UNIVERSIDADE FEDERAL DO CEARÁ CAMPUS QUIXADÁ
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

DOUGLAS HENRIQUE MENDES DOS SANTOS

PROCESSO PARA MIGRAÇÃO DE APLICAÇÕES MULTI-PAGE PARA SINGLE-PAGE

QUIXADÁ

2016

DOUGLAS HENRIQUE MENDES DOS SANTOS

PROCESSO PARA MIGRAÇÃO DE APLICAÇÕES MULTI-PAGE PARA SINGLE-PAGE

Trabalho de Conclusão de Curso submetido à Coordenação do Curso Bacharelado em Sistemas de Informação da Universidade Federal do Ceará como requisito parcial para obtenção do grau de Bacharel.

Área de concentração: computação

Orientador Prof. Régis Pires Magalhães

QUIXADÁ
2016

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

S234p Santos, Douglas Henrique Mendes dos.
Processo para migração de aplicações Multi-page para Single-page / Douglas Henrique Mendes dos Santos. – 2016.
43 f.: il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Sistemas de Informação, Quixadá, 2016.
Orientação: Prof. Me. Regis Pires Magalhães.

1. Software - desenvolvimento. 2. Sites da Web. 3 Aplicações Web. 4. Aplicação Single-page. 5. Front-end e back-end. I. Título.

CDD 005

DOUGLAS HENRIQUE MENDES DOS SANTOS

PROCESSO PARA MIGRAÇÃO DE APLICAÇÕES MULTI-PAGE PARA SINGLE-PAGE

Trabalho de Conclusão de Curso submetido à Coordenação do Curso Bacharelado em Sistemas de Informação da Universidade Federal do Ceará como requisito parcial para obtenção do grau de Bacharel. Área de concentração: computação

Aprovado em: _____ / __ / 2016.

BANCA EXAMINADORA

_____

Prof. Msc. Regis Pires Magalhães (Orientador)
Universidade Federal do Ceará-UFC

_____

Prof. Msc. Camilo Camilo Almendra
Universidade Federal do Ceará-UFC

_____

Prof. Msc. Bruno Góis Mateus
Universidade Federal do Ceará-UFC

Aos meus pais, irmão, amigos e professores...

"Quem esquece de onde veio, não sabe para onde vai."

(Bráulio Bessa)

RESUMO

Com o aumento de usuários conectados via internet, a demanda por aplicações Web mais rápidas, interativas e que proporcionam uma melhor experiência de utilização para os usuários está se tornando cada vez maior. Esses requisitos estão sendo satisfeitos com a chegada das aplicações *Single-page*, que diferente das *Multi-pages*, podem proporcionar uma melhor experiência de utilização para o usuário. Este trabalho definiu um processo e ferramentas para a migração de aplicações *Multi-page* para *Single-page.* Foi realizado um estudo de caso e um grupo de foco para validar o processo e as ferramentas usadas. Com a migração do estudo de caso bem-sucedida e os *feedbacks* no grupo de foco serem positivos, foi concluído que o processo e ferramentas para migração propostos nesse trabalho, aplicados de forma correta podem realizar uma migração bem-sucedida. Concluímos também que a técnica *Single-page* não beneficia todos os tipos de aplicações. Portanto, uma análise prévia é sempre um requisito antes de migrar qualquer aplicação Web para uma arquitetura *Single-page.*

Palavras chave: Software - desenvolvimento. Aplicações Web. Aplicação *Single-page*. Aplicação. *Multi-page*. *Front-end* e *Back-end*.

.

ABSTRACT

The demand for faster and interactive Web applications that provide a better experience for the users is increasing with the growth of users connected via the Internet. These requirements are satisfied with the arrival of the single-page applications. They are different from multiple page applications and can provide a better experience for the user. This work defines a process and tools for migrating from Multi-page to Single-page applications. We conducted a case study and a focus group to validate the process and tools used. The successful case study migration and the positive feedbacks from the focus group indicate that the process and tools for migration proposed in this work applied correctly can perform a successful migration. We also conclude that the single-page technique do not benefit all kind of applications. Therefore a prior analysis is always a requirement before migrating any Web application to a Single-page client architecture.

Keywords: Migration application. Web Site. Single-page Application. Multipage Application. Front-end and back-end.

LISTA DE ILUSTRAÇÕES

Figura 1 – Representação *front-end* e *back-end* 13
Figura 2 – Representação da arquitetura MVC no lado Servidor 15
Figura 3 – Representação da arquitetura MVC hibrida 15
Figura 4 – Representação da arquitetura MVC no lado Cliente 16
Figura 5 – Processo de Migração 22
Figura 6 – *Wireframe* da aplicação SisEventos 24
Figura 7 – *Mockup* da visão de Inscrições da aplicação SisEventos 25
Figura 8 – Representação visual da visão base da aplicação SisEventos 27
Figura 9 – Aplicação SiNutri em MPA 33
Figura 10 – *Mockup* visão base da aplicação SiNutri em SPA 34
Figura 11 – *Mockup* visão histórico de consultas 35
Figura 12 – Organização de pacotes *front-end* da aplicação SiNutri SPA 38
Gráfico 1 – Comparação de interesses das tecnologias 30

LISTA DE QUADROS

Quadro 1 – Comparação entre os trabalhos relacionados e o proposto 21
Quadro 2 – Comparação entre Wireframe e Mockup 26
Quadro 3 – Métodos da página de Histórico de consultas em MPA 33
Quadro 4 – Tecnologias utilizada para implementação do *front-end* 37

SUMÁRIO

1 INTRODUÇÃO 11

2 FUNDAMENTAÇÃO TEÓRICA 13

2.1 Front-end 13

2.2 Arquitetura MVC - Modelo, Visão e Controlador 14

2.2.1 MVC no lado Servidor 14

2.2.2 MVC híbrida: lado Cliente e lado Servidor 15

2.2.3 MVC no lado Cliente 15

2.3 Multi-page Application (MPA) 16

2.4 Single-page Application 17

3 TRABALHOS RELACIONADOS 19

3.1 Migração de uma aplicação Web multi-page para Interfaces Ajax single-page 19

3.2 Práticas de projetos de interfaces de usuário em aplicações Web single-page 19

3.3 Uma abordagem automática para reengenharia de um site comum com AJAX 20

4 PROCESSO DE MIGRAÇÃO 22

4.1 Conhecer Aplicação 22

4.2 Projetar SPA 23

4.3 Criar um back-end orientado a serviços 27

4.4 Selecionar tecnologias front-end 28

4.4.1 Tecnologias SPA 28

4.4.2 Tecnologias para estilizar as visões 30

4.5 Implementar front-end 31

5 ESTUDO DE CASO 32

5.1 Análise do SiNutri 32

5.2 SiNutri em SPA 34

5.3 Back-end orientado a serviços 36
5.4 Implementação do front-end 36
5.5 Avaliar processo de migração 40
6 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS 41
REFERÊNCIAS 42

1 INTRODUÇÃO

Com o grande aumento de usuários conectados via Internet, utilizando diversos meios de acesso como celulares, computadores e outros tipos de dispositivos, que diferem no tamanho, processamento e outras características, tornou-se necessária a criação de tecnologias para melhorar a interatividade e velocidade nas respostas de aplicações Web para usuários.

A maior interatividade, rapidez nas respostas e satisfação do usuário são benefícios que devem ser alcançados não só por aplicações recém-criadas, mas também por aplicações antigas que podem ter sua usabilidade, velocidade e interatividade melhoradas. Atualmente existem diversas técnicas que podem ser utilizadas por aplicações Web no *front-end* para prover maior interatividade e rapidez. Neste trabalho serão abordadas duas técnicas, a primeira se trata da técnica de múltiplas páginas (*Multi-page application* - MPA), na qual toda aplicação tem sua estrutura baseada em diversas páginas. Já a segunda se trata da técnica de única página (*Single-page application* - SPA), em que toda aplicação tem sua estrutura baseada em somente uma única página, cujos conteúdos são dinamicamente atualizados em sua estrutura.

Para que aplicações Web proporcionem esses benefícios, deve ser realizada a migração da aplicação para o uso de tecnologias que possam ajudar a proporcionar esses benefícios. Com isso, a migração de aplicações está se tornando cada vez mais importante para aplicações Web (MESBAH; VAN, 2007). Partindo dessas necessidades, este trabalho tem como objetivo principal definir um processo e ferramentas para realizar a migração de uma aplicação MPA para SPA. O processo e as ferramentas serão validados com a realização da migração de um sistema real e com a formação de um grupo de foco para identificação dos pontos fortes e deficiências.

A aplicação Web escolhida para passar pelo processo de migração é o Sistema de Atendimento Nutricional - SiNutri[1], a mesma foi desenvolvida pelos membros do Núcleo de Práticas em Informática - NPI da Universidade Federal do Ceará - UFC no Campus Quixadá (ALMENDRA et al., 2015; GONÇALVES et al., 2013). O NPI-UFC tem como objetivo disponibilizar aos alunos um ambiente de uma fábrica de software, onde se têm processos de desenvolvimento de software bem definido e onde são executados projetos com clientes reais (GONÇALVES et al., 2013). Um desses projetos desenvolvidos foi o

[1] https://github.com/npi-ufc-qxb/sinutri

SiNutri: sistema desenvolvido para facilitar o atendimento a serviços de nutrição a comunidade acadêmica da UFC. No capítulo 5 o SiNutri será abordado com mais detalhes.

No decorrer do trabalho também foi criada uma aplicação SPA básica, chamada SisEventos, para facilitar o entendimento do autor sobre a técnica SPA e as tecnologias utilizadas para sua implementação. Além disso, essa aplicação também irá contribuir de forma didática para facilitar o aprendizado de outras pessoas sobre desenvolvimento de aplicações SPA. Inclusive, de acordo com Almendra et al. (2015), o NPI-UFC fornece treinamentos com tecnologias a serem utilizadas no decorrer do estágio. A aplicação SisEventos pode ser utilizada no treinamentos para auxiliar os alunos no entendimento da técnica SPA.

O restante deste trabalho está organizado da seguinte forma: no capítulo 2, é apresentada a fundamentação teórica, onde são abordados os principais conceitos aqui utilizados; no capítulo 3 são apresentados alguns trabalhos que possuem relação com o tema que está sendo abordado; no capítulo 4 é apresentado o processo de migração criado; no capítulo 5 é apresentado o estudo de caso que aplica o processo de migração criado neste trabalho; por fim, na Seção 6 temos as considerações finais e trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

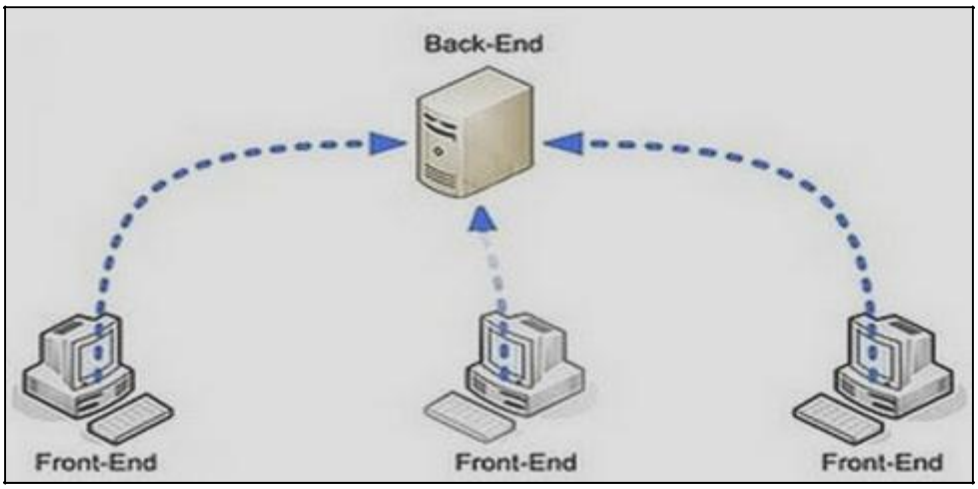
A seguir serão descritos alguns conceitos chaves deste trabalho. São eles: *front-end*, arquitetura MVC e as técnicas, *Multi-Page Application* (MPA) e *Single-Page Application* (SPA). Esses conceitos possuem um forte relacionamento e são de grande importância para o entendimento dos objetivos deste trabalho.

2.1 Front-end

Front-end refere-se a toda camada visual e de interação com o usuário de uma aplicação. "O *front-end* está evoluindo, longe estão os dias de codificação de arquivos HTML estáticos e CSS reservados." (FENDER; YOUNG, 2015, p.1).

Facilidade de uso, rapidez e uma interface atraente são três características de grande importância de uma aplicação Web que o usuário espera ao acessá-la. De acordo com Souders (2013), uma considerável parte do tempo gasto pelos usuários esperando o carregamento das páginas é com o *front-end*.

Figura 1 - Representação front-end e back-end



Fonte: basesdatos2011.wikispaces.com

No decorrer da utilização de uma aplicação Web, o usuário interage com diversos tipos de dados através do *front-end*. Com isso, surge a seguinte pergunta: quem fornece esses dados que são exibidos para o usuário? Na figura 1, encontramos um elemento chamado *Back-End*, que tem como objetivo responder as requisições realizadas pela aplicação *front-end*. A requisição depende da necessidade da aplicação, que também depende do que o usuário está querendo realizar. Um exemplo de requisição seria uma consulta na aplicação em que o usuário deseja visualizar todas as suas compras

realizadas. O *front-end*, então, envia essa requisição ao *back-end* que se encarrega de realizar uma consulta em um banco de dados e retornar as informações para o usuário.

A camada *front-end* tem um papel muito importante nas aplicações Web, pois todo contato do usuário final com a aplicação *back-end* ocorre através dessa camada. Portanto, as técnicas utilizadas em aplicações Web, como MPA e SPA, geram resultados que podem ser visualizados pelos usuários finais através da camada *front-end*.

2.2 Arquitetura MVC - Modelo, Visão e Controlador

Modelo, visão e controlador (MVC) trata-se de uma arquitetura de software criada com o objetivo de estruturar uma aplicação em camadas para facilitar o desenvolvimento e melhorar a manutenção de código.

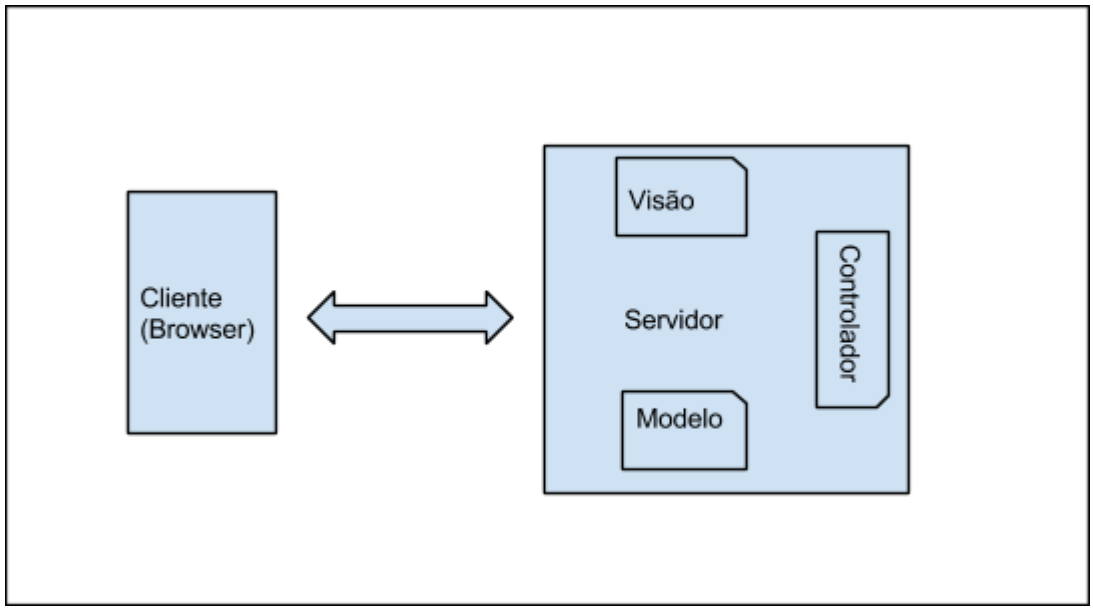
De acordo com Burbeck (1992) na arquitetura MVC a entrada do usuário, modelagem do mundo externo e o visual apresentado para o usuário são separados e manipulados de maneiras diferentes. A camada modelo é a responsável por representar os dados da aplicação. Visão é a camada que interage com o usuário. Todas as telas que o usuário interage na aplicação estão nessa camada. A terceira camada chamada controlador fica responsável por gerenciar a comunicação entre o modelo e a visão.

De acordo com Morales-Chaparro (2007), a arquitetura MVC pode ser utilizada de três maneiras diferentes em aplicações Web, são elas: MVC no lado do Servidor, MVC híbrida: lado cliente e lado servidor e MVC no lado do cliente. A seguir serão apresentadas as características de cada abordagem.

2.2.1 MVC no lado Servidor

Nessa abordagem as camadas modelo, visão e controlador são executadas no lado do servidor (*back-end*) como ilustradas na Figura 2. Toda geração de HTML, redirecionamento de páginas, validação de dados e decisões sobre regras de negócios são realizadas no servidor e enviadas para o cliente, que irá receber e visualizar através do navegador (*browser*). Toda requisição HTTP ou HTTPS ao servidor faz com que a página inteira seja recarregada novamente, mesmo que somente uma pequena porção dela tenha sido modificada (MORALES-CHAPARRO, 2007).

Figura 2 - Representação da arquitetura MVC no lado Servidor

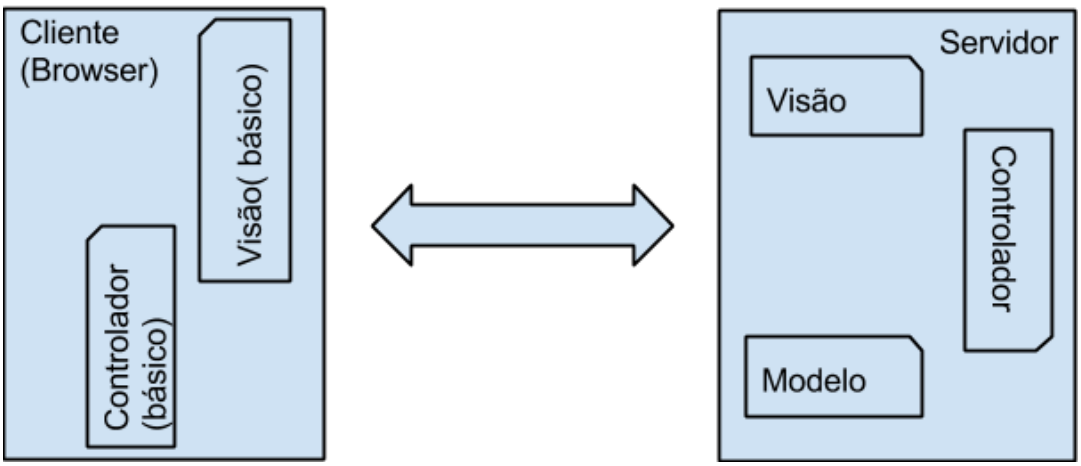


Fonte: Elaborada pelo autor.

2.2.2 MVC híbrida: lado Cliente e lado Servidor

Na abordagem híbrida as camadas modelo, visão e controlador continuam sendo executados em sua maioria no lado do servidor como ilustrado na Figura 3. A diferença entre essa abordagem e a MVC *Server-side* é que algumas funcionalidades das camadas visão e controlador como: validações de campos e respostas as iterações do usuário na interface são executadas no lado do cliente, descartando a necessidade de requisitar essas tarefas ao servidor (MORALES-CHAPARRO, 2007).

Figura 3 - Representação da arquitetura MVC hibrida



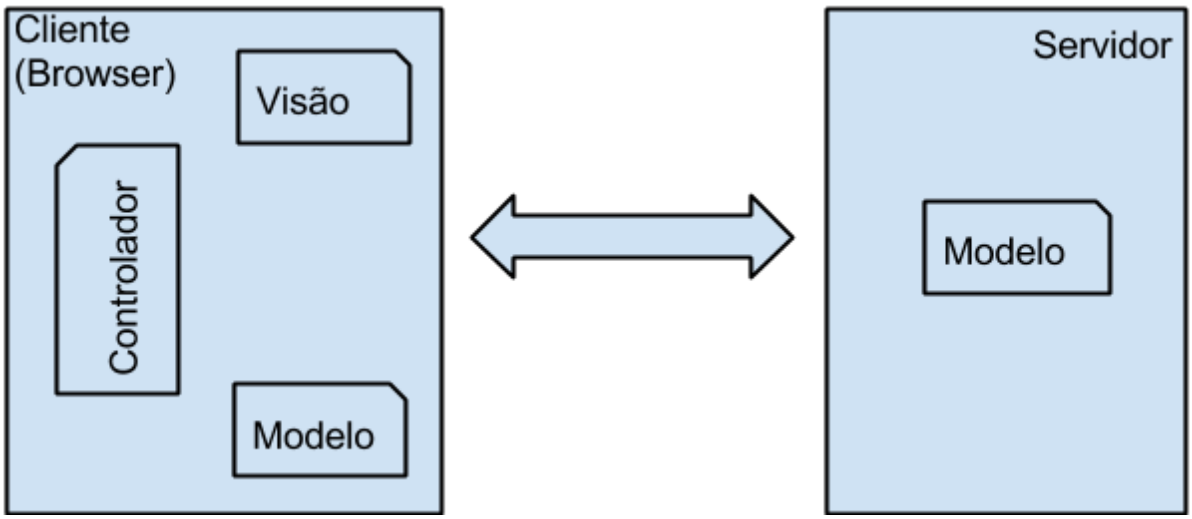
Fonte: Elaborada pelo autor

2.2.3 MVC no lado Cliente

Nessa abordagem as camadas visão, controlador e as lógicas de negócio do modelo são executadas no lado do cliente como ilustrado na Figura 4. Com isso, toda

geração de HTML, redirecionamento de páginas, validação de dados e decisões sobre regras de negócio são realizados no lado do cliente (MORALES-CHAPARRO, 2007).

Figura 4 - Representação da arquitetura MVC no lado Cliente



Fonte: Elaborada pelo autor

2.3 Multi-page Application (MPA)

MPA trata-se da técnica utilizada pelas aplicações Web tradicionais, que possuem diversas páginas em toda aplicação e frequentemente utilizam a arquitetura MVC no lado Servidor.

Segundo Mesbah e Van (2007), aplicações Web que utilizam a técnica de MPA têm a página inteira recarregada conforme iterações do usuário, por mais que apenas um componente da página necessite ser atualizado. Esse recarregamento da página inteira é devido as camadas responsáveis por gerar a resposta para o usuário se encontrarem no servidor e retornarem como resposta a página a ser carregada novamente para visualização do resultado. Considere o exemplo da seguinte aplicação MPA: O usuário entra na aplicação Web que fornece o serviço de pesquisa por endereço com base no CEP. A página de busca por endereço possui três componentes principais: um campo para digitar o CEP, um botão para confirmar a busca e um campo de resultado que é preenchido com o endereço encontrado. Ao entrar nessa página, o campo de digitar o CEP se encontra vazio e o campo de resultado também. Quando o usuário digita o CEP e clica no botão buscar, é enviada uma requisição a aplicação no servidor. Essa requisição é processada e ao final retorna uma nova página de busca por endereço para ser carregada, contendo o campo de resultado preenchido com as informações do endereço encontrado. Nesse exemplo uma página inteira teve que ser recarregada para poder atualizar somente um componente da página, característica comum em aplicações MPA.

Essa característica acaba não promovendo uma boa experiência para o usuário, além disso, o uso da técnica pode resultar em aplicações que realizam mais requisições ao servidor, aumentando assim o tráfego de dados na rede. Logo, essa técnica não é apropriada para aplicações que necessitam de maior interatividade com o usuário, atualização dinâmica de conteúdo, maior rapidez e fluidez na navegação da aplicação.

A técnica MPA tem suas vantagens e desvantagens dependendo do tipo da aplicação na qual ela está sendo utilizada. Algumas aplicações podem se beneficiar mais com a utilização da técnica MPA do que outras. Por exemplo: Uma aplicação Web possui em sua estrutura algumas páginas que geralmente podem ser acessadas através de um *menu*. Essas páginas contêm apenas textos e imagens que são atualizados com pouca frequência. Pelo perfil dessa aplicação, ela não precisa utilizar técnicas como a SPA para aumentar a interatividade, rapidez nas respostas e atualização dinâmica dos conteúdos, já que os mesmos em sua maioria são estáticos, portanto, essa aplicação se beneficia mais com a utilização da técnica MPA do que outras técnicas que irão promover recursos que ela não necessita.

2.4 Single-page Application

SPA trata-se de uma técnica utilizada em aplicações Web em que toda a aplicação é baseada em uma única página com o objetivo de promover maior interatividade para o usuário. Segundo Mikowski e Powell (2013), uma aplicação SPA consegue promover uma prestação de serviço mais rápida e uma aplicação mais interativa para os usuários. Esses benefícios são alcançados através das arquiteturas que uma aplicação SPA pode adotar. Uma frequentemente utilizada é a arquitetura MVC no lado Cliente. Com essa arquitetura, a aplicação SPA pode responder a diversas ações dos usuários sem o recarregamento da página inteira.

Considerando o exemplo do tópico 2.3, em que se tratava de uma aplicação MPA, que fornece um serviço de pesquisa por endereço com base no CEP. Nesse exemplo a página inteira foi recarregada para mostrar apenas o resultado do endereço encontrado, não promovendo uma boa experiência para o usuário e aumentando o tráfego de dados entre cliente e servidor. Ao contrário, em uma aplicação SPA, o servidor retornaria apenas os dados do endereço em um formato JSON ou XML e somente o componente de resultado seria atualizado com o endereço retornado, reduzindo assim o tráfego de dados entre cliente e servidor e enriquecendo a experiência do usuário.

A técnica SPA possui diversas outras vantagens em comparação com a MPA, algumas delas são:

- Atualização de fragmentos de HTML da aplicação quando necessário, descartando a necessidade da atualização de toda a página no momento que é requisitado algo do servidor.
- Aplicação no servidor realiza menos tarefas, reduzindo assim o consumo de recursos computacionais do servidor.
- Redução da quantidade de dados na comunicação entre cliente e servidor. Os códigos HTML, CSS e JavaScript são enviados a primeira vez para o cliente, após isso, as comunicações são em sua maioria de dados brutos.

A utilização da técnica SPA proporciona uma aplicação mais fluida, rápida, dinâmica e de fácil utilização, melhorando assim a experiência do usuário com a aplicação. “A técnica SPA é uma forma de construir soluções de página única em toda aplicação, com base em técnicas dinâmicas e assíncronas.” (TESARIK et al., 2013). A técnica SPA também possui algumas desvantagens, são elas:

- O primeiro carregamento de uma aplicação SPA pode acabar demorando mais do que uma aplicação MPA, pois é feito o carregamento dos arquivos para funcionamento da aplicação, que podem ser muitos por se tratar de uma única página.
- Aplicações SPA trabalham com tecnologias não utilizadas em aplicações Web tradicionais, tornando necessário o aprendizado de novas linguagens e *frameworks* para o seu desenvolvimento.
- Aplicações SPA utilizam tecnologias modernas da Web. Caso o navegador esteja com o JavaScript desatualizado ou desabilitado, poderá impossibilitar o uso da aplicação.

3 TRABALHOS RELACIONADOS

Esta seção tem como objetivo apresentar e comparar alguns trabalhos que possuem relação com o tema que está sendo abordado neste trabalho proposto. O Quadro 1 compara algumas características dos trabalhos relacionados e o proposto.

3.1 Migração de uma aplicação Web multi-page para Interfaces Ajax single-page

Mesmo com a enorme popularidade da Web, é possível encontrar diversas aplicações que não possuem boa usabilidade e interatividade com o usuário. Com isso, de acordo com Mesbah e Van (2007), muitas organizações estão começando a considerar a migração de suas aplicações para esse novo paradigma chamado SPA, o qual promete melhor satisfação dos seus clientes em relação à sua experiência de uso com menores tempos de resposta e com uma interface mais rica, interativa e responsiva.

No trabalho de Mesbah e Van (2007) é proposto um novo processo para realização da migração de aplicações MPA para SPA utilizando Ajax. Para realizar essa migração, existe a necessidade de identificar os componentes e páginas candidatas a SPA. Essa necessidade é a principal questão abordada por Mesbah e Van (2007), tendo como pré-requisito, o conhecimento de toda a estrutura de navegação da aplicação.

No decorrer do trabalho é também apresentada uma técnica de engenharia reversa com o objetivo de realizar agrupamento de páginas com estruturas semelhantes. Além da técnica, também foi criada uma ferramenta chamada RETJAX, na qual é realizado um processo para a extração de percursos de navegação e a identificação de páginas e componentes candidatos para nova estrutura em SPA.

O trabalho aqui proposto tem como objetivo principal a definição de um processo e ferramentas para realizar a migração de uma aplicação MPA para SPA, diferente do trabalho de Mesbah e Van (2007), que tem como foco principal criar um processo e uma ferramenta para identificar componentes e páginas candidatas a SPA.

3.2 Práticas de projetos de interfaces de usuário em aplicações Web single-page

Como a técnica SPA centraliza os recursos da aplicação em uma única página, acaba afetando o processo de concepção do desenvolvimento da aplicação. A migração de uma aplicação MPA para SPA exige grande atenção em vários aspectos no decorrer do processo, desde o entendimento da estrutura da aplicação, antes de ser migrada, como também em como essa aplicação se comportará com a utilização da nova técnica para atender os mesmos recursos fornecidos anteriormente. Para alcançar esses resultados, Tesarik et al.

(2008) propõem soluções utilizando padrões em nível de projeto de página para uma implementação bem sucedida da técnica SPA.

O trabalho apresentado por Tesarik et al. (2008) e o trabalho aqui proposto enfatizam a importância que se deve dar ao entendimento da estrutura da aplicação MPA antes de ser migrada para uma SPA. No entanto, os trabalhos diferem nos níveis de solução e objetivos a serem alcançados. Enquanto Tesarik et al. (2008) busca propor soluções a nível de projeto para uma boa implementação da SPA, o trabalho aqui proposto busca definir um processo e ferramentas para realizar a migração de uma aplicação MPA para SPA de maneira geral, não só a nível de projeto.

3.3 Uma abordagem automática para reengenharia de um site comum com AJAX

De acordo com Wang et al. (2008), a pouca interatividade, usabilidade e o maior tempo de respostas das aplicações Web para o usuário são os principais motivos para a redução da satisfação e visitas dos usuários ao site.

Para resolução desses problemas, Wang et al. (2008) propõem a utilização da tecnologia AJAX (Asynchronous Javascript e XML), com o principal objetivo de tornar as páginas Web mais rápidas e interativas atualizando determinadas partes da página em resposta às ações do usuário, descartando a necessidade do recarregamento da página inteira, promovendo assim melhor satisfação do usuário. Para melhor aproveitamento dos benefícios dessa técnica, é necessária a reestruturação da aplicação.

O trabalho apresentado por Wang et al. (2008) têm como objetivo principal a reestruturação de uma aplicação Web clássica para uma utilizando AJAX. Para realização desse processo foi criada uma ferramenta chamada ACT, que de acordo com Wang et al. (2008), executa o processo de forma rápida e eficiente, reduzindo assim a carga de trabalho do desenvolvimento.

O trabalho aqui proposto e o trabalho apresentado por Wang et al. (2008) possuem semelhanças em seus objetivos de promover maior interatividade para o usuário, enquanto diferem nos métodos e ferramentas utilizados para a reestruturação das aplicações clássicas para interativas. O trabalho proposto busca definir o passo a passo para migração com a utilização de ferramentas que simplifiquem e que tenha como resultado uma aplicação SPA bem estruturada, enquanto o apresentado por Wang et al. (2008) busca migrar uma aplicação MPA de uma forma rápida, enquanto.

Quadro 1 - Comparação entre os trabalhos relacionados e o proposto

	Técnica utilizada	**Realiza Migração**	**Construção de ferramenta para auxiliar na Migração**
(MESBAH; VAN, 2007)	SPA	Sim	*RETJAX*
(TESARIK et al., 2008)	SPA	Não	Não
(WANG et al, 2008)	*Ajaxification*	Sim	ACT
Trabalho Proposto	SPA	Sim	Não

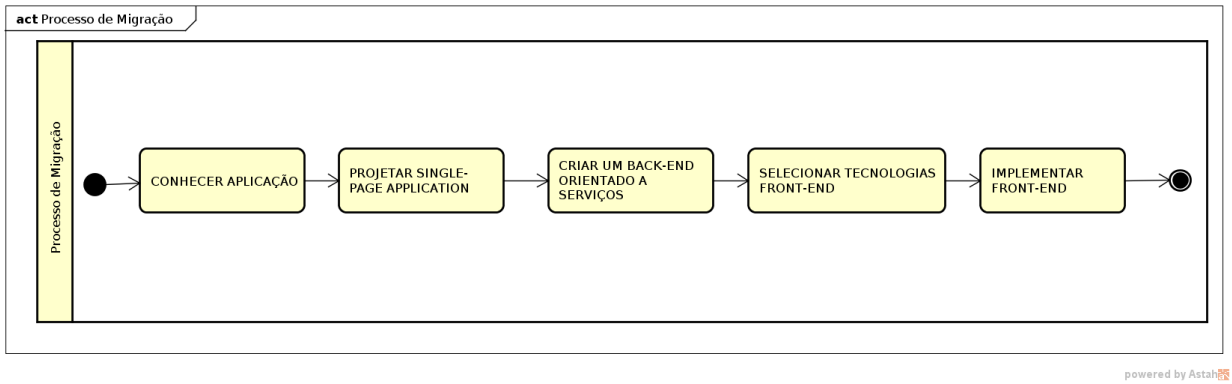
Fonte: Elaborada pelo autor.

Cada trabalho relacionado apresentado nesse capítulo serviu como fonte de conhecimento para a criação do processo detalhado no próximo capítulo.

4 PROCESSO DE MIGRAÇÃO

O objetivo principal do trabalho aqui proposto é a criação de um processo de migração de uma aplicação MPA para SPA. O processo é composto por cinco passos bem definidos para que a migração seja realizada de forma bem-sucedida. A imagem a seguir apresenta os cinco passos do processo de migração que serão descritos nos tópicos a seguir.

Figura 5 - Processo de Migração



Fonte: Elaborada pelo autor

4.1 Conhecer Aplicação

Para realização da migração de uma aplicação de forma correta, é primordial que a pessoa ou equipe que está responsável pela migração tenha conhecimento sobre os requisitos e como esses foram implementados na aplicação em MPA. Para que esse ponto seja atendido e a migração possa ser realizada, o processo aqui descrito tem como primeiro passo o entendimento da aplicação.

Existem muitas técnicas para obter conhecimento sobre os requisitos e de como foram implementados, algumas delas mais complexas e outras mais simples e ágeis, exemplos:

- Análise dos documentos da Aplicação (Documento de visão, requisitos e casos de uso).
- Entrevista com desenvolvedores e usuários da aplicação.
- Utilização da aplicação como usuário.
- Análise dos diagramas de Classes e Sequencia.
- Análise do código desenvolvido.

A análise dos documentos da aplicação pode promover um entendimento geral sobre a mesma, mostrando para quais problemas a aplicação foi desenvolvida para solucionar, como também, quais são os seus requisitos em detalhes e os seus respectivos casos de uso. Caso a aplicação não possua documentação, pode-se utilizar de entrevistas

com os desenvolvedores e usuários da aplicação. Essas pessoas podem passar um conhecimento real da utilização no dia-a-dia. A utilização da aplicação como usuário é outra técnica listada acima de grande eficácia, o uso da aplicação pode facilitar o entendimento das funcionalidades, problemas não resolvidos e pontos a serem melhorados.

O diagrama de classes é uma representação de todo conjunto de classes e suas relações dentro de um Software. Através desse diagrama é possível conhecer todos os atributos, métodos e relações de cada classe. O diagrama de sequencia representa a comunicação interna de iterações da aplicação. Através desse diagrama é possível entender a comunicação entre o *front-end* e *back-end* da aplicação em MPA, no qual é bastante importante para poder identificar camadas de código que podem ser reaproveitadas na aplicação em SPA. Na falta dos diagramas de sequencia ou como complemento, pode também ser criado um mapeamento de todo *back-end*. Uma forma de criar o mapeamento é adicionando um *logger* a cada método do *back-end*, para que ao interagir com o *front-end* da aplicação, sejam visualizados quais métodos foram requisitados. A análise do código implementado pode esclarecer como cada método foi implementado e onde são chamados para execução. Ajuda também na identificação de erros de implementação e melhorias necessárias.

4.2 Projetar SPA

Com o entendimento dos requisitos, funcionalidades e como foram desenvolvidos, pode-se então projetar a aplicação em SPA. A execução desta etapa tem como resultado a representação visual de cada visão da aplicação em SPA, que será utilizada como base na etapa de implementação. As visões projetadas nas representações visuais devem atender os mesmos recursos disponibilizados na aplicação em MPA. Existem diversas formas e ferramentas para criação de representações visuais de softwares. Entre essas formas, a *Wireframe* e *Mockup* se destacam considerando o contexto de migração, pois proporcionam ao processo agilidade e facilidade, como também há uma grande quantidade de ferramentas acessíveis na *web.*

Wireframe é uma representação visual básica, que tem como principal objetivo mostrar quais são os componentes principais da aplicação, sem a necessidade de entrar em detalhes dos componentes, porém, deixando claro o objetivo de cada um. Uma vantagem na utilização do *Wireframe* é a rapidez na sua criação, porém, como desvantagem, tem o não detalhamento do que está sendo projetado, passando essa responsabilidade para outra etapa.

Podem ser encontradas na web diversas ferramentas que facilitam a criação de *Wireframe*, algumas delas são:

- *Pidoco*[2]
- *Wireframe* CC[3]
- *Visual Paradigm*[4]

A Figura 6 é um exemplo de *Wireframe*, criada por meio da ferramenta *Wireframe* CC. O exemplo se trata da aplicação SisEventos citada na introdução desse trabalho. Essa aplicação tem como principais requisitos o controle de Instituições, Participantes, Atividades e Inscrições nas atividades de um evento. No *Wireframe* da Figura 6, foi projetada a visão de Inscrições dos participantes em atividades, na qual ao ser acessada, tem como requisito mostrar os participantes cadastrados, quais atividades cada participante está inscrito e as informações de cada atividade. Observa-se que a visão foi estrutura de maneira simples, porém, definindo seus principais componentes e a finalidade de cada um.

Figura 6 - *Wireframe* da aplicação SisEventos

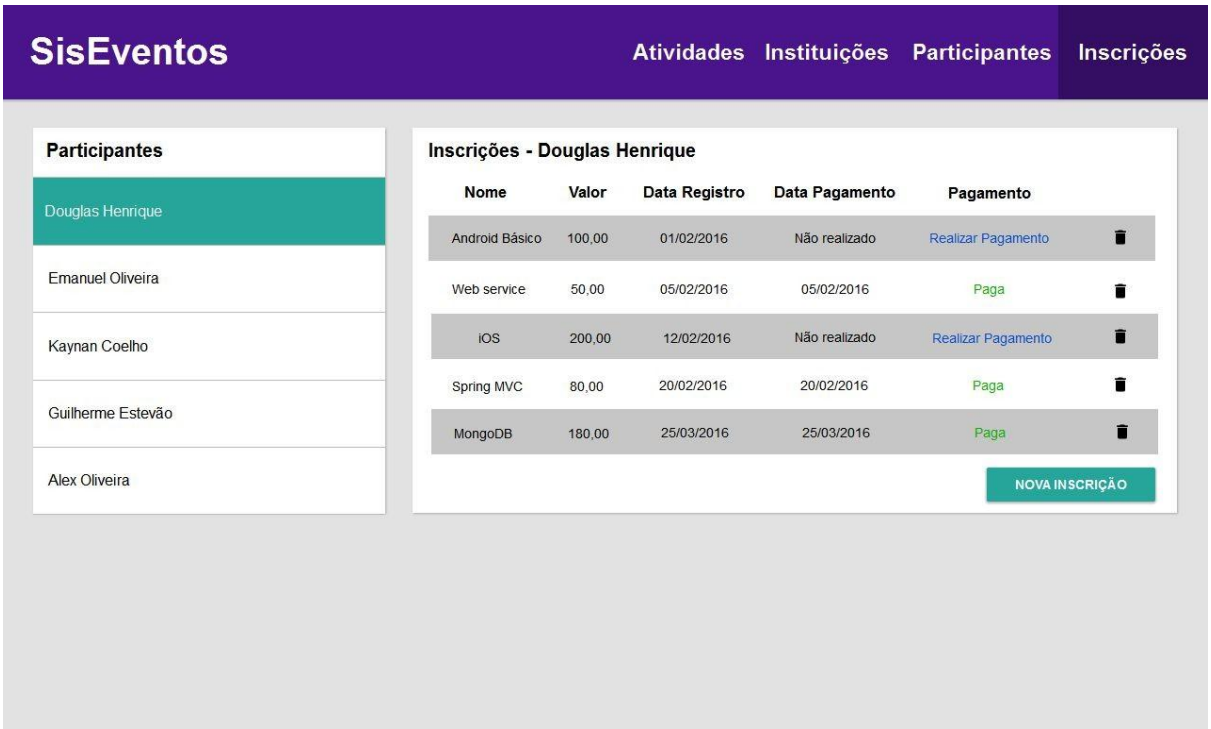
Fonte: Elaborada pelo autor

[2] https://pidoco.com
[3] https://wireframe.cc/
[4] https://www.visual-paradigm.com/

O *Mockup*, diferente do *Wireframe*, é uma representação visual trabalhada de forma mais detalhada, definindo os componentes da visão com informações do contexto real da aplicação. Também são definidos atributos visuais como cores, tamanhos, ícones, entre outros. O resultado final com o *Mockup* é uma representação visual de média a alta fidelidade, ou seja, bem próximo do que será entregue ao final da implementação. Existem também diversas ferramentas que facilitam a criação de *Mockups*, algumas são listadas a seguir:

- *Proto.io*[5]
- *NinjaMock*[6]
- *UXpin*[7]

Figura 7 - Mockup da página de Inscrições da aplicação SisEventos



Fonte: Elaborada pelo autor

A Figura 7 representa a aplicação SisEventos em forma de *Mockup*, criada por meio da ferramenta *Proto.io*. Comparado ao *Wireframe*, pode ser notado grande diferença. O *mockup* foi projetado com informações do contexto real da aplicação, listando participantes, atividades de um participante selecionado e os dados de cada atividade. Atributos visuais como cor, item selecionado, linhas da tabela, diferença visual entre atividade que já foi

[5] https://proto.io
[6] https://ninjamock.com
[7] https://www.uxpin.com/

realizado o pagamento e ícones também foram definidos. Como dito, o *Mockup* é bem próximo do que será entregue ao final do desenvolvimento, tornando assim um meio mais eficiente para validação pelos gerentes do projeto do que um *Wireframe*, como também, facilita o entendimento do que deve ser implementado pelos desenvolvedores, reduzindo as reimplementações. O Quadro 2 mostra uma comparação entre as duas formas apresentadas.

Quadro 2 - Comparação entre *Wireframe* e *Mockup*

Formas	**Fidelidade**	**Características**	**Tempo**	**Ferramentas**
Wireframe	Baixa.	Principais componentes, sem detalhes visuais.	Curto tempo de desenvolvimento.	Pidoco, Wireframe CC e Visual Paradigm.
Mockup	Média a Alta.	Componentes principais e secundários, detalhamento dos componentes.	Curto a médio tempo de desenvolvimento.	*Proto.io*, *NinjaMock* e *UXpin.*

Fonte: Elaborada pelo autor.

Acompanhando a abordagem do tópico 2.4, *Single-page Application*, a técnica SPA tem diversas vantagens em comparação com a MPA, uma delas é a atualização de componentes específicos, descartando a necessidade de recarregamento de toda a página. Esse ponto é de grande importância nessa etapa. Após selecionar a forma de criar as representações visuais das visões, a primeira visão a ser criada deve ser base da aplicação SPA. A visão base da aplicação geralmente possui componentes que serão mantidos em todas as visões e um componente principal que será atualizado com uma visão de acordo com a navegação do usuário, regras de negócio e outros critérios da aplicação. A Figura 8 exemplifica a representação visual da visão base da aplicação SisEventos, que tem o seu componente principal atualizado com as outras visões conforme o usuário navega pelo menu superior.

Figura 8 - Representação visual da visão base da aplicação SisEventos

Fonte: Elaborada pelo autor

Após criar a representação visual da visão base da aplicação SPA, pode ser criado as representações visuais das outras visões da aplicação.

4.3 Criar um back-end orientado a serviços

Como descrito no tópico 4.1.2, Comunicação interna da aplicação, uma aplicação que utiliza a técnica MPA, tem seu *front-end* muito acoplado ao *back-end*, enquanto uma aplicação que utiliza a técnica SPA, tem um baixo acoplamento entre essas duas camadas e uma comunicação bem estruturada. Essa etapa tem como objetivo realizar a migração do *back-end* em MPA para um *back-end* orientado a serviços. Esse novo *back-end*, deve atender ao que foi projetado na etapa anterior. Usando como exemplo o *Mockup* da página de Inscrições da aplicação, para que possa ser atendido aos seus requisitos, deve ser criado um serviço no *back-end* que ao ser requisitado pelo *front-end* retorne os participantes e as atividades que estão inscritos. O *front-end* então recebe a resposta do *back-end* e se encarrega de colocar as informações visíveis para o usuário.

Para realizar essa migração, pode ser utilizado como base o processo definido em Sá (2016), o qual descreve um processo para realizar a migração de uma Arquitetura monolítica para uma arquitetura orientada a serviços. Também pode ser utilizado como base o método para desenvolvimento de Software baseado em microsserviços definido em Rosa (2016). Ambos os trabalhos buscam orientar a criação de um *back-end* orientado a serviços para comunicação com o *front-end*.

4.4 Selecionar tecnologias *front-end*

Nessa etapa serão apresentadas as principais tecnologias *front-end* para implementação da técnica SPA. De forma complementar, também serão apresentadas tecnologias que simplificam e aceleram a implementação do estilo das páginas da aplicação. O processo aqui proposto não busca impor tecnologias a serem utilizadas, mas sim, apresentar as principais opções. Dessa maneira, a pessoa ou equipe que está realizando a migração, pode escolher as tecnologias que mais se adequam ao contexto da aplicação.

4.4.1 Tecnologias SPA

Este tópico apresenta tecnologias *front-end* populares para construção de aplicações *front-end* SPA. Cada tecnologia possui uma boa documentação para auxiliar na utilização.

Angular JS[8]: é um *framework* JavaScript mantido pelo Google. Como é um *framework* bem flexível, possibilita ao desenvolvedor optar se deseja utilizar a arquitetura MVC ou qualquer outra de sua preferência, com isso, a arquitetura padrão do Angular JS foi denominada MVW - Model-View-Whatever pelos seus colaboradores. Entre suas diversas funcionalidades, pode se destacar a criação de componentes Web reutilizáveis, chamados de diretivas. O Angular JS possui uma comunicação *real-time* entre as camadas de *view* e *model*, também conhecida como *data-binding*, que realiza a atualização automática da *view* quando ocorre alguma mudança na camada *model*. Para o roteamento de páginas da aplicação, o Angular JS dispõe de um módulo chamado *ngRouter*, que facilita a configuração de roteamento em um único arquivo. O Angular JS também possui como padrão os serviços *$http e $resource* para facilitar a comunicação com serviços *HTTPs* remotos.

Knockout[9]: é uma biblioteca JavaScript que tem como padrão de arquitetura a MVVM - *Model-View-View-Model*. A MVVM é uma arquitetura semelhante ao MVC, a diferença é que a camada *Controller* é substituída pela *ViewModel*, responsável pela manipulação de dados e o *data-binding* da aplicação. Para o roteamento de páginas, deve ser utilizado bibliotecas externas, pois o Knockout não possui essa funcionalidade como padrão. Caso a aplicação tenha que se comunicar com serviços remotos, o Knockout disponibiliza funções de comunicação HTTP, as quais em *background* fazem uso da biblioteca jQuery.

[8] https://angularjs.org/
[9] http://knockoutjs.com

Ember JS[10]: é um *framework* JavaScript, adotante do padrão de arquitetura MVVM - *Model-View-View-Model*. Ember JS utiliza a biblioteca externa *Handlebars* para dar suporte ao *data-binding* e a criação de componentes reutilizáveis.

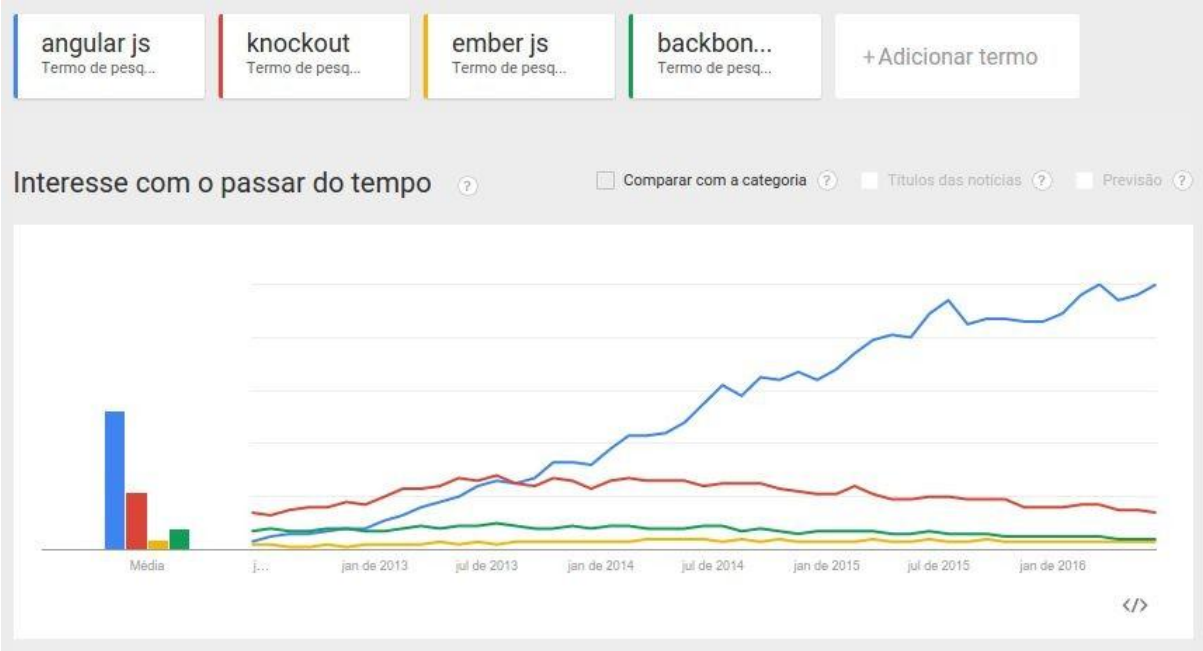
Para comunicação com serviços remotos o Ember JS faz uso da biblioteca jQuery, fornecendo a chamada das funções HTTP da biblioteca. O roteamento de página pode ser criado de forma simples e sofisticado, sem o uso de bibliotecas externas, facilitando o aprendizado.

Backbone JS[11]: é um framework JavaScript que utiliza como padrão a arquitetura MVW - *Model-View-Whatever*. O *data-binding* não vem como padrão no Backbone JS como nos frameworks Knockout e Angular JS. Para que se possa criar essa ligação de forma simples pode ser utilizada a biblioteca externa Epoxy JS, desenvolvida exclusivamente para suprir essa necessidade no Backbone JS. O roteamento de página vem como padrão, porém, é bem simples e pode deixar a desejar em casos mais complexos. Caso a aplicação tenha que se comunicar com serviços remotos, pode ser adicionado como dependência a biblioteca jQuery, podendo assim fazer o uso de suas funções de comunicação HTTP em conjunto com as funções do Backbone JS.

O Gráfico 1 apresenta os interesses dos usuários pelas tecnologias entre Junho de 2012 à Junho de 2016, o mesmo foi gerado pela ferramenta Google Trends, na qual se baseia nas buscas realizadas pelas tecnologias: Angular JS, Knockout, Ember JS e Backbone JS no Google. O *framework* Angular JS ultrapassou ao Knockout em Julho de 2013, desde então vem ganhando grande popularidade entre os desenvolvedores, enquanto o Knockout, Backbone JS e o Ember JS permanecem abaixo.

[10] http://emberjs.com/
[11] http://backbonejs.org/

Gráfico 1 - Comparação de interesse das tecnologias



Fonte: Google Trends

4.4.2 Tecnologias para estilizar as visões

Este tópico apresenta duas tecnologias *front-end* populares para estilização de aplicações web. Ambas tecnologias possuem uma boa documentação e podem acelerar a etapa de estilização.

Materialize CSS[12]: é um *framework front-end*, possui diversos componentes implementados em HTML, CSS e JavaScript com base na linguagem visual *Material Design*, criada pelo Google com o objetivo de disponibilizar princípios para criação de aplicações com um bom design e usabilidade. O uso do Materialize CSS possibilita também a criação de um design responsivo, se adequando ao tamanho da tela do dispositivo que está acessando a aplicação.

Bootstrap[13]: um *framework front-end* criado pelos desenvolvedores do Twitter com o objetivo de criar um padrão de design nas aplicações dentro da empresa. Com o suporte a criação de aplicações com design responsivo, facilidade de utilização, crescimento dos componentes disponíveis e várias outras evoluções, o Bootstrap se tornou um dos frameworks mais populares para desenvolvimento do *front-end* de aplicações Web

.

[12] https://materializecss.com
[13] https://getbootstrap.com/

4.5 Implementar *front-end*

Esta etapa é a ultima do processo aqui proposto, objetivando especificamente a implementação do *front-end* da aplicação utilizando a técnica SPA. A implementação deve ter como base os requisitos e funcionalidades identificados e a representação visual projetada, para que os recursos disponibilizados na versão da aplicação em MPA também sejam disponibilizados em SPA.

No próximo capítulo será apresentado de forma aprofundada um estudo de caso utilizando o processo de migração proposto nesse trabalho. A aplicação Web do estudo de caso teve a técnica SPA implementada com a utilização do *framework* Angular JS, para a estilização das páginas foi utilizado o *framework* Materialize CSS e para a criação de um *back-end* orientado a serviços foi utilizado o *Spring framework*. Os critérios de escolha para cada tecnologia utilizada será detalhado no decorrer do estudo de caso.

5 ESTUDO DE CASO

Este capítulo descreve a execução e resultados da migração de uma aplicação Web MPA para uma SPA utilizando o processo detalhado na seção anterior. A aplicação migrada se chama SiNutri e foi desenvolvida no NPI-UFC para facilitar o atendimento a serviços de nutrição a comunidade acadêmica da UFC Campus de Quixadá. A migração para a técnica SPA foi realizada nos módulos de *Login*, Busca por paciente e Histórico do paciente. A descrição da migração a seguir foi focada no módulo de Histórico do paciente, pois entre os três, é o mais complexo e que exigiu maior esforço na migração.

5.1 Análise do SiNutri

Após a realização da migração, a aplicação deve disponibilizar os mesmos recursos em nível de funcionalidades aos usuários. Para isso, foi realizado um estudo dos requisitos e funcionalidades da aplicação em MPA, para que fossem entendidos e desenvolvidos na aplicação em SPA.

De acordo com o documento de visão do SiNutri, o módulo de Histórico do paciente está entre os principais da aplicação, sendo definido como de alta prioridade e com o objetivo de disponibilizar para o usuário a visualização das últimas consultas realizadas pelo paciente. Com a análise dos documentos e a utilização da aplicação como nutricionista, foram identificados os seguintes requisitos:

1. Visualização das últimas consultas do paciente.
2. Visualização das informações pessoais do paciente.
3. Gráficos de histórico do peso,
4. Gráficos de histórico do Índice de massa corporal (IMC).
5. Gráficos de histórico da Circunferência da cintura (CC).
6. Funcionalidades relacionadas a consultas: Visualizar detalhes, editar consulta, gerar relatório da consulta.
7. A página deve conter uma opção para iniciar nova consulta com o paciente.

A Figura 9 apresenta a página de histórico de consultas da aplicação SiNutri em MPA. Ela atende a todos os requisitos listados acima e foi desenvolvida com as tecnologias JSP no *front-end* e Java no *back-end*. Também foi utilizada a arquitetura MVC para a estruturação da aplicação.

Figura 9 - Aplicação SiNutri em MPA

Fonte: Elaborada pelo autor

A aplicação SiNutri não possui nenhum diagrama de sequência para o entendimento da comunicação entre *front-end* e *back-end* da página de histórico, que facilitaria a identificação de camadas de código do *back-end* da página a serem reutilizadas, para contornar essa ausência, foi criado um mapeamento de todos os métodos dos controladores da aplicação. Com esse mapeamento, foi possível identificar quais controladores, métodos, consultas ao banco e modelos de dados utilizados para atender aos requisitos da página. A partir do resultado do mapeamento, foi criado o Quadro 3 que apresenta os métodos utilizados para o carregamento da página de histórico de consultas. Esses métodos foram utilizados como base para a criação dos serviços para atendar as requisições da visão histórico de consultas detalhada no próximo tópico.

Quadro 3 - Métodos da página de Histórico de consultas em MPA

Nome	***Controller***	**Descrição**
getPaginaHistorico	PacienteController	Redireciona para a página de histórico e passa um objeto com as informações do paciente
getPesoByConsulta	NutricaoController	Carrega as consultas nutricionais do paciente

Fonte: Elaborada pelo autor

5.2 SiNutri em SPA

As representações visuais das visões do SiNutri foram criadas como *Mockups*. Essa escolha se deu principalmente por já existir uma versão funcional do SiNutri, na qual se pode ter uma noção básica da estrutura que as visões podem ter. Com isso, optou-se pelo *mockup*, no qual possibilitou a criação de visões com componentes bem definidos e detalhados. A ferramenta *proto.io* foi escolhida pela sua facilidade de uso, variedade de componentes web e pela flexibilidade de escolher componentes desenhados com base em padrões visuais como *Material Design*, Windows 8, entre outros.

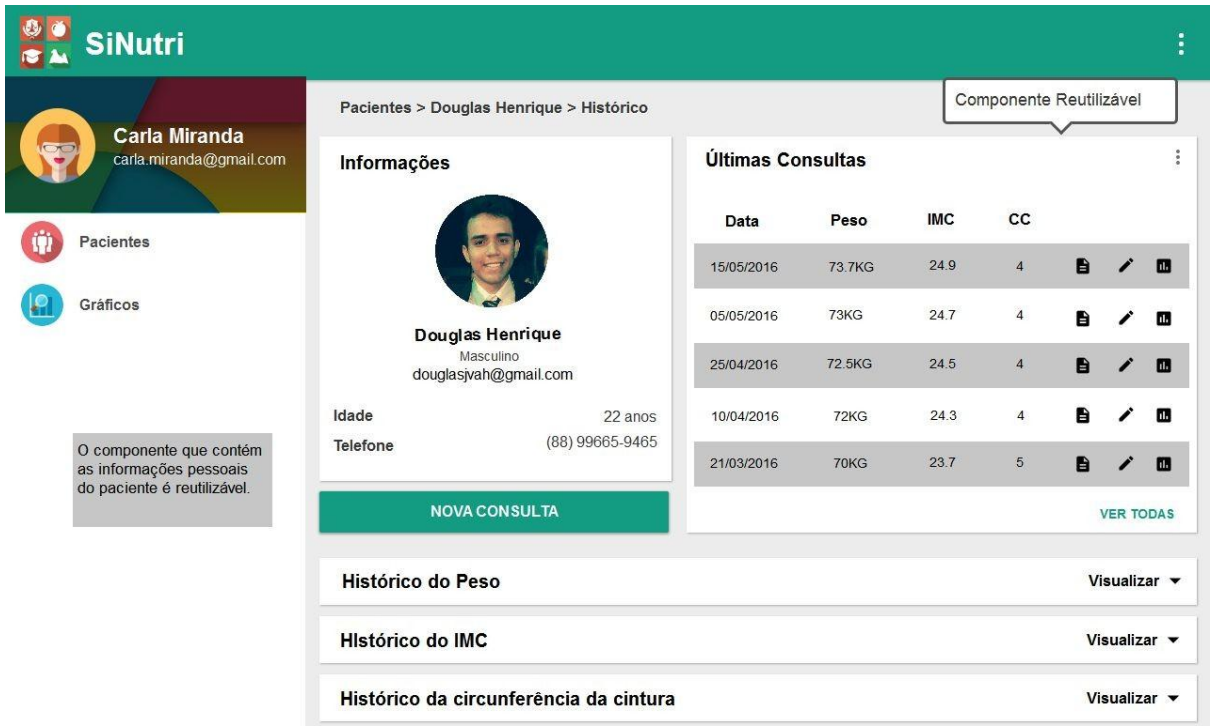
Como definido na etapa do processo detalhada no tópico 4.2, a primeira visão a ser projetada da aplicação em SPA, deve ser a visão base. No SiNutri, a visão base definida é apresentada na Figura 10.

Figura 10 - *Mockup* visão base da aplicação SiNutri em SPA

Fonte: Elaborada pelo autor

A visão base da aplicação SiNutri apresentada na Figura 10, é formada por três componentes: *Toolbar*, Menu de navegação e Componente principal. A *toolbar* contém um *menu* no lado direito com configurações da aplicação. O *menu* de navegação contém as opções para o usuário iniciar a navegação dentro da aplicação. O componente principal é o que será atualizado com todas as outras visões da aplicação SiNutri. Após criar a visão base da aplicação, foi criada a representação visual da visão histórico de consultas. A Figura 11 apresenta a visão histórico de consultas, criada utilizando a ferramenta *proto.io*.

Figura 11 - *Mockup* visão histórico de consultas



Fonte: Elaborada pelo autor

Para identificar os componentes, foi analisado a página de histórico de consultas da aplicação SiNutri MPA. A análise foi em busca das partes da página que satisfazem os requisitos, seja um formulário, gráfico, tabela ou listagem de dados. Para cada componente encontrado, foi analisada também a possibilidade de ser utilizado em outros locais da aplicação. Os seguintes componentes foram identificados para a visão de histórico:

- **Informações pessoais**.
  Componente reutilizável que satisfaz o requisito número 2 da lista de requisitos do tópico 5.1, responsável pelas seguintes informações pessoais do paciente: Foto, nome sexo, e-mail, idade e telefone. Esse componente pode ser reutilizado na página de edição de perfil do nutricionista.
- **Consultas nutricionais**.
  Componente reutilizável que satisfaz os requisitos número 1, 6 e 7 da lista de requisitos do tópico 5.1, responsável por listar as consultas nutricionais do paciente. Para cada consulta nutricional é exibida as seguintes informações: Data, peso, IMC e a circunferência da cintura. No momento que o nutricionista clica na opção “Ver todas”, a visão que lista todas as consultas é visualizada, a qual utiliza esse mesmo componente.

- **Gráfico histórico do peso**.
  Componente que também satisfaz o requisito número 3 da lista de requisitos do tópico 5.1, responsável pelo gráfico que informa o peso e a data de cada consulta realizada pelo paciente.
- **Gráfico histórico do IMC**.
  Componente que satisfaz o requisito número 4 da lista de requisitos do tópico 5.1, responsável pelo gráfico que informa o IMC e a data de cada consulta realizada pelo paciente.
- **Gráfico histórico da circunferência da cintura**.
  Componente que satisfaz o requisito número 5 da lista de requisitos do tópico 5.1, responsável pelo gráfico que informa a circunferência da cintura e a data de cada consulta realizada pelo paciente.

5.3 Back-end orientado a serviços

Com o *front-end* da página de histórico de consultas projetado, foi realizada a migração do *back-end* da aplicação SiNutri em MPA para um *back-end* orientado a serviços. Para realizar a migração do *back-end* foi utilizado como base o processo criado em Sá (2016).

O primeiro serviço criado no *back-end* orientado a serviços se chama getPessoa. O mesmo tem como objetivo atender as requisições da visão histórico de consultas projetada no tópico 5.3. Esse serviço ao ser requisitado retorna um objeto pessoa em formato JSON com base no CPF passado como parâmetro, esse objeto possui em seu corpo as informações pessoais e consultas nutricionais do paciente. Todas as entidades necessárias para a criação do serviço foi reaproveitada do *back-end* MPA. A camada de código responsável pela busca das informações pessoais e consultas nutricionais do paciente no banco de dados foi reaproveitada do método getPaginaHistorico listado no Quadro 3.

No decorrer da migração do *back-end* da aplicação SiNutri, foram encontradas dificuldades com a tecnologia utilizada para autenticação e controle de acesso da aplicação. Para contornar esse problema e prosseguir com a migração, essa tecnologia foi removida e as funcionalidades de autenticação e controle de acesso não foram disponibilizadas na aplicação em SPA.

5.4 Implementação do front-end

Para poder iniciar a implementação do *front-end* da aplicação SiNutri SPA, foi selecionado as seguintes tecnologias:

Quadro 4 - Tecnologias utilizadas para implementação do *front-end*

Nome	**Utilização**	**Versão**
Angular JS	Implementar SPA	1.5.5
Módulo *UI Router* do Angular JS	Roteamento de estados no Angular JS	0.2.18
Materialize CSS	Estilização da visões	0.97.6
Bower[14]	Gerenciar dependências *front-end*	1.5.2

Fonte: Elaborada pelo autor

Os critérios definidos para selecionar as tecnologias apresentadas no Quadro 4 acima foram:

- Simplicidade para criação de aplicações SPA
- Experiência do autor com as tecnologias
- Utilização da tecnologia no mercado

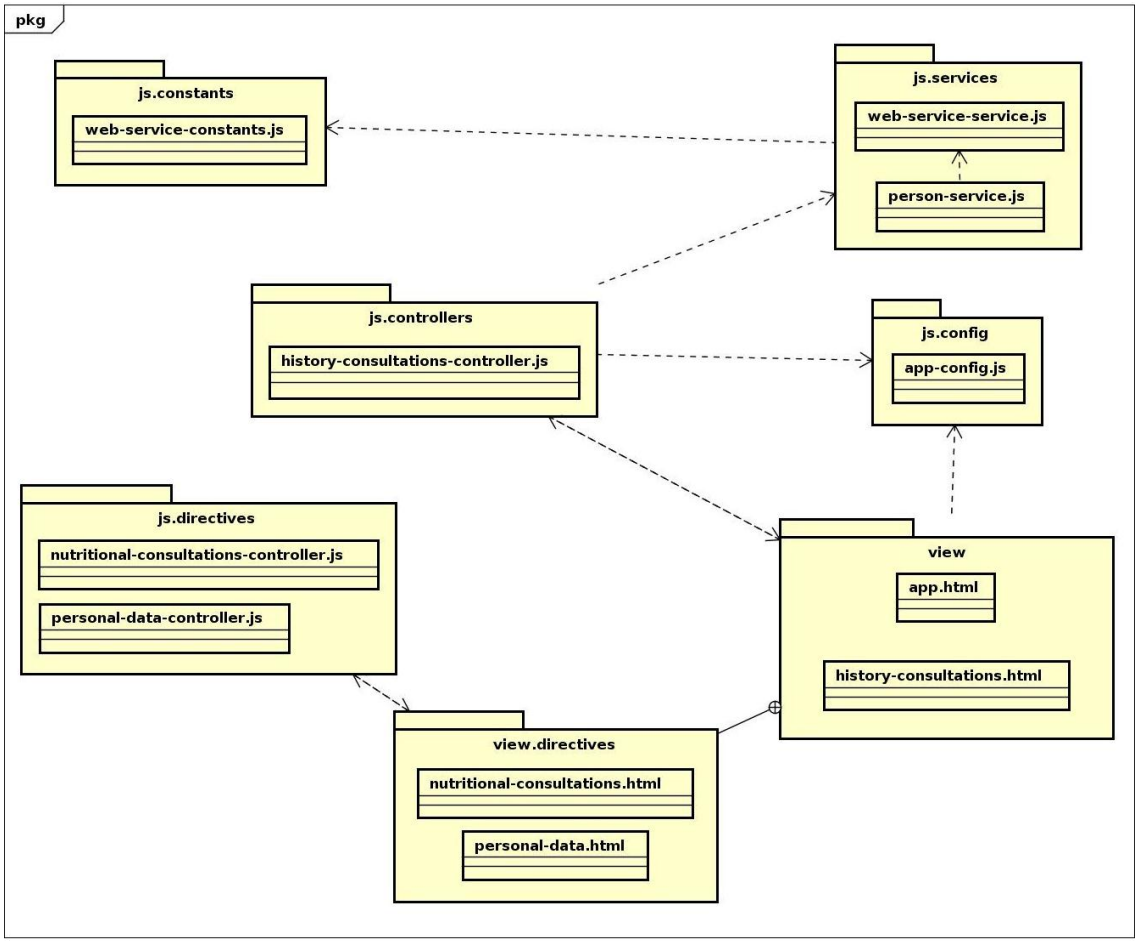
No tópico 5.2 foi projetada as visões da aplicação SiNutri. A primeira se trata da visão base da aplicação, representada na Figura 10, na qual possui um componente principal que será atualizado com as demais visões conforme a navegação do usuário na aplicação. A segunda foi à visão Histórico de consulta, que será injetada dentro do componente principal da visão base da aplicação, apresentando como resultado a visão representada na Figura 11. A implementação do *front-end* e suas visões seguiu a seguinte sequencia de passos:

1. Configuração do ambiente de desenvolvimento *front-end*.
2. Configuração dos *frameworks* utilizados com a ferramenta Bower.
3. Implementação (HTML, CSS e JavaScript) da visão base representada na Figura 10.
4. Implementação (HTML, CSS e JavaScript) da visão de Histórico de consultas representada na Figura 11.
5. Implementação do roteamento das visões com o Angular JS.

A Figura 12 apresenta a organização de pacotes *front-end* da aplicação SiNutri após a implementação ter sido realizada.

[14] https://bower.io/

Figura 12 - Organização de pacotes *front-end* da aplicação SiNutri SPA



Fonte: Elaborada pelo autor

Após a configuração do ambiente de desenvolvimento e dos *frameworks* utilizados, foi implementada a visão base representada na Figura 10. Para a visão base foi criado o arquivo app.html, no qual possui os seguintes componentes: *toolbar*, *menu* de navegação e componente principal que será atualizado com as outras visões de acordo com a navegação no *menu*.

Após a visão base, foi implementada a visão histórico de consultas. Essa visão é mais complexa, pois contém cinco componentes: Informações pessoais, Consultas nutricionais, Gráfico histórico do peso, Gráfico histórico do IMC e Gráfico histórico da circunferência da cintura. Entre os cinco componentes, o de Informações pessoais e consultas nutricionais foram implementados como diretivas. Diretivas no Angular JS são

basicamente extensões HTML que podem ser utilizadas em vários locais da aplicação. A visão histórico de consultas foi implementada da seguinte forma:

- **Diretiva informações pessoais.**

  A diretiva informações pessoais é formada pela *view nutritional-consultations* e pelo *controller nutritional-consultations-controller*. Para que essa diretiva seja populada, deve ser passado como parâmetro um objeto que contenha as informações do paciente. Essa diretiva pode ser reutilizada na visão de perfil do usuário autenticado na aplicação.

- **Diretiva consultas nutricionais**

  A diretiva consultas nutricionais é formada pela *view personal-data* e pelo *controller personal-data-controller*. Para que essa diretiva seja populada, deve ser passado como parâmetro uma lista de consultas nutricionais. Essa diretiva pode ser reutilizada em uma visão que lista todas as consultas nutricionais do paciente.

- **Visão histórico de consultas.**

  A *view history-consultations* e o *controller history-consultations-controller* são os responsáveis por iniciar a construção da visão histórico de consultas. O *controller history-consultations-controller* utiliza o *person-service* para realizar uma requisição ao serviço getPessoa no *back-end*. O serviço retorna um objeto Pessoa com as informações pessoais e consultas nutricionais do paciente. A *view history-consultations* ao receber o objeto Pessoa atualiza em seu corpo os gráficos de histórico de peso, IMC e circunferência da cintura, como também, passa para as diretivas de informações pessoais e consultas nutricionais os valores que precisam para serem populadas. O resultado ao final desse processo é a visão de histórico de consultas representada na Figura 11.

Com as duas visões do *front-end* desenvolvidas, foi implementado o roteamento de visões. O Angular JS junto com seu módulo *UI Router* para roteamento, trabalha com um provedor de estados da aplicação. Um estado corresponde a uma visão da aplicação. O módulo *UI Router* basicamente faz o controle de qual visão/estado da aplicação o usuário irá visualizar. A configuração de roteamento das visões da aplicação SiNutri pode ser encontrada dentro do arquivo *app-config*.

Com o roteamento das visões implementado, foi realizado testes básicos utilizando a aplicação como usuário final. Os testes tiveram resultados positivos. Todo o código da aplicação SiNutri em SPA está disponível no seguinte repositório:

- **https://github.com/douglasjva/sinutri-spa**.

5.5 Avaliar processo de migração

Além da migração realizada no estudo de caso, também foi formado um grupo de foco para avaliar o processo definido neste trabalho. Segundo Dias (2000), o principal objetivo de um grupo de foco é identificar percepções, sentimentos, atitudes e ideias dos participantes do grupo em relação a um determinado assunto.

Oito desenvolvedores foram convidados para participar do grupo de foco, mas apenas quatro compareceram. Os quatro participantes já trabalharam no desenvolvimento de aplicações no NPI-UFC. Inclusive, dois participantes já haviam trabalhado no desenvolvimento da aplicação SiNutri em MPA. Apenas um participante não conhecia a técnica SPA.

A reunião teve como introdução a explicação do objetivo da reunião, definição e exemplos de aplicações MPA e SPA e uma descrição breve do trabalho proposto. Após a introdução, foi apresentado os passos do processo de migração, sempre exemplificando com a aplicação SiNutri. Como os participantes já tinham conhecimento do SiNutri ou de aplicações parecidas, facilitou o entendimento das carateristas das aplicações MPA e SPA.

A dinâmica da reunião fluiu bastante pelo fato dos participantes estarem trabalhando com aplicações MPA. Conforme os passos do processo iam sendo discutidos, os participantes colocaram exemplos reais do dia-a-dia das aplicações que trabalham e era discutido como o processo de migração responderia a esses exemplos. Foi nessas discussões que foi identificado a necessidade de uma etapa no processo para testes na camada *front-end* da aplicação. Foi notado também, que na etapa de migração do *back-end* no processo, alguns participantes do grupo tiveram dificuldade para entender como é realizado e quais os resultados ao final. Os participantes acharam bastante interessantes e importantes à criação de *Mockups* para representar visões da aplicação, pois auxiliam no desenvolvimento e podem servir para validar com os gerentes de projeto o que será implementado. Também foi destacado nas discussões a importância e benefícios que a etapa de conhecer a aplicação a ser migrada pode trazer, considerada primordial para a realização correta da migração.

A reunião durou cerca de 1h30min. O processo foi aprovado pelos participantes e elogiado como bem estruturado, mas, é necessário dar atenção à parte de testes *front-end* e de uma explicação mais clara na migração do *back-end*.

6 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

Com a evolução da Web e o crescente número de usuários conectados a Internet, está se tornando cada vez mais necessárias aplicações Web que promovam uma boa experiência de utilização para o usuário. A partir dessa necessidade, esse trabalho definiu um processo e ferramentas para realizar a migração de uma aplicação MPA para SPA.

O processo definido neste trabalho é composto por cinco passos, são eles: Conhecer aplicação, Projetar SPA, Criar um *back-end* orientado a serviços, Selecionar tecnologias *front-end* e Implementar *front-end.* Como forma de validar o processo proposto, foi realizado um estudo de caso, onde uma aplicação MPA chamada SiNutri, com a utilização do processo foi migrada para SPA. A migração da aplicação SiNutri utilizando o processo foi bem-sucedida, porém, como dito, foram encontradas dificuldades com uma das tecnologias utilizadas na aplicação. O problema foi contornado e a migração prosseguiu.

Também foi formado um grupo de foco com objetivo de validar o processo de migração criado. Na reunião do grupo participaram quatro pessoas, nas quais fizeram observações para melhoria do processo e tiraram dúvidas sobre a técnica SPA. Também foi criado no decorrer do trabalho uma aplicação SPA básica, chamada SisEventos. Essa aplicação foi criada com objetivo de facilitar o entendimento da técnica SPA e suas tecnologias. Essa aplicação também poderá ser utilizada de forma didática por pessoas que desconhecem a técnica SPA.

Com a migração realizada no estudo de caso e os *feedbacks* do grupo de foco, foram identificadas algumas possibilidades de trabalhos futuros, são eles:

- Adicionar uma etapa ao processo para testes na camada *front-end*.
- Incorporar um *diagram flow* no processo. Esse diagrama irá apresentar uma visão geral da navegação da aplicação em SPA.
- Realizar a migração de todos os módulos do SiNutri.

REFERÊNCIAS

BURBECK, Steve. **Applications programming in smalltalk-80 (tm): How to use model-view-controller (mvc**). Smalltalk-80 v2, v. 5, 1992.

DIAS, Cláudia Augusto. Grupo focal: técnica de coleta de dados em pesquisas qualitativas. **Informação & Sociedade: Estudos**, v. 10, n. 2, 2000.

FENDER, JOE; YOUNG, CARWIN. (2014). **Front-end fundamentals.** Estados Unidos: LeanPub, 2015.

GONÇALVES, E. J. T. ; BEZERRA, C. I. M. ; ALMENDRA, C. C. ; SAMPAIO, A. L. ; VASCONCELOS, D. R.. **Núcleo de Práticas em Informática: Contribuindo para a Formação em Sistemas de Informação Através do Desenvolvimento de Projetos de Software.** In: XXXIII CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO - XXI Workshop sobre Educação em Computação (WEI), 2013, Macéio/AL. Anais do XXXIII CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO, 2013. v. 1. p. 601-610.

MESBAH, Ali; VAN DEURSEN, Arie. **Migrating multi-page Web applications to single-page Ajax interfaces.** In: Software Maintenance and Reengineering, 2007. CSMR'07. 11th European Conference on. IEEE, 2007. p. 181-190.

MIKOWSKI, Michael S.; POWELL, Josh C. **Single Page Web Applications**. Manning Publications Co, 2013.

MORALES-CHAPARRO, R. et al. **MVC Web design patterns and rich internet applications.** Proceedings of the Jornadas de Ingeniería del Software y Bases de Datos, 2007.

ROSA, Thiago Pereira. **Um método para o desenvolvimento de Software baseado em microsserviços**, 2016.

SÁ, André Luís Ferreira. **Migração de uma arquitetura monolítica para uma arquitetura orientada a serviços**, 2016.

SOUDERS, Steve. **High-performance Web sites**. Communications of the ACM, v. 51, n. 12, p. 36-41, 2008.

TESARIK, J.; DOLEZAL, L.; KOLLMANN, Christian. **User interface design practices in simple single-page Web applications.** In: Applications of Digital Information and Web Technologies, 2008. ICADIWT 2008. First International Conference on the. IEEE, 2008. p. 223-228.

WANG, Qingling et al. **An Automatic Approach to Reengineering Common Website with AJAX.** In: Next Generation Web Services Practices, 2008. NWESP'08. 4th International Conference on. IEEE, 2008. p. 185-190.