“Sistema de gerenciamento de módulos”

Por

Victor de Azevedo Nunes

Trabalho de Graduação

Universidade Federal da Bahia
ceapgmat@ufba.br
wiki.dcc.ufba.br/PMCC/

SALVADOR, Abril/2017

Universidade Federal da Bahia
Departamento de Ciência da Computação
Programa Multiinstitucional de Pós-graduação em Ciência da Computação

Victor de Azevedo Nunes

“Sistema de gerenciamento de módulos”

Trabalho apresentado ao Programa de Programa Multiinstitucional de Pós-graduação em Ciência da Computação do Departamento de Ciência da Computação da Universidade Federal da Bahia como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: *Ivan do Carmo Machado*

SALVADOR, Abril/2017

Eu dedico esta dissertação...

Agradecimentos

Meus agradecimentos...

Walking on water and developing software from a specification are easy if both are frozen

—EDWARD V BERARD

Resumo

Meu resumo

Palavras-chave: palavras chave

Abstract

My abstract...

Keywords: key words...

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Acrônimos

ALM Application Lifecycle Management

1 Introdução

O desenvolvimento de software normalmente encontra problemas relacionados ao cumprimento dos prazos de entrega. Tal situação pode ocorrer devido, a exemplo do aumento de demandas extras, desvios nos requisitos e problemas internos na equipe. Assim, a solução para atrasos de entrega pode estar em soluluções específicas, entretanto, existem elementos universais que podem mitigar ou combater esse problema, como o reuso de software. Este trabalho de conclusão de curso busca padronizar o controle de acesso dos softwares, abastraindo-o e tornando-o passível de maior reuso como um serviço reutilizável no desenvolvimento de software, reduzindo assim o tempo de desenvolvimento e manutenção.

1.1 Motivação

Otimizar o desenvolvimento de software é uma tarefa árdua e bastante estudada. A área de Engenharia de Software estuda maneiras de melhorar o desenvolvimento de software mediante processos e práticas de desenvolvimento. Qualquer melhoria durante a construção de um sistema, normalmente acaba por prover uma economia no orçamento de desenvolvimento e/ou manutenção. Assim, é possível inferir que melhorias no processo de desenvolvimento de sistemas, pode satisfazer as necessidades desde o programador, com um trabalho facilitado, até o cliente final com redução de custo.

1.2 Problema

O impacto dessa má prática atinge fortemente a manutenibilidade do software, grande transtorno diante de uma alteração que poderia ser simples caso. A abordagem do reuso

pode inclusive, abranger mais de um projeto numa organização, fazendo que até mesmo módulos inteiros sejam reaproveitados.

1.3 Objetivos

- Objetivo geral: Seguindo a boa prática de projetar o software, o objetivo geral deste trabalho é prover o reuso de código, utilizando padões que mantenham o componente de código na melhor forma possível para ser acoplado em outros softwares sem muitas dificuldades adicionais.

- Objetivo específico: O controle de acesso de um sistema, já que o mesmo está presente sem remodelações significativas nos sistemas de uma organização. Assim, o objetivo é desenvolver tal módulo como um serviço, deixando-o isolado e gerando a possibilidade de qualquer sistema consumir tal módulo como um serviço. Dessa forma, ao projetar um novo software para uma organização por exemplo, ao invés de incluir o módulo de controle de acesso, projetaria-se o código para consumir o serviço de contrle de acesso que seria um projeto à parte. Ele também apresenta a vantagem de ter interoperabilidade entre linguagens de programação, já que realiza a comunicação através de JSON, seguindo padrões atuais. Assim, tem-se uma constância no módulo de acesso independentemente da possível necessidade de migração de tecnologia.

1.4 Metodologia

A fim de selecionar as tecnologias a serem utilizadas na implementação deste trabalho, nesta seção as mesmas serão apresentadas com base nas indústria de desenvolvimento de software. (estado da prática). Para embasar esse cenário, serão exibidas estatísticas que refletem a situação do mercado.

Baseando-se nas tecnologias no cenário atual do desenvolvimento, dispomos de algumas opções eficientes para a implementação da solução. Dentre as possibilidades, considerando a facilidade para futura manutenção e continuidade do projeto, tende-se a optar por uma tecnologia popular. A linguagem de programação adotada é PHP. A escolha é fundamentada na pesquisa da RedMonk de 2015 [1] , que evidencia o uso das linguagens de programação de acordo com as discussões no StackOverflow e repositórios no GitHub. É possível constatar a popularidade do PHP no cenário atual apresentado em

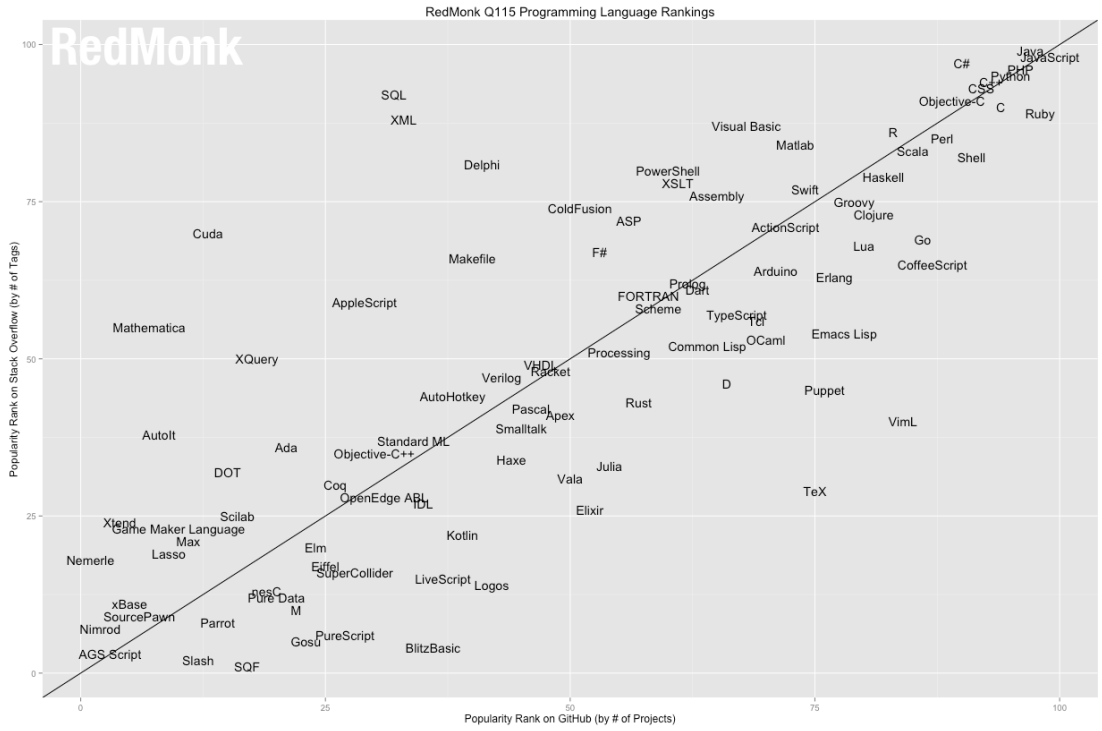

Figura 1.1 Ranking das liguagens de programação no Stack Overflow e Github

1.4, na qual o PHP é apresentado na terceira colocação, apenas atrás do lider JavaScript e do segundo colocado, o Java.

Entretanto, não seria inteligente desenvolver um sistema completo sem o auxílio de um framework. Dentre os frameworks disponíveis para PHP, hoje o destaque está com o Laravel, que se encontra no topo dentre os mais utilizados no momento.

A WebHostFace, uma empresa de hospedagem, compilou várias estatísticas para criar um infográfico mostrando os frameworks PHP mais populares de 2015. Utilizando informações sobre os próprios clientes, o Google Trends, estatísticas de repositórios do GitHub e a pesquisa do SitePoint “Best PHP Frameworks 2015”, a WebHostFace elaborou o gráfico 1.4.

Como pode ser verificado no gráfico 1.4, tem-se a evidência que o Laravel em 2015 teve a maior popularidade em projetos pessoais e tem a maior comunidade entre os concorrentes, o que o torna uma boa escolha para a escrita de um software que será continuado por terceiros.

Para elaborar os recursos de interface e integrar ao back-end PHP do sistema, será adotado o já conhecido AngularJS, ferramenta sólida e conhecida no aspecto em questão.

Dados coletados via Google Trends em, que propõe comparações entre termos pes-

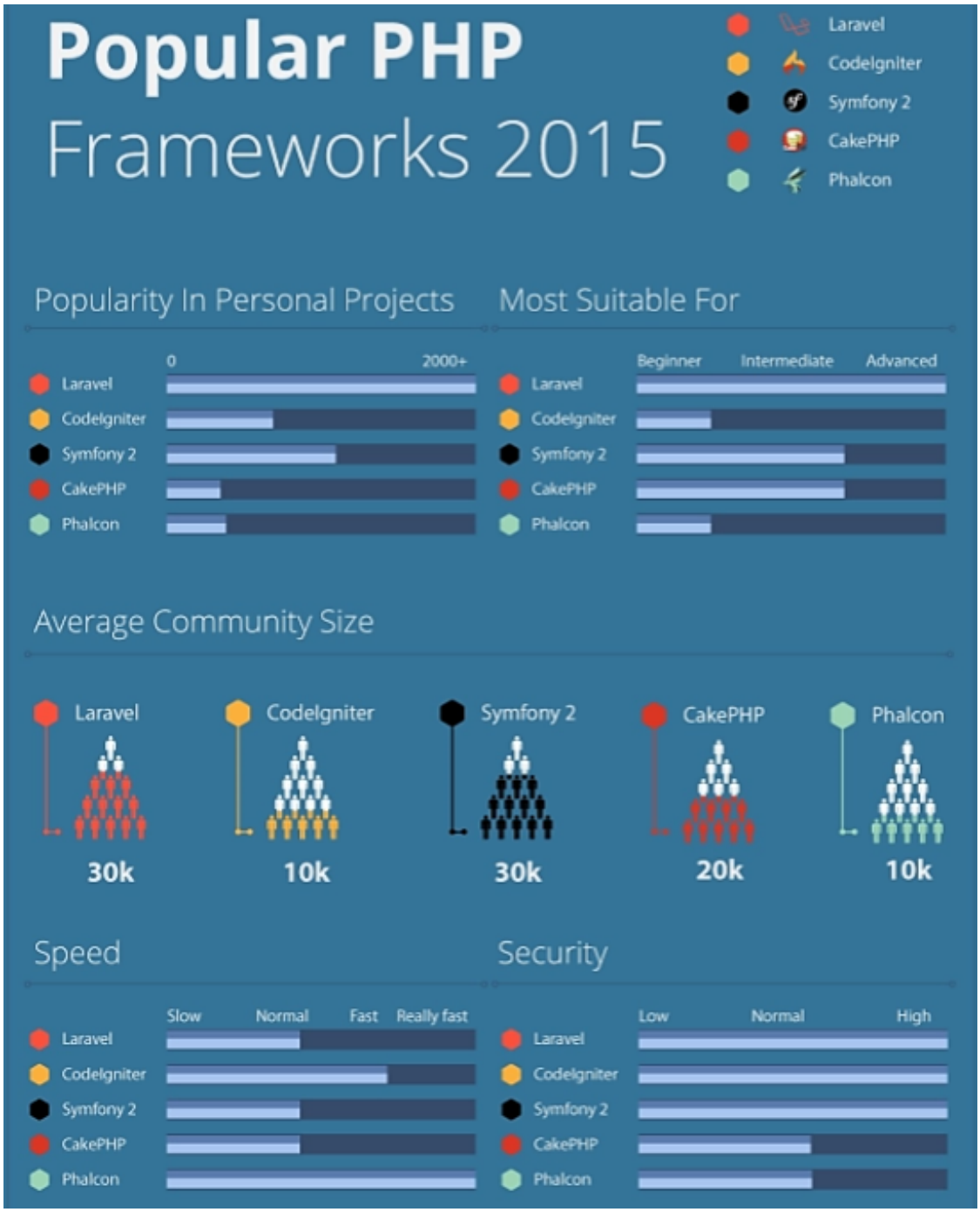

Figura 1.2 Infográfico da WebhostFace, exibindo a popularidade dos Frameworks PHP em 2015

Figura 1.3 Gráfico do Google Trends exibindo as pesquisas por ferramentas front-end

quisados, revela a popularidade do AngularJS diante de alguns dos principais concorrentes. O gráfico 1.4 evidencia o cenário.

1.5 Resultados Esperados

Realizando o uso do proposto módulo de controle de acesso como serviço, espera-se que ocorram os seguintes ganhos:

- Servir a projetos de diferentes liguagens.
- Atender a projetos de diferentes plataformas (Web, Desktop, Mobile).
- Facilidade de configuração das permissões.
- Extinção do módulo de permissões nos projetos em que o trabalho proposto será aplicado.

1.6 Fora de Escopo

Diante da possibilidade de gerência dos sistemas, módulos e funcionalidades contratadas, seria possível estender o sistema aqui proposto, complementando-o com uma

abordagem financeira, controlando o acesso do cliente mediante o pagamento do que está contratado pelo cliente. Também existe a possibilidade de realizar o controle por número de usuários que estão logados no sistema contratado ou por módulo, mediante uma evolução na implementação. Uma possível solução para a última situação seria o sistema consumidor enviar uma notificação (post) ao sistema de gerenciamento de módulo num determinado intervalo de tempo, o usuário logado no sistema. O sistema de gerenciamento de módulos responderia internamente à sinalização, mantendo registro de usuário ativo. Caso o usuário final saia do sistema, enviaria o sinal de logout e caso fechasse o navegador, a sessão seria encerrada por tempo de inatividade (falta de envio da notificação de uso).

1.7 Estrutura do Trabalho

Neste tópico, tem-se uma breve apresentação dos capítulos que compõem este trabalho de conclusão de curso:

- Capítulo 1: Referencial teórico, fundamentando os principais conceitos que compõem os elementos marcantes do trabalho.
- Capítulo 2: Introdução, apresentando uma visão geral do trabalho.
- Capítulo 3: Proposta, contendo o detalhamento do trabalho proposto, baseando-se nos aterfatos da engenharia de software.
- Capítulo 4: Análise
- Capítulo 5: Conclusão

2

Referencial Teórico

Projetar o desenvolvimento de um software requer muito planejamento, pois as falhas iniciais podem custar bastante caro ou até mesmo inviabilizar a continuação de um projeto. Assim, a escolha da arquitetura ideal para a aplicabilidade é essencial na concepção de um produto de software. De todo o modo, sempre busca-se fazer mais com menos. Diante de tal filosofia, temos neste capítulo, uma breve discussão sobre alguns elementos de projeto e arquitetura de software, a fim de contextualizar este trabalho de conclusão de curso. O capítulo corrente é composto por quatro seções. A 2.1 trata de Software como serviço, discutindo alguns elementos do contexto que são relevantes para o trabalho proposto. A 2.2 discute sobre a empregabilidade do reuso de software. A 2.3 seção, trata sobre aspectos envolvidos na modularização dos softwares. Por fim, a 2.4 aborda as aplicações web, discutindo sobre aspectos relevantes sobre a aplicação web que compõe este trabalho.

2.1 Software como serviço

Segundo La e Chun [3], o princípio da definição de Software como um Serviço (Sofware as a Service - SaaS) é um serviço complementar para aplicações da computação em nuvem (cloud computing). No entanto, as áreas não se confundem. SaaS deve ser entendido como um mecanismo de suporte às soluções existentes na cloud. Os SaaS existem justamente para maximizar o reuso de serviços repetidos e não centrais em uma aplicação remota.

Como vantagens, diversos fatores podem ser favoráveis para a adoção de um SaaS, como custo e manutenção dentre outros fatores aplicáveis a soluções específicas. Lechesaet al. [4], quantificam índices sobre os fatores determinantes para adoção ou não de um SaaS voltado para ERP na África do Sul. Os principais fatores determinantes para

adoção desse mecanismo de software são sua fluidez quanto à rede e a segurança. Esses fatores estão presentes na aplicação desenvolvida neste trabalho de conclusão de curso.

Devido ao fato de ter um serviço constantemente na nuvem, fica o questionamento sobre a segurança da informação manipulada. Sabe-se que a vulnerabilidade na Web não é restrita ao SaaS, atingindo diversos âmbitos. Rai et al. [5] consideram que o avanço da computação em nuvem não é um problema apenas para os serviços Web do ponto de vista da segurança, pois muitos trabalhos na literatura mostram a área como mais um ponto de vulnerabilidade para diversos setores, a exemplo de infraestrutura. No mesmo artigo mencionado de Rai et al. [5], também realizaram-se estudos exploratórios junto a empresas usuárias de serviços em computação em nuvem e consideram que a perspectiva de SaaS também pode fortalecer a segurança nas aplicações de cloud computing, pois o software de autenticação compartilhado por várias aplicações em nuvem, oferece uma melhor padronização e consequente facilidade de prevenção a erros de vulnerabilidade específicas de cada módulo da pesquisa. Esse ponto de vista é fundamenta para a compreensão de SaaS.

A arquitetura de armazenamento de dados de um Saas pode variar de acordo com a necessidade do contexto. Huixin [6] descreve possíveis modelagens nesse sentido. Essa abordagem pode ser com um banco de dados único, fazendo com que diferentes clientes compartilhem o mesmo banco, diferindo os dados através de controle de usuário, ou isolando os diferentes clientes através de bancos de dados exclusivos para cada um. Esse fator também pode ser combinado com a arquitetura da aplicação, caso ofereça aplicação única para todos os clientes ou aplicação compartilhada. Diante das possíveis abordagens, a modelagem de dados do software pode ser decidida pela regra de negócio. Este trabalho optou por aplicação única e banco de dados compartilhado.

Devido ao conceito particular de obtenção de software, tanto pela visão do cliente como pela visão do vendedor, é necessário compreender os diversos aspectos que podem ser relevantes ao orçar um Saas. O recente trabalho de T. Kaur et al. [7] orienta um modelo para compor o custo de um Saas. O custo total seria composto pelos fatores que dão suporte ao funcionamento do software. Tais fatores incluem infra-estrutura, configurabilidade, customização, parâmetros de QoS (Quality of service) como escalabilidade, disponibilidade, usabilidade, pontualidade e desempenho da resposta, portabilidade, custo total de propriedade e retorno do investimento. Esses fatores caracterizam o custo de forma eficaz, possibilitando ao fornecedor, prover um Serviço de acordo com a exigência do consumidor em vários pacotes de serviços.

O conceito de software como serviço se aplica neste trabalho de conclusão de curso,

pois o mesmo estará disponível na web com alta disponibilidade adotando as características apresentadas para qualquer pessoa que desejar usá-lo. Assim, basta se cadastrar, configurá-lo e usar como um serviço, sem preocupação com a manutenção do mesmo.

2.2 Reuso de software

De acordo com o livro Software pratical reuse [8], o reuso de software é a utilização de qualquer informação que um desenvolvedor pode necessitar no processo de criação de software. O livro de Basili e Rombach [9] define reutilização de software como o uso de tudo o que está associado a um projeto de conhecimento. Assim, o objetivo da reutilização de software é reciclar o design, código e outros componentes de um produto de software e assim reduzir o custo, o tempo e melhorar a qualidade do produto. Segundo Keswani et al. [10], o componente reutilizável de software pode ser qualquer parte de seu desenvolvimento, como um fragmento de código, design, casos de teste, ou até mesmo a especificação de requisitos de uma funcionalidade do software.

O reuso de software pode ter impacto positivo em diversos aspectos do software, vejamos alguns, conforme apresentados no C.R.U.I.S.E Book [11] :

- Qualidade: As correções de erro tornam-se úteis em todos os locais em que ocorreu, padronizando e facilitando a manutenção.
- Produtividade: O ganho de produtividade é alcançado devido ao menor número de artefatos desenvolvido. Isso resulta em menor esforço de teste e também análise e design, reduzindo custos.
- Confiabilidade: A utilização de componentes bem testados aumenta a confiança no software. Além disso, a utilização de um mesmo componente em vários sistemas, aumenta a possibilidade de detecção de erros e reforça a confiança no componente.
- Redução do Esforço: A reutilização de software proporciona uma redução do tempo de desenvolvimento, o que reduz o tempo necessário para o produto ser disponibilizado no mercado para trazer rentabilidade.
- Trabalho redundante e tempo de desenvolvimento: Desenvolver um sistema do zero significa desenvolvimento redundante de muitos componentes, como requisitos, especificações, casos de uso, arquitetura, etc. Isso pode ser evitado quando

estes estão disponíveis como componentes reutilizáveis e podem ser compartilhados, resultando em um processo de desenvolvimento otimizado.

- Documentação: Embora a documentação seja muito importante para a manutenção de um sistema, muitas vezes é negligenciada. A reutilização de componentes de software reduz a quantidade de documentação a ser escrita, entretanto depende da qualidade do que está escrito. Assim, apenas a estrutura do sistema e os novos artefatos desenvolvidos necessitam ser documentados.

- Custo de manutenção: Menos defeitos e manutenções são esperados quando tem-se comprovada a qualidade dos componentes utilizados.

- Tamanho da equipe: É comum haver casos em que a equipe de desenvolvimento sofre sobrecarga. Entretanto, dobrar o tamanho da equipe de desenvolvimento não necessariamente duplica produtividade. Se muitos componentes podem ser reutilizados, é possível desenvolver com equipes menores, levando a melhor comunicação e aumento da produtividade.

Apesar dos benefícios da reutilização de software, ela não é suficientemente aproveitada. Existem fatores que influenciam direta ou indiretamente na sua adoção. Esses fatores podem ser de aspecto gerencial, organizacional, econômico, conceitual ou técnico. Veremos a seguir alguns aspectos que podem gerar conflito com a cultura de reuso de software, segundo o C.R.U.I.S.E Book [11] :

- Falta de apoio da gestão: Como a reutilização de software gera custos iniciais, a medida pode não ser amplamente alcançada em uma organização sem o apoio de alto nível de gestão. Os gestores têm de ser informados sobre os custos iniciais e serem convencidos sobre economias futuras.

- Gerenciamento do Projeto: Gerenciar projetos tradicionais é uma tarefa árdua, principalmente, os que praticam a reutilização de software. Utilizando a técnica em larga escala, tem-se impacto sobre todo o ciclo de vida do software.

- Estruturas organizacionais inadequadas: As estruturas organizacionais devem considerar diferentes necessidades que surgem quando a reutilização em larga escala está sendo adotada. Por exemplo, uma equipe particionada pode ser alocada somente para desenvolver, manter e certificar componentes reutilizáveis de software.

- Incentivos de gestão: É comum a falta de incentivo para deixar os desenvolvedores gastarem tempo elaborando componentes do sistemas. A produtividade é muitas vezes medida apenas no tempo necessário para concluir um projeto. Assim, fazer qualquer trabalho além disso, embora benéfico para a empresa como um todo, diminui o seu sucesso. Mesmo quando os componentes reutilizáveis são utilizados, os benefícios obtidos são uma pequena fração do que poderia ser alcançado caso houvesse reutilização explícita, planejada e organizada.

- Dificuldade de encontrar software reutilizável: Para reutilizar os componentes, devem existir formas eficientes de busca. Além disso, é importante ter um repositório bem organizado contendo componentes com um eficiente meio de acesso.

- Não reutilização do software encontrado. O acesso fácil ao software existente não necessariamente aumentar a reutilização. Os componentes reutilizáveis devem ser cuidadosamente especificados, projetados, implementados e documentados, pois em alguns casos, modificar e adaptar o código pode ser mais custoso que a programação da funcionalidade necessária a partir do zero.

- Modificação: É muito difícil encontrar um componente que funcione exatamente da mesma maneira que queremos. Desta forma, são necessárias modificações e devem existir formas de determinar os seus efeitos sobre o componente.

A definição do reuso de software pode ser associada a este trabalho, pois ainda que como um serviço, estaremos adotando uma aplicação única para servir aos mais diversos softwares de uma empresa, por exemplo. Logo, ao adotar o projeto, pode ser interessante aplicar algumas métricas para tomar conhecimento de possíveis vantanjens como as citadas nessa seção, a exemplo de economia de tempo de desenvolvimento, e custo do projeto.

2.3 Modularização

Conforme é frisado por Wickramaarachchi e Lai [12], o conceito de modularização na indústria de software tem uma longa história e tem sido utilizado para melhorar o processo de desenvolvimento de software em diferentes estágios. Os principais conceitos por trás da modularização do software foram introduzidos por pesquisadores pioneiros há quarenta anos, com uma notável contribuição feita por Melvin Conway e David Parnas, que tem representação notável na engenharia de software.

Modularizar um software é um bom padrão a ser adotado. Segundo Wickramaarachchi e Lai [12], a modularização é importante na identificação de dependências e reduz as dificuldades diante de uma possível necessidade de grandes alterações. De uma perspectiva da engenharia de software, uma modularização geralmente tem várias vantagens, tais como: tornar a complexidade do software mais gerenciável, facilitar o trabalho paralelo e tornar o software mais maleável para acomodar o futuro incerto que um software pode ter. O objetivo final da modularização do software é aumentar a produtividade ea qualidade do software. Tal conceito encontra-se bastante difundido e estái incorporado em linguagens de programação e ferramentas de software. O presente trabalho favorece ao uso da modularização de um software e até mesmo pode ser considerado um módulo a ser acoplado a qualquer software, mediante a compatibilidade.

Ao realizar a adoção do trabalho proposto, fica bastante evidente a "responsabilidade"do mesmo no escopo do projeto. Assim, é possível tratá-lo como um módulo do projeto que o usa, mesmo que seja consumido como um serviço.

2.4 Aplicações web

A popularidade das soluções Web aumentou exponencialmente na última década e todos os dias cresce o número de pessoas usuárias desse tipo de software. E seguindo um padrão próprio, Kumar et al. [13] sugerem que para o desenvolvimento web, deve-se manter a prática eficaz de produzir diagramas UML. A abordagem baseada na web oferece uma maneira fácil e eficaz para gerenciar e controlar o processo de desenvolvimento por meio de artefatos de modelagem. Tal abordagem pode ser usada quando há uma exigência de lidar com mudanças muito rápidas e grandes em requisitos de forma muito eficaz em muito menos tempo, gerando assim um menor impacto.

Para atender à fomentada demanda de aplicativos web, é necessário adotar métodos de desenvolvimentos que sejam ágeis, eficientes e de fácil manutenção. Yu Ping et al. [14] propõem o uso do modelo MVC (Model, View e Controller) no desenvolvimento para softwares web. O modelo apresentado tornou-se um padrão popular e divide o software em camadas com propósito definido, tornando-o de mais fácil manutenção.

O Ajax (Asynchronous Javascript and XML) revolucionou a web. Conforme demonstrado por Yuping [15], ao usar a tecnologia Ajax, podemos enriquecer a experiência do usuário em aplicações baseadas em navegador de internet, e fornecer uma variedade de aplicações interativas para atender às necessidade de humanização das aplicações. Os aplicativos Ajax em execução no navegador se comunicam com um servidor Web de

forma assíncrona e atualizam apenas uma parte da página.

No artigo de Tesarik et al. [16], temos que o design de software SPA (Single page application) é uma maneira interessante de criar um software Web numa uma única página. Essa solução de página única sem navegação funciona apenas com base em técnicas dinâmicas e assíncronas, como o AJAX (citado no parágrafo anterior). No entanto, esta abordagem coloca o desenvolvedor antes de alguns desafios substanciais. Para projetar a interface do usuário que mostra as informações, é importante projetar corretamente a tela para manipular os dados do aplicativo numa única página. O design da página deve ser elaborado para maximizar a decomposição da página em componentes distintos que encapsulam os principais casos de uso. Também recomenda-se que para elaborar os artefatos visuais, explore os mais ricos recursos na implementação, como HTML5, JavaScript, Ajax, CSS3, e outras tecnologias que se apliquem. Entretanto, desenvolver uma rica interface com o uso de diversas tecnologias/frameworks pode ocasionar um esforço maior para explorar as possibilidades. Tais componentes da interface de uma aplicação SPA normalmente são alimentados mediante o consumo de uma API Rest via requisições AJAX. Assim, é possível particionar as responsabilidades de processamento do software entre cliente e servidor.

Como é possível denotar no trabalho de Salvadori e Siqueire [17], REST é uma arquitetura muito popular para integração de aplicativos web, que permite compartilhar e reutilizar informações através de sistemas. Aplicações de grande escala baseadas em no REST deve ser implementadas utilizando estratégias e mecanismos para produzir sistemas que sejam fáceis de desenvolver, reutilizar e manter. As interfaces de integração fornecidas por esses sistemas, chamadas de Web API, têm uma influência importante nas características da implementação resultante, pois a sua forma de resposta deve ser casada com a interface que o consome.

Os elementos comentados nessa seção encontram-se presentes neste trabalho de conclusão de curso. Juntos, montam a estrutura tecnológica necessária aliada à arquitetura adotada. Esses elementos seguem tendências atuais dos softwares web, fazendo com que esse trabalho esteja composto por tecnologias modernas que estão com boa aceitação no mercado.

3
Proposta

A adoção de padrões é uma premissa básica para o desenvolvimento de software, pois ao trabalhar com variações dentro de uma organização, a tendência é gerar confusão. Assim, o objetivo deste trabalho é projetar e implementar um módulo padrão de permissões de acesso, de modo que o mesmo possa ser utilizado por diferentes plataformas, como desktop, Web e mobile, eliminando assim este módulo do desenvolvimento de um software e utilizando o trabalho proposto como um SaaS provedor das permissões do software cliente a ser desenvolvido.

É comum encontrar frameworks de desenvolvimento de software que automatizam a geração de esquemas de configuração de módulos com perfis de acesso. Entretanto, apesar de ser um facilitador, tal política pode se tornar confusa em algus cenários, a exemplo de uma empresa que trabalha com sistemas sob encomenda, onde o cliente pode até mesmo determinar a linguagem ou framework de desenvolvimento. Nesse caso, a empresa teria que treinar os seus colaboradores a configurar os softwares para cada uma das ferramentas utilizadas, o que potencialmente ocasionaria dificuldades de entendimento comum.

Diante da possibilidade de manter a configuração dos sistemas de uma mesma instituição com diferentes módulos de permissão, seria ideal que todos os sistemas de software por ela desenvolvidos utilizassem um mesmo módulo de configuração de persmissão de acesso, para que esta parte comum, presente na maioria dos softwares, fosse padronizada.

O capítulo corrente é composto por três seções. A seção 3.1 trata da arquitetura do software proposto, contendo informações sobre as tecnologias utilizadas, diagramas UML e os requisitos do sistema. A seção 3.2 demonstra o fluxo do processo empregado na aplicação desenvolvida e por fim, a seção 3.3 trada do que há de novo sob o ponto de vista tecnológico.

3.1 Arquitetura da aplicação

A aplicação desenvolvida neste trabalho foi arquitetada com o modelo SPA(Single page application) utilizando o AngularJs. Tal arquitetura se tornou tão popular quanto o famoso MVC (Model-View-Controller), e é bastante utilizada no desenvolvimento de aplicações web e mobile. Em linhas gerais, o conceito SPA tende a reduzir a necessidade de se implementar código server-side, e sim potencializar as ações em client-side. Assim, boa parte da aplicação passa a ser processada no cliente (dentro do navegador Web).

não faça simplesmente copiar de um blog. tente ir além, e prover fundamentação contida em livros-texto da área. um sugestão: capture as ideias do autor do blog, porém complemente-as com outras fontes, assegurando-se de que o que o autor do blog descreve de fato faz sentido na prática

Algumas vantagens da SPA:

- Partilha de processamento do software, visto que a aplicação consome uma API REST(back-end) e deve tratar os dados para a exibição no front-end.

- Com da partilha do processamento, tem-se uma menor codificação no servidor. Tal aspecto vem acompanhado com a distinção das responsabilidades, pois apenas o código do front-end trata de interface.

- Diante das características de uma SPA, a "página única"tende a ser de fácil entendimento aos usuários, simplificando a navegação.

- Como os acessos via dispositivos móveis é bastante significativo, há uma relevância no consumo de dados do software. Nesse quesito, uma SPA se comporta bem, devido à forma como consome os dados, com requisições AJAX que retornam JSON(Javascript object notation), que na realidade se resume apenas a dados estruturados.

O grande ator de app SPA é o código Javascript executado no cliente. Toda a aplicação pode ser construída simplesmente manipulando-se o DOM (Document Object Model) de forma nativa, ou com o uso de bibliotecas e frameworks Javascript que auxiliam na construção da aplicação. Estas bibliotecas e frameworks fornecem recursos para manipulação dinâmica do DOM, definição de templates de tela, chamadas assíncronas ao servidor, organização do código Javascript, etc. Dentres as diversas bibliotecas Javas-

cript disponíveis, tem-se entre as mais difundidas: AngularJs[1] , VueJs[2], Backbone[3], ReactJs[4], Ember[5] e outras.

Do lado servidor, tem-se a execução das linguagens de programação tradicionais como PHP, ASP.NET, JSP e etc. Assim, de acordo com a necessidade, as mesmas provêem host de arquivos, acesso a banco de dados e tratam regras de negócios que não podem estar no código JavaScript do Front-end por questões de segurança. E é do lado servidor que a arquitetura REST (Representational State Transfer) pode ser utilizada, com o intuito de fornecer serviços do servidor à aplicação SPA proposta neste trabalho. É comum encontrar aplicação SPA utilizando serviços RESTFul. Uma aplicação no servidor que utiliza a arquitetura REST para prover serviços, então é chamada de RESTFul. Neste trabalho, foi desenvolvida uma aplicação RESTFul com o Framework PHP Laravel.

Ao construir uma aplicação utilizando a arquitetura REST, o protocolo HTTP é usado em sua essência, utilizando os métodos de requisição ao servidor: GET, POST, PUT e DELETE (os mais comuns), e cada um deles indica uma determinada ação a ser executada em um recurso específico do servidor.

A seguir, a subseção 3.1.1 trata das tecnologias utilizadas no trabalho proposto, a 3.1.2 apresenta os diagramas UML do projeto e 3.1.3 trata das funcionalidades, formalizando os requisitos funcionais e não funcionais.

3.1.1 Tecnologias utilizadas

Laravel framework

Laravel [6] é um framework PHP livre e open-source para o desenvolvimento de sistemas Web que utilizam o padrão MVC (model, view, controller). O Laravel foi desenvolvido sob o MIT License, com o código-fonte hospedado no GitHub. Em Agosto de 2015, o Laravel já era o principal framework de projetos PHP no GitHub.

Algumas características proeminentes do Laravel são sua sintaxe simples e concisa, um sistema modular com gerenciador de dependências dedicado, várias formas de acesso a banco de dados relacionais e vários utilitários indispensáveis no auxílio ao desenvolvimento e manutenção de sistemas. Diante da popularidade do framework, tem-se

[1] https://angularjs.org/
[2] https://vuejs.org/
[3] http://backbonejs.org/
[4] https://facebook.github.io/react/
[5] http://emberjs.com/
[6] https://laravel.com/

uma grande comunidade, o que facilita a aprendizgem pois facilmente encontra-se tutoriais. Inclusive além da documentação, o próprio Laravel disponibiliza o Laracasts [7], que traz aos desenvolvedores uma série de vídeos ensinando as mais diversas funcionalidades encontradas no Laravel.

AngularJs

AngularJS é um framework JavaScript open-source, mantido por Google, que auxilia na execução de SPA. O framework lê o HTML que contém tags especiais do framework e então executa a diretiva na qual esta tag pertence, e faz a ligação entre a apresentação e seu modelo, representado por variáveis JavaScript comuns. O framework adapta e estende o HTML tradicional para uma melhor experiência com conteúdo dinâmico, com a ligação direta e associação bidirecional dos dados (two-way data-binding) que permite sincronização automática de models e views. Como resultado, AngularJS abstrai a manipulação do DOM e melhora os testes.

Bootstrap

Bootstrap é um popular framework front-end que facilita a criação de sites com tecnologia responsiva. O Bootstrap possui diversos componentes (plugins) em JavaScript (jQuery) que auxiliam o desenvolvedor a implementar, menu-dropdown, modal, carousel, slideshow, entre outros com facilidade, apenas acrescentando algumas configurações no código.

3.1.2 Diagramas

Nesta seção, temos a apresentação de alguns diagramas UML (Unified Modeling Language) com a finalidade de embasar o trabalho proposto, permitindo representar o sistema de forma padronizada.

A imagem 3.1.2 traz o diagrama de casos de caso, com o objetivo de os usuários obterem melhor entendimento das principais funcionalidades de sistema.

Com a imagem 3.1.2, é possível compreender com mais clareza o funcionamento do back-end do software, visualizando as classes utilizadas para manipular os dados.

Por fim, para prover um entendimento de funcionamento interno do software, a imagem 3.1.2 traz o diagrama de entidade relacionamento, fornecendo uma ilustração da organização do banco de dados utilizado.

[7] https://laracasts.com/

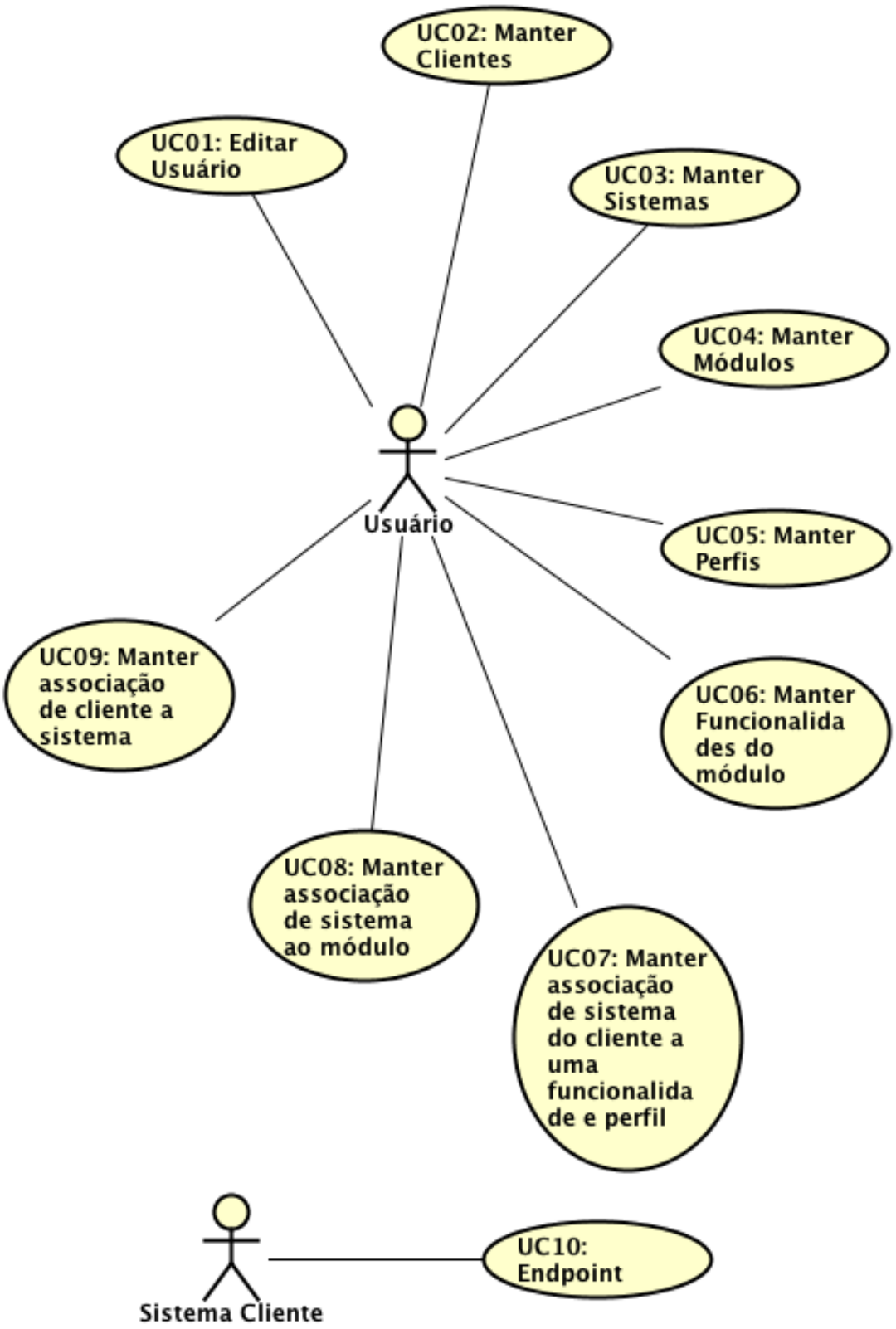

Figura 3.1 Diagrama de caso de uso

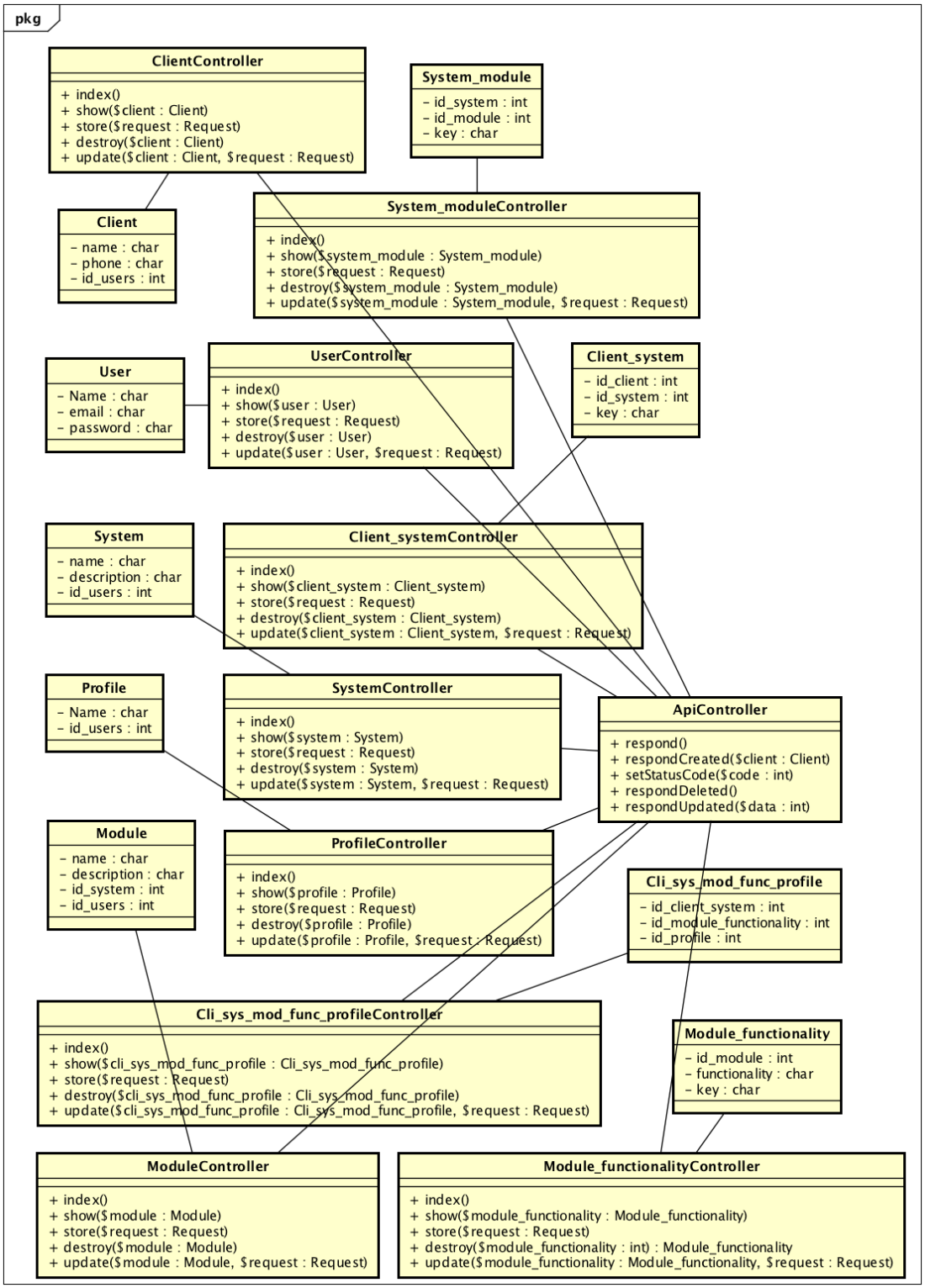

Figura 3.2 Diagrama de classe

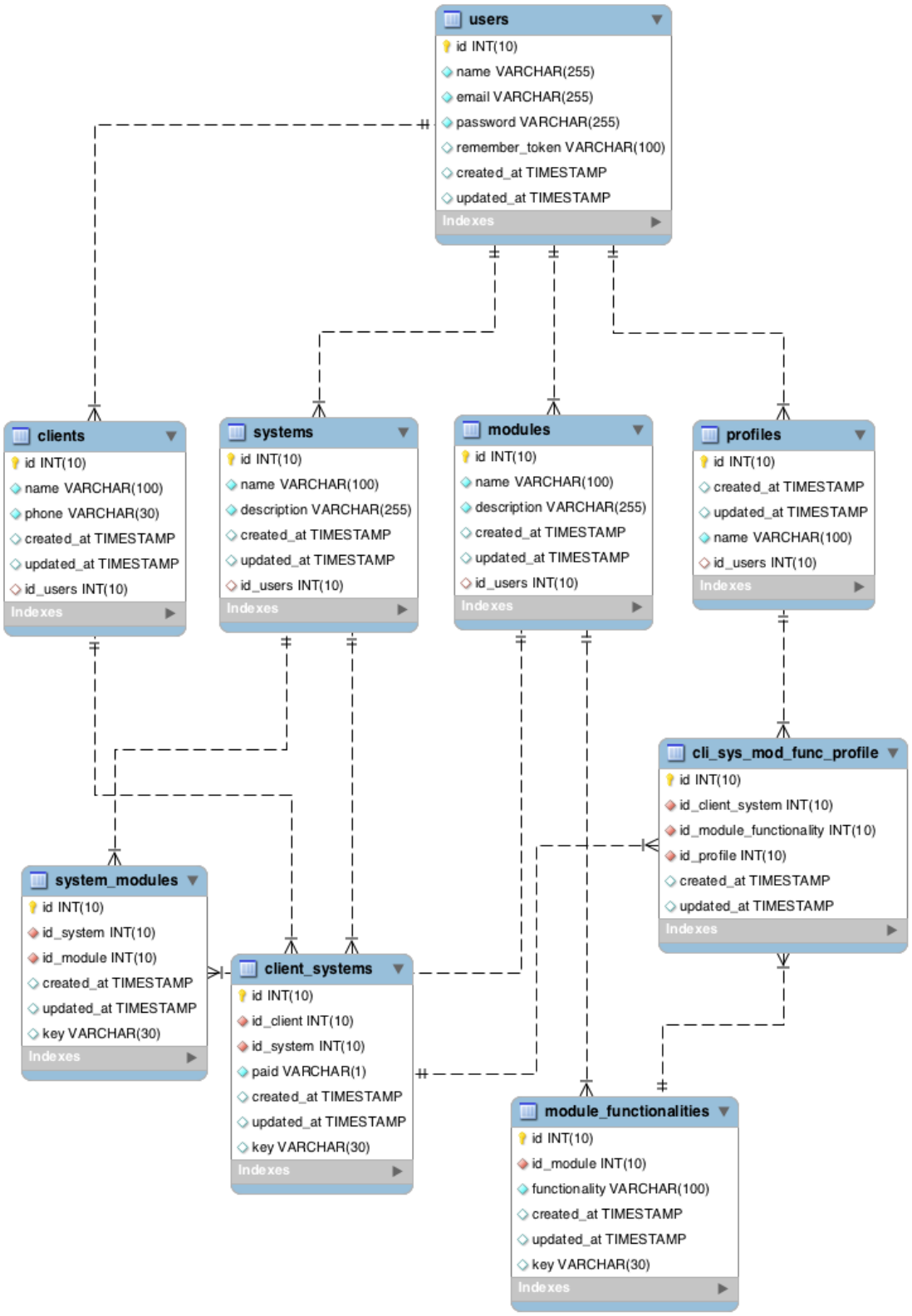

Figura 3.3 Diagrama entidade relacionamento

3.1.3 Funcionalidades

Esta seção apresenta os requisitos funcionais e não-funcionais implementados na aplicação proposta neste trabalho.

Requisitos não funcionais

- RN01: Usuários Simultâneos

  Descrição: O sistema deverá suportar processamento multiusuário, ou seja, vários usuários poderão utilizar o sistema simultaneamente.

- RN02: Segurança

  Descrição: O sistema só permitirá acesso aos dados com autorização. Os usuários deverão se identificar usando um login e uma senha, e a referida senha será criptografada com a metodologia Bcrypt no banco de dados.

- RN03: Alta disponibilidade

  Descrição: O sistema deverá ter disponibilidade de aproximadamente 99 porcento do tempo, a ser verificado com os logs do servidor.

- RN04: Desempenho

  Descrição: O tempo de resposta das consultas não deve ultrapassar 3 segundos, a ser validado com teste ao em produção.

Requisitos funcionais

- RF 01: Login

  Descrição: O sistema deve conter tela de login com os campos email e senha. Após inserção dos dados, o sistema deve validar os dados e caso positivo, encaminhar o usuário ao sistema. Caso os dados sejam inválidos, exibir mensagem de erro para o usuário.

- RF 02: Edição de usuário

  Descrição: O sistema deve conter um formulário que seja carregado com os dados do usuário da sessão e permitir a atualização dos dados. Este requisito está relacionado ao UC01 da imagem 3.1.2.

- RF 03: Cadastro de cliente

  Descrição: O sistema deve conter um formulário para cadastro de clientes. Após a inserção de um registro, o sistema deve automaticamente exibir o registro em formato de edição e conter botão para ir a uma consulta que liste os clientes cadastrados para o usuário da sessão. Este requisito está relacionado ao UC02 da imagem 3.1.2.

- RF 04: Cadastro de sistema

  Descrição: O sistema deve conter um formulário para cadastro de sistemas. Após a inserção de um registro, o sistema deve automaticamente exibir o registro em formato de edição e conter botão para ir a uma consulta que liste os sistemas cadastrados para o usuário da sessão. Este requisito está relacionado ao UC03 da imagem 3.1.2.

- RF 05: Cadastro de módulo

  Descrição: O sistema deve conter um formulário para cadastro de módulos. Após a inserção de um registro, o sistema deve automaticamente exibir o registro em formato de edição e conter botão para ir a uma consulta que liste os módulos cadastrados para o usuário da sessão. Este requisito está relacionado ao UC04 da imagem 3.1.2.

- RF 06: Cadastro de perfil

  Descrição: O sistema deve conter um formulário para cadastro de perfil. Após a inserção de um registro, o sistema deve automaticamente exibir o registro em formato de edição e conter botão para ir a uma consulta que liste os perfils cadastrados para o usuário da sessão. Este requisito está relacionado ao UC05 da imagem 3.1.2.

- RF 07: Cadastro associativo de cliente aos sistemas

  Descrição: O sistema deve conter um formulário para cadastro associativo de clientes a sistemas. Após a inserção de um registro, o sistema deve automaticamente gerar uma chave de acesso do dado registro para o usuário consultar permissões e exibir o registro em formato de edição e conter botão para ir a uma consulta que liste os registros associativos de clientes aos sistemas cadastrados para o usuário da sessão. Este requisito está relacionado ao UC09 da imagem 3.1.2.

- RF 08: Cadastro associativo de sistemas aos módulos

  Descrição: O sistema deve conter um formulário para cadastro associativo de sistemas a módulos. Após a inserção de um registro, o sistema deve automaticamente gerar uma chave de acesso do dado registro para o usuário consultar permissões e exibir o registro em formato de edição e conter botão para ir a uma consulta que liste os registros associativos de sistemas aos módulos cadastrados para o usuário da sessão. Este requisito está relacionado ao UC08 da imagem 3.1.2.

- RF 09: Cadastro de funcionalidade dos módulos

  Descrição: O sistema deve conter um formulário para cadastro de funcionalidade dos módulos já cadastrados. Após a inserção de um registro, o sistema deve automaticamente gerar uma chave de acesso do dado registro para o usuário consultar permissões e exibir o registro em formato de edição e conter botão para ir a uma consulta que liste os registros de funcionalidades cadastradas para o usuário da sessão. Este requisito está relacionado ao UC06 da imagem 3.1.2.

- RF 10: Composição de permissão para os módulos dos clientes

  Descrição: O sistema deve conter um formulário para cadastro associativo de permissão para os módulos dos clientes. Após a inserção de um registro, o sistema deve automaticamente exibir o registro em formato de edição e conter botão para ir a uma consulta que liste os registros associativos de permissão cadastrados para o usuário da sessão. Este requisito está relacionado ao UC07 da imagem 3.1.2.

- RF 11: Endpoint

  Descrição: O sistema deve disponibilizar uma URL para receber uma requisição via post, onde o deve receber um parâmetro "key", contendo a chave de acesso desejada para consultar as permissões cadastradas. Ao realizar a requisiçao enviando a "key", o sistema deve retornar um texto no formato Json, contendo os dados das permissões que foram compostas no sistema pelo usuário. A chave enviada pelo usuário pode ser de qualquer um dos níveis de configuração realizado no sistema e o o endpoint deve retornar a resposta sempre num mesmo formato, viabilizando um tratamento único da resposta pelo cliente. Este requisito está relacionado ao UC10 da imagem 3.1.2.

3.2 Processo

Para melhor compreensão do sistema proposto, a figura 3.2 apresenta uma visão geral do processo implementado na ferramenta, utilizando, para tal, a notação BPMN.

3.3 O que há de novo ?

Seguindo a norma de padranização de código, com o software proposto é possível unificar o módulo de permissões de acesso e tratá-lo como um serviço, tornando possível eliminá-lo de qualquer software que faça uso do mesmo. A bordagem proposta neste trabalho de conclusão não é trivial, trata-se de uma aplicabilidade diferente do controle de permissão dos perfis, que não foi adotado por nenhum sistema disponível no estado da prática.

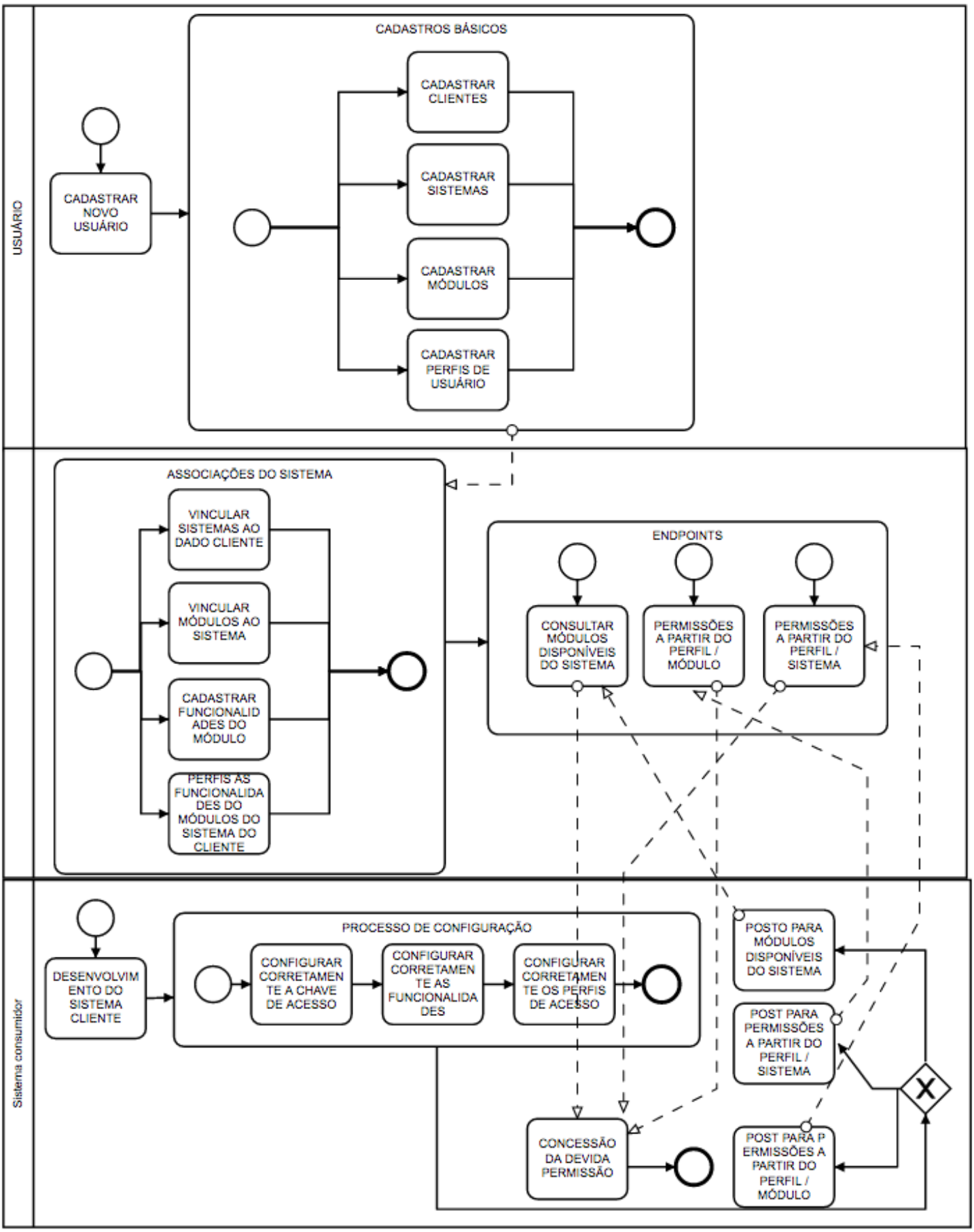

Figura 3.4 Diagrama do processo(BPMN) do software em questão

4

Análise dos dados

Como não foi encontrado no estado da prática, uma aplicação que contenha as mesmas características do trabalho proposto, neste capítulo tem-se a análise de uma pesquisa acerca da motivação das pessoas perante a proposta deste trabalho. Assim, a seção 4.1 traz o objetivo da pesquisa realizada. A seção 4.2 busca definir o nicho correto de pessoas a quem a pesquisa deve ser direcionada. A seção 4.3 contém uma apresentação da pesquisa aplicada e a seção 4.4, os resultados provenientes da pesquisa.

4.1 Objetivo da pesquisa

Como não foi encontrado um software que provesse as mesmas funcionalidades do trabalho proposto, então tem-se o mesmo como um novo produto no mercado. Entretanto, antes de expôr um produto, é recomendado que realize uma análise de mercado para saber a viabilidade da ideia. Baseado nisso, o objetivo da pesquisa deste trabalho é buscar saber se a ideia tem uma boa aceitação no mercado.

4.2 Público alvo

Diante do fato de o trabalho proposto estar ligado à área de desenvolvimento de software, independentemente da tecnologia usada para trabalhar, será adotado como público alvo da pesquisa as pessoas ligadas ao desenvolvimento e manutenção de software, podendo ser programador, analista, engenheiro de software, estudantes, pesquisadores ou áreas afins.

4.3 Método de pesquisa

A fim de validar a proposta deste trabalho de conclusão de curso, trabalhará-se com dados primários, ou seja, dados coletados e analisados através de um formulário de pesquisa. Tal formulário é composto por dez perguntas, cada uma contendo quatro opções de resposta.

4.4 Resultados

5
Conclusão

Referências Bibliográficas

[1] RedMonk, "Gráfico de linguanges - redmonk." http://goo.gl/vDSGLx, 2016.

[2] Google, "Googletrends - frameworks front-ent mais buscados em 2015." https://goo.gl/VUhIrv, 2016.

[3] H. La and S. Kim, "A Systematic Process for Developing High Quality SaaS Cloud Services," in *Cloud Computing* (M. Jaatun, G. Zhao, and C. Rong, eds.), vol. 5931 of *Lecture Notes in Computer Science*, ch. 25, pp. 278–289, Berlin, Heidelberg: Springer Berlin / Heidelberg, 2009.

[4] M. Lechesa, L. F. Seymour, and J. Schuler, "Erp software as service (saas): Factors affecting adoption in south africa.," in *CONFENIS* (C. Møller and S. S. Chaudhry, eds.), vol. 105 of *Lecture Notes in Business Information Processing*, pp. 152–167, Springer, 2011.

[5] R. Rai, G. Sahoo, and S. Mehfuz, "Securing software as a service model of cloud computing: Issues and solutions.," *CoRR*, vol. abs/1309.2426, 2013.

[6] H. Chen, "Architecture strategies and data models of software as a service: A review," in *2016 3rd International Conference on Informative and Cybernetics for Computational Social Systems (ICCSS)*, pp. 382–385, Aug 2016.

[7] T. Kaur, D. Kaur, and A. Aggarwal, "Cost model for software as a service," in *2014 5th International Conference - Confluence The Next Generation Information Technology Summit (Confluence)*, pp. 736–741, Sept 2014.

[8] M. Ezran, M. Morisio, and C. Tully, *Practical Software Reuse*. London, UK, UK: Springer-Verlag, 2002.

[9] V. R. Basili and H. D. Rombach, "Support for comprehensive reuse," *Softw. Eng. J.*, vol. 6, pp. 303–316, Sept. 1991.

[10] R. Keswani, S. Joshi, and A. Jatain, "Software reuse in practice," in *2014 Fourth International Conference on Advanced Computing Communication Technologies*, pp. 159–162, Feb 2014.

[11] E. S. Almeida, A. Alvaro, V. C. Garcia, J. C. C. P. Mascena, V. A. d. A. Burégio, L. M. Nascimento, D. Lucrédio, and S. L. Meira, *C.R.U.I.S.E - Component Reuse in Software Engineering*. CESAR, 2007.

[12] D. Wickramaarachchi and R. Lai, "Software modularization in global software development," in *2014 International Conference on Data and Software Engineering (ICODSE)*, pp. 1–6, Nov 2014.

[13] S. R. Kumar, R. Sharma, and K. Gupta, "Strategies for web application development methodologies," in *2016 International Conference on Computing, Communication and Automation (ICCCA)*, pp. 160–165, April 2016.

[14] Y. Ping, K. Kontogiannis, and T. C. Lau, "Transforming legacy web applications to the mvc architecture," in *Software Technology and Engineering Practice, 2003. Eleventh Annual International Workshop on*, pp. 133–142, Sept 2003.

[15] J. Yuping, "Research and application of ajax technology in web development," in *2014 IEEE Workshop on Electronics, Computer and Applications*, pp. 256–260, May 2014.

[16] J. Tesarik, L. Dolezal, and C. Kollmann, "User interface design practices in simple single page web applications," in *2008 First International Conference on the Applications of Digital Information and Web Technologies (ICADIWT)*, pp. 223–228, Aug 2008.

[17] I. Salvadori and F. Siqueira, "A maturity model for semantic restful web apis," in *2015 IEEE International Conference on Web Services*, pp. 703–710, June 2015.

Apêndice