/servers-in-computer-networking-817380>.

MIURA, T. Practical use of client/server system in business process re-engineering. In: IEEE. **Computer Software and Applications Conference, 1994. COMPSAC 94. Proceedings., Eighteenth Annual International**. [S.l.], 1994. p. 126–131.

MONGOOSE. **Mongoose ODM**. 2011. Acesso em 23/05/2018. Disponível em: <http://mongoosejs.com/>.

MUSCIANO, C.; KENNEDY, B. **HTML & XHTML: The Definitive Guide: The Definitive Guide**. [S.l.]: "O'Reilly Media, Inc.", 2002.

NEOTERIC. **Single-page application vs. multiple-page application**. 2016. Acesso em 06/05/2018. Disponível em: <https://medium.com/@NeotericEU/single-page-application-vs-multiple-page-application-2591588efe58>.

RILEY, X. **What is EJS?** 2015. Acesso em 23/05/2018. Disponível em: <http://ejs.co>.

ROCHA, C. A. d. S. et al. Análise de desempenho em ambientes cliente/servidor 2-camadas e 3-camadas. Florianópolis, SC, 2002.

SCHNEIDER, A. H. Desenvolvimento web com client side rendering: combinando single page application e serviços de backend. 2016.

SILVA, M. S. **Criando sites com HTML: sites de alta qualidade com HTML e CSS**. [S.l.]: Novatec Editora, 2008.

SILVA, M. V. D. et al. Avaliação de desempenho de uma plataforma de comutação telefônica para serviços especializados de atendimento ao cliente. Florianópolis, SC, 2002.

STRICKLAND, J. **How Cloud Computing Works**. 2008. Acesso em 05/05/2018. Disponível em: <https://computer.howstuffworks.com/cloud-computing/cloud-computing1.htm>.

TARNOVSKIY, S. **Mixing MPA and SPA: worst of both worlds**. 2015. Acesso em 06/05/2018. Disponível em: <https://blogs.perficient.com/2015/01/26/mixing-mpa-and-spa-worst-of-both-worlds/>.

TILKOV, S.; VINOSKI, S. Node. js: Using javascript to build high-performance network programs. **IEEE Internet Computing**, IEEE, v. 14, n. 6, p. 80–83, 2010.

VEGA, J. **Client-side vs. server-side rendering: why it's not all black and white**. 2017. <https://medium.freecodecamp.org/what-exactly-is-client-side-rendering-and-hows-it-different-from-server-side-rendering-bd5c786b340d>. Acesso em 15/05/2018.

WANG, C.; HILL, J.; KNIGHT, J.; DAVIDSON, J. **Software tamper resistance: Obstructing static analysis of programs**. [S.l.], 2000.

WODEHOUSE, C. **A Beginner's Guide to Back-End Development**. 2017. Acesso em 05/05/2018. Disponível em: <https://www.upwork.com/hiring/development/a-beginners-guide-to-back-end-development/>.

WODEHOUSE, C. **A Beginner's Guide to Front-End Development**. 2017. Acesso em 05/05/2018. Disponível em: <https://www.upwork.com/hiring/development/beginners-guide-to-front-end-development/>.