“ModMan - Gerenciamento de Permissões para SaaS”

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

Victor de Azevedo Nunes

"ModMan - Gerenciamento de Permissões para SaaS"

Trabalho apresentado ao Departamento de Ciência da Computação da Universidade Federal da Bahia como requisito parcial para obtenção do grau de Bacharel em Sistemas de Informação.

Orientador: *Ivan do Carmo Machado*

SALVADOR, Abril/2017

Dedico este trabalho aos meus pais, Adelmo e Telma,
irmãos Vivian e Vinicius e à minha namorada Laís, pela
expectativa da realização deste marco.

Agradecimentos

Agradeço primeiramente a Deus, por ter me dado força para seguir em frente e vencer as diversas dificuldades encontradas no caminho.

Agradeço os meus pais, Adelmo e Telma, pelo amor, dedicação na formação proporcionada que me fez chegar até aqui e compreensão pelos momentos em que estive ausente devido à dedicação aos estudos. Aos meus irmãos, que também contribuíram à minha formação.

À minha namorada Laís, pelo companheirismo, incentivo, compreensão em todos os momentos e por acreditar na minha evolução.

Ao amigo Guilherme pela grande contribuição, me ajudando com muita dedicação o que se fez necessário, sem medir esforços. Também ao amigo Vinicius, que sempre me incentivou e não mediu esforços para me ajudar quando precisei.

E à Universidade, pelos ensinamentos e por proporcionar amigos que levarei comigo.

"Ninguém nunca conseguiu alcançar sucesso simplesmente fazendo o que lhe é solicitado. É a quantidade e a excelência do que está além do solicitado que determina a grandeza da distinção final."

—CHARLES KENDALL ADAMS

Resumo

Este trabalho de conclusão de curso utiliza elementos da engenharia de software para propor um software como serviço a fim de otimizar o processo de construção e manutenção dos softwares. Assim, o objetivo deste SaaS é gerenciar as permissões de acesso de sistemas cliente, provendo o reuso de software e padronizando as soluções. O sistema proposto neste trabalho encontra-se implementado e disponível no Github, e traz fundamentos sobre a arquitetura e tecnologias utilizadas, bem como avaliações sobre possibilidades de utilização do mesmo em diversos ambientes, como Web e mobile.

Palavras-chave: Software; Reuso; SaaS; Web; PHP

Abstract

This finals period uses elements of software engineering to propose software as a service to optimize the process of building and maintaining the software. Thus, the goal of this project is to generate access permissions to client systems, providing the reuse of software and standardizing as solutions. The system proposed in this work is implemented and available at Github, bringing fundamentals about the architecture and technologies used, as well as evaluations on possibilities of using it in various environments, such as Web and mobile.

Keywords: Software; Reuse; SaaS; Web; PHP

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Acrônimos

SPA	Single Page Application
JSON	Javascript Object Notation
PHP	PHP: Hypertext Preprocessor
SaaS	Software as a Service
ERP	Enterprise Resource Planning
QoS	Quality of Service
UML	Unified Modeling Language
MVC	Model-View-Controller
Ajax	Asynchronous Javascript and XML
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
API	Application Programming Interface
DOM	Document Object Model
BPMN	Business Process Model and Notation
REST	Representational State Transfer

1 Introdução

O desenvolvimento de software usualmente enfrenta problemas relacionados ao não cumprimento dos prazos de entrega. Tal situação pode ocorrer devido ao aumento de demandas não planejadas no início do projeto, desvios na interpretação e/ou implementação dos requisitos, bem como problemas internos na equipe. Assim, a solução para atrasos de entrega pode estar em soluções que resolvam problemas específicos. Vale ressaltar que a literatura apresenta estratégias que podem mitigar ou combater esse problema, como por exemplo, o reuso de software. Este trabalho de conclusão de curso propõe uma solução de padronização do controle de acesso dos softwares, abstraindo-o e tornando-o passível de maior reuso como um serviço reutilizável no desenvolvimento de software, reduzindo assim o tempo de desenvolvimento e manutenção de aplicações de software. O capítulo corrente é composto por sete seções. A Seção 1.1 aborda a motivação do trabalho, a Seção 1.2 o problema do contexto, a Seção 1.3 mostra os objetivos almejados, a Seção 1.4 comenta sobre a metodologia aplicada, a Seção 1.5 demonstra os resultados esperados com a aplicabilidade do trabalho proposto, a Seção 1.6 discorre sobre elementos fora do escopo do projeto e por fim, a Seção 1.7 exibe a estrutura da monografia.

1.1 Motivação

Otimizar o desenvolvimento de software é uma tarefa bastante estudada, com uma vasta literatura. 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, bem como a redução no tempo de entrega dos projetos. Assim, é possível que melhorias no processo de desenvolvimento de sistemas, podem satisfazer desde as necessidades do programador, com um trabalho facilitado, até às necessidades do cliente final, com a redução de custo.

1.2 Problema

A reutilização não-sistemática de trechos de código é um problema que pode acarretar no aumento da complexidade do software, bem como no tempo de desenvolvimento e manutenção. Este problema atinge fortemente a manutenibilidade do software, o que pode gerar grande transtorno diante de uma alteração que poderia ser simples caso o projeto fosse concebido considerando o potencial de reuso. Deste modo, a reutilização sistemática de código é uma prática bem-vinda em projetos de software. A abordagem de reuso pode inclusive, abranger mais de um projeto numa organização, possibilitando que módulos inteiros sejam reaproveitados.

1.3 Objetivos

- **Objetivo geral**: O objetivo geral do ModMan é prover o reuso de código, utilizando padões que mantenham o componente de código na melhor forma possível para ser acoplado em outros softwares de modo facilitado.

- **Objetivo específico**: Neste trabalho, buscamos implementar um mecanismo de reutilização de módulo de controle de acesso de um sistema, já que o mesmo está presente sem remodelações nos sistemas de uma organização. Assim, o objetivo foi 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 controle de acesso, o qual seria um projeto à parte. Também apresenta a vantagem de ter interoperabilidade entre linguagens de programação, já que realiza a comunicação através de JSON (JavaScript Object Notation), seguindo padrões atuais. Assim, tem-se uma constância no módulo de acesso independentemente da possível necessidade de migração de tecnologia.

1.4 Metodologia

Esta seção apresenta as tecnologias utilizadas na implementação do ModMan com base na indústria de desenvolvimento de software (estado da prática). Para embasar esse cenário, a seguir apresentamos dados que refletem a atual situação do mercado, a nível global.

Baseando-se nas tecnologias livres comummente empregadas no cenário atual do desenvolvimento Web, há diversas opções eficientes para a implementação da solução. Dentre as

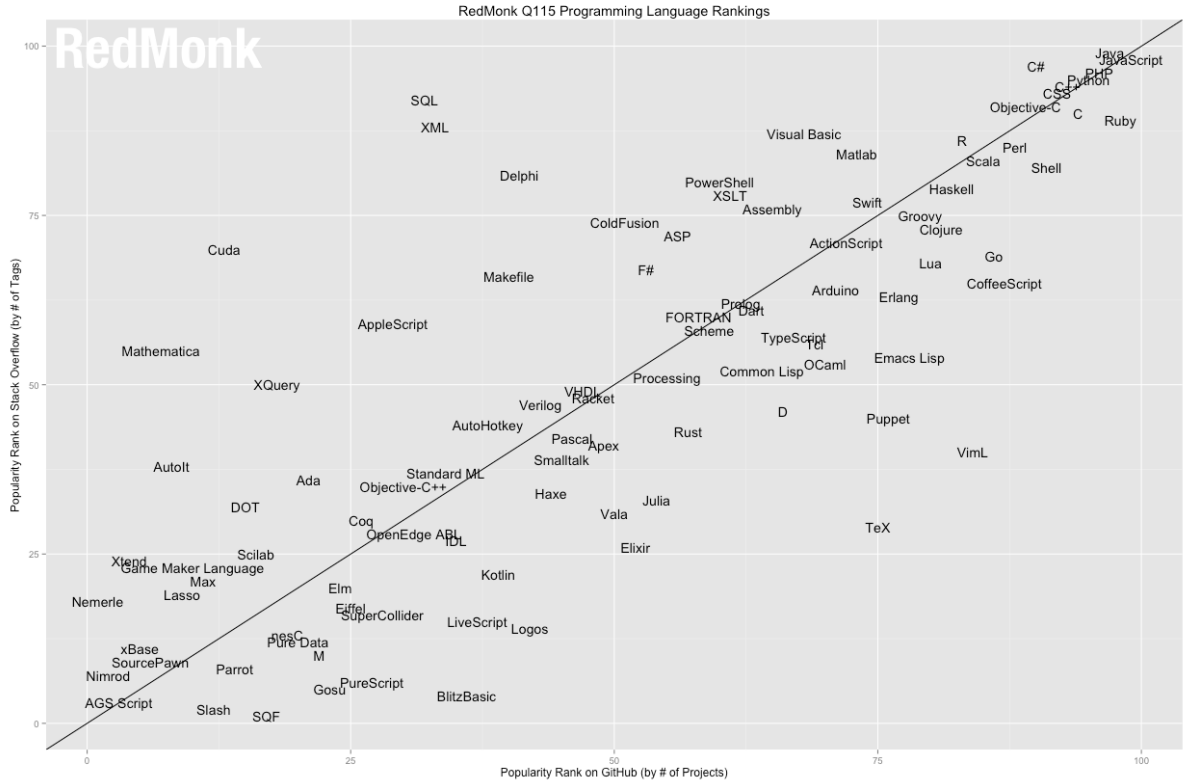

Figura 1.1 Ranking das liguagens de programação no Stack Overflow e Github

possibilidades, considerando a facilidade para futura manutenção e continuidade do projeto, tende-se a optar por uma tecnologia popular. Como linguagem de programação, a proposta deste trabalho foi desenvolvida utilizando o PHP (PHP: Hypertext Preprocessor). A escolha é fundamentada na pesquisa da RedMonk de 2015 [11], que evidencia o uso das linguagens de programação de acordo com as discussões no StackOverflow[1] e repositórios no GitHub[2]. A Figura 1.1 apresenta a popularidade das liguagens de programação, na qual o PHP é apresentado na terceira colocação, apenas atrás do líder, JavaScript, e do segundo colocado, Java.

Adicionalmente, dentre os frameworks disponíveis para PHP, de acordo com a Figura 1.2, hoje o destaque está com o Laravel[3], que se encontra como o mais utilizado no momento.

A WebHostFace[4], 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 infográfico apresentado

[1] http://pt.stackoverflow.com/
[2] github.com
[3] https://laravel.com/
[4] https://www.webhostface.com/

na figura 1.2.

De acordo com a Figura 1.2, o Laravel em 2015 teve a maior popularidade em projetos pessoais e tem a maior comunidade entre os concorrentes. Além disso, em comparação aos concorrentes, tem alto nível de segurança e velocidade de resposta com nível normal. Tais características analisadas na Figura 1.1, tornam o Laravel uma boa escolha para a escrita de um software que possivelmente será mantido por terceiros.

Para elaborar os recursos de interface e integrar ao back-end PHP do sistema, será adotado o AngularJS[5], ferramenta sólida e conhecida no aspecto em questão.

Dados coletados via Google Trends, que propõem comparações entre termos pesquisados, revelam a popularidade do AngularJS diante de alguns dos principais concorrentes. O gráfico 1.3 evidencia o cenário, mostrando que o AngularJs lidera como termo pesquisado dentre framwworks relacionados.

1.5 Resultados Esperados

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

- Servir a projetos de diferentes linguagens;
- Atender a projetos de diferentes plataformas (Web, Desktop, Mobile);
- Facilitar a configuração das permissões;
- Extinguir o módulo de permissões nos projetos em que o trabalho proposto será aplicado.

1.6 Fora de Escopo

As permissões de acesso podem estar disponíveis mediante diversas configurações. Por exemplo, pode ser condicional ao pagamento da mensalidade. Essas características que tendem a levar o ModMan para se assemelhar a um ERP (Enterprise Resource Planning), não estão no escopo da implementação apesar de ser possível que o ModMan adquira essas caracterísitcas mediante evoluções no sistema. Atualmente, o ModMan é um sistema limitado, que restringe a sua atuação somente às permissões de acesso numa forma simples.

[5]https://angularjs.org/

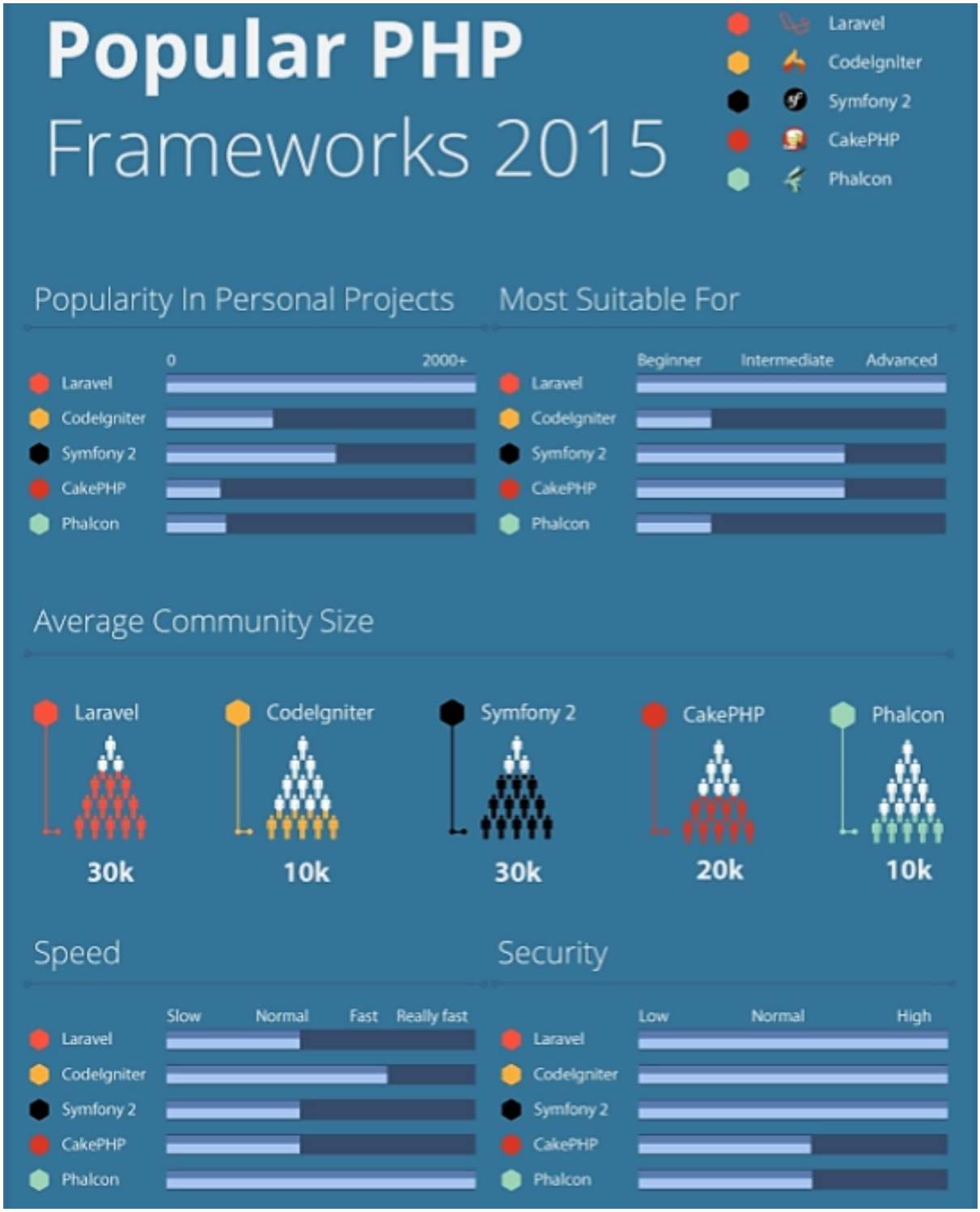

Figura 1.2 Infográfico da WebhostFace, exibindo a popularidade dos Frameworks PHP em 2015

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

1.7 Estrutura do Trabalho

Nesta seção, tem-se uma breve apresentação dos demais capítulos que compõem este trabalho de conclusão de curso:

- Capítulo 2: *Referencial teórico*, fundamentando os principais conceitos que compõem os elementos principais do trabalho;
- Capítulo 3: *Proposta*, contendo o detalhamento do trabalho proposto, baseando-se nos aterfatos da engenharia de software;
- Capítulo 4: *Avaliação empírica*, apresentando a prova de conceito do trabalho;
- Capítulo 5: *Conclusão*, apresentando o desfecho do trabalho.

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 é essencial na concepção de um produto de software. De todo o modo, sempre busca-se a premissa de fazer mais com meno. Assim, 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 cinco seções. A Seção 2.1 trata de Software como serviço, discutindo alguns elementos do contexto que são relevantes para o trabalho proposto. A Seção 2.2 discute sobre a empregabilidade do reuso de software. A Seção 2.3 trata dos aspectos envolvidos na modularização dos softwares. A Seção 2.4 aborda as aplicações web, tratando sobre aspectos relevantes acerca da aplicação web que compõe este trabalho e, por fim, a Seção 3.3 sintetiza os tema abordados nas seções do capítulo corrente, discorrendo sobre a relevância do estudo dos mesmos.

2.1 Software como serviço

Segundo La e Chun [8], 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*). Embora interligadas, no entanto, as áreas não se confundem, uma vez que um SaaS deve ser entendido como um mecanismo de suporte às soluções existentes na cloud. Os SaaS existem justamente para maximizar o reuso de serviços comuns em aplicações.

Como vantagens, SaaS representa uma forte tendência no desenvolvimento de soluções Web, graças às tecnologias emergentes que a potencializam, a exemplo de requisições AJAX (Asynchronous JavaScript And XML). Existem diversos fatores que podem ser favoráveis à adoção de um SaaS, como custo e manutenção. Uma investigação recente [9], apresenta o resultado de uma pesquisa que quantifica 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 foram sua fluidez quanto à rede e a segurança. Esses fatores estão presentes na aplicação desenvolvida neste trabalho de conclusão de curso, pois são essenciais para o funcionamento do ModMan.

Devido ao fato de ter um serviço constantemente na nuvem, é um fator impoartante a segurança da informação manipulada. Sabe-se que a vulnerabilidade na Web não é restrita ao SaaS, atingindo diversos âmbitos. Rai et al. [10] consideram que o avanço da computação em nuvem não é um problema apenas para os serviços Web do ponto de vista da segurança, pois trabalhos na literatura mostram a área como mais um ponto de vulnerabilidade para diversos setores, a exemplo de infraestrutura. Rai et al. [10], também reportam resultados de estudos exploratórios realizados junto a empresas usuárias de serviços em computação em nuvem. Segundo os autores, a perspectiva de Saas também pode fortalecer a segurança nas aplicações de cloud computing. Isso ocorre 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.

A arquitetura de armazenamento de dados de um Saas pode variar de acordo com a necessidade do contexto. Huixin [3] ilustra 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 multiclientes, devido à maior facilidade para manutenção.

Devido à forma de obtenção de software (produto), 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. [5] 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, 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. E caso fosse necessário determinar o custo para um cliente, os elementos aplicados no trabalho de T. Kaur et al. [5], que foram comentados nessa seção, poderiam ser levados em consideração.

2.2 Reuso de software

De acordo com o livro Pratical Software Reuse [4], 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 [2] 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. [6], 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, conforme apresenta o C.R.U.I.S.E Book [1] :

- **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 economia nas fases de análise e design, reduzindo o custo total do projeto.
- **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.
- **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 menor tempo de desenvolvimento e custo associado.

- **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 encontrar 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.

O C.R.U.I.S.E Book [1] também destaca alguns aspectos que podem gerar conflito com a cultura de reuso de software:

- **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 reutulizáveis. 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 fácil acesso ao software existente não necessariamente aumenta 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.

Segundo Keswani et al. [6], para o reuso de software dar retornos apropriados, o processo deve ser sistemático e planejado. Qualquer organização que implemente a reutilização de software deve identificar os melhores métodos e estratégias de reutilização para obter a máxima produtividade.

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

2.3 Modularização

Conforme destacado por Wickramaarachchi e Lai [14], 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 de software foram introduzidos por pesquisadores pioneiros há quarenta anos, com notável contribuição feita por Melvin Conway e David Parnas, que tem representação notável na engenharia de software.

Segundo Wickramaarachchi e Lai [14], a modularização é importante na identificação de dependências e reduz as dificuldades diante de uma possível necessidade de grandes alterações. Na perspectiva da engenharia de software, 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 e a qualidade do software. Tal conceito encontra-se bastante difundido e está incorporado em linguagens de programação e ferramentas de software.

O presente trabalho favorece o uso da modularização de 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. [7] sugerem que para o desenvolvimento Web, deve-se manter a prática eficaz de produzir artefatos de engenharia de software. 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.

O Ajax (Asynchronous Javascript and XML) foi essencial para a evolução da web, sendo uma tecnologia indispensável para o advento da web 2.0. 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.

De acordo com Tesarik et al. [13], a arquitetura de software SPA (Single page application) é uma forma 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. 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 é interessante que na elaboração dos artefatos visuais, sejam explorados os mais ricos recursos na implementação, como HTML5, JavaScript, Ajax, CSS3, e outras tecnologias que se apliquem. Entretanto, desenvolver uma interface rica com o uso de diversas tecnologias/frameworks pode ocasionar um esforço maior para explorar as possibilidades. Os 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.

Segundo Salvadori e Siqueire [12], REST é uma arquitetura muito popular para integração de aplicativos Web, e permite compartilhar e reutilizar informações através de sistemas. Aplicações de grande escala baseadas em REST devem 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 integrada 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.

2.4.1 Tecnologias

Dentro do contexto das aplicações Web, esta subseção evidencia as tecnologias que foram aplicadas ao desenvolvimento do ModMan.

Laravel framework

O Laravel 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 uma grande comunidade, o que facilita a aprendizgem, pois facilmente encontram-se tutoriais. Inclusive, além da documentação, o próprio Laravel disponibiliza o Laracasts [1], uma série de vídeos ensinando as mais diversas funcionalidades encontradas no Laravel.

AngularJs

AngularJS é um framework JavaScript open-source, mantido pelo 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, além de fazer 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 framework front-end popular que facilita a criação de páginas Web com tecnologia responsiva, que se adapta corretamente às diferentes resoluções de tela. O Bootstrap possui diversos componentes (plugins) em JavaScript (jQuery) que auxiliam o desenvolvedor na implementação, como por exemplo: menu-dropdown, modal, carousel, slideshow, entre outros com facilidade, apenas acrescentando algumas configurações no código.

2.5 Resumo do capítulo

Este capítulo apresentou a abordagem teórica voltada para temas relacionados ao Modman. Isso é evidenciado pela busca na literatura por trabalhos que abordam os principais aspectos utilizados nesta monografia, como a efetividade do SaaS e possíveis vantagens e desvantagens da utilização do reuso e modularização dos softwares. O embasamento de referências literárias fornece maior segurança sobre a escolha dos elementos que compõem o trabalho. O mesmo embasamento também é apresentado em relação aos aspectos técnicos, discutindo trabalhos que analisam o comportamento de tecnologias atualmente empregadas no desenvolvimento de páginas Web e que foram adotadas para o desenvolvimento do Modman.

[1] https://laracasts.com/

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 do ModMan 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.

Assim, este capítulo é composto por três seções. A Seção 3.1 trata da arquitetura do ModMan, contendo informações sobre as tecnologias utilizadas, diagramas UML e os requisitos do sistema. A Seção 3.2 discorre acerca das funcionalidades do ModMan, descrevendo os requisitos funcionais e não funcionais e por fim, a Seção 3.3 traz um breve resumo sobre o presente capítulo.

3.1 Arquitetura do ModMan

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 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 então potencializa as ações em *client-side*. Assim, boa parte da aplicação passa a ser processada no cliente (dentro do navegador Web).

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 a partilha do processamento, tem-se uma menor codificação no servidor. Tal aspecto vem acompanhado da divisão 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 têm quantidade relevante, é necessário atenção 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 uma aplicação 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 Javascript 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

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

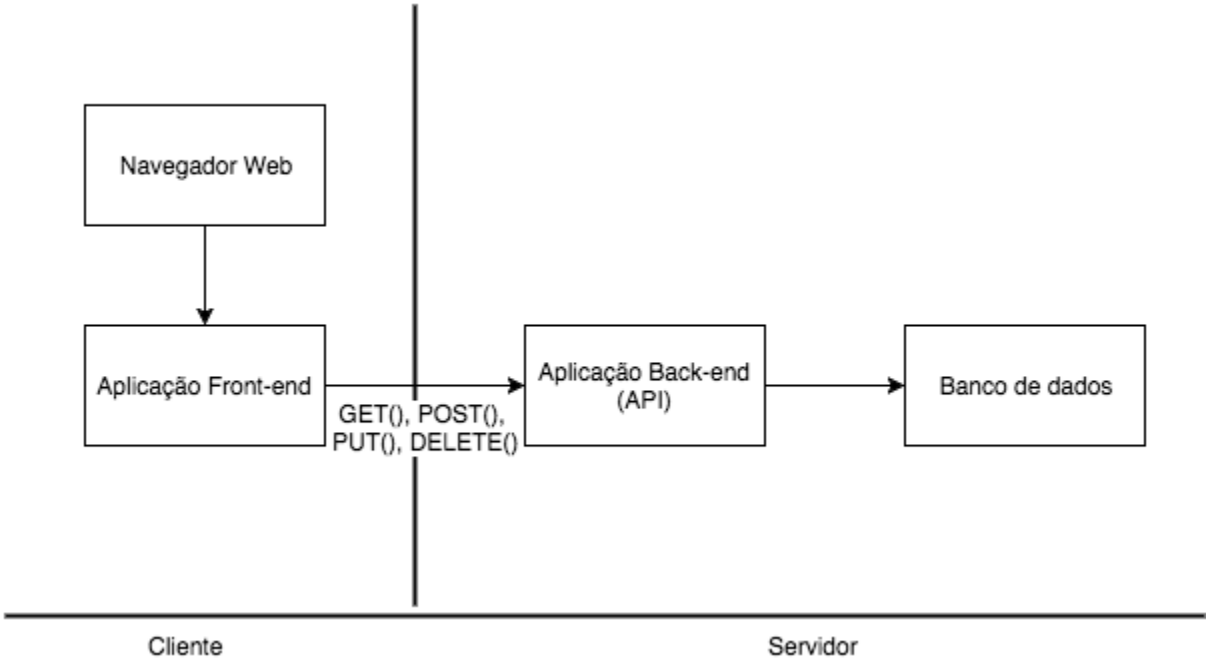

Figura 3.1 Arquitetura

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, é chamada de RESTFul. Neste trabalho, foi desenvolvida uma aplicação RESTFul com o Framework PHP Laravel.

A Figura 3.1 exibe de forma genérica a organização das camadas do software desenvolvido, demonstrando o fluxo. A caixa "Aplicação Front-end"é onde a se encaixa a aplicação SPA, e na caixa "Aplicação Back-end (API)"é onde funciona a aplicação RESTFul, desenvolvida com o 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.

De acordo com a arquitetura apresentada, o código do software encontra-se no Github[6], dividido em dois repositórios: Um voltado exclusivamente para o back-end da aplicação,

[6]https://github.com/

disponível no repositório ModMan[7] e um para o front-end, disponível no repositório ModMan-Front[8]. E em funcionamento, o software encontra-se no site do ModMan[9].

A seguir, a subseção 3.1.1 apresenta os diagramas UML do projeto e a Seção **??** trata das funcionalidades, formalizando os requisitos funcionais e não funcionais.

3.1.1 Diagramas UML

Os diagramas UML(Unified Modeling Language) apresentados a seguir tem a finalidade de embasar o ModMan, permitindo representar o sistema de forma padronizada.

A Figura 3.2 traz o diagrama de casos de uso com o objetivo de prover maior entendimento da forma com que a aplicação foi arquitetada, representando as ações que o usuário pode ter diante do uso do sistema do ModMan.

Com a figura 3.3 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 figura 3.4 traz o diagrama de entidade relacionamento, fornecendo uma ilustração da organização do banco de dados utilizado.

Relacionando os diagramas apresentados com a Figura 3.1, o diagrama de caso de uso presente na Figura 3.2 representa o usuário em ação diante da caixa entitulada "Aplicação Front-end". O diagrama de classes presente na Figura 3.3 está relacionado com a caixa entitulada por "Aplicação Back-end(API)"e o DER, representado na Figura 3.4, na caixa "Banco de dados". Também, os diagramas UML aqui apresentados estão relacionados entre si. O diagrama de classe está intimamente ligado ao MER, normalmente cada classe está relacionada a uma tabela correspondente no MER. Também, temos que os casos de uso quando codificados, se relacionam com as classes correspondentes do diagrama de classe.

3.2 Funcionalidades do ModMan

Esta seção apresenta os requisitos funcionais e não-funcionais que foram escritos para o ModMan. Assim, estes requisitos refletem as funcionalidades implementadas.

Requisitos não funcionais

- RN01: Usuários Simultâneos

[7] https://github.com/victorazv/ModMan
[8] https://github.com/victorazv/Modman-front
[9] http://modman.ga

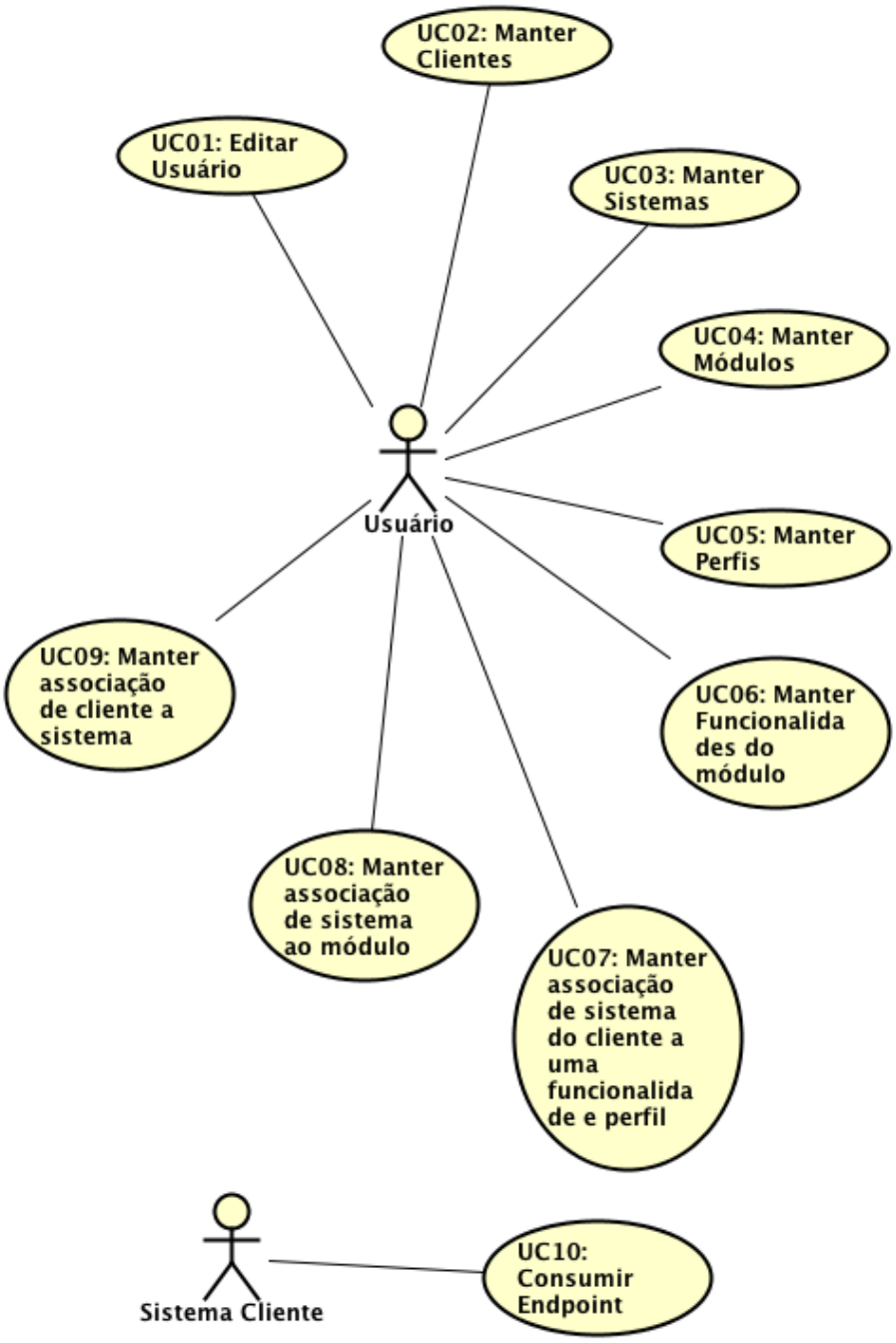

Figura 3.2 Diagrama de caso de uso

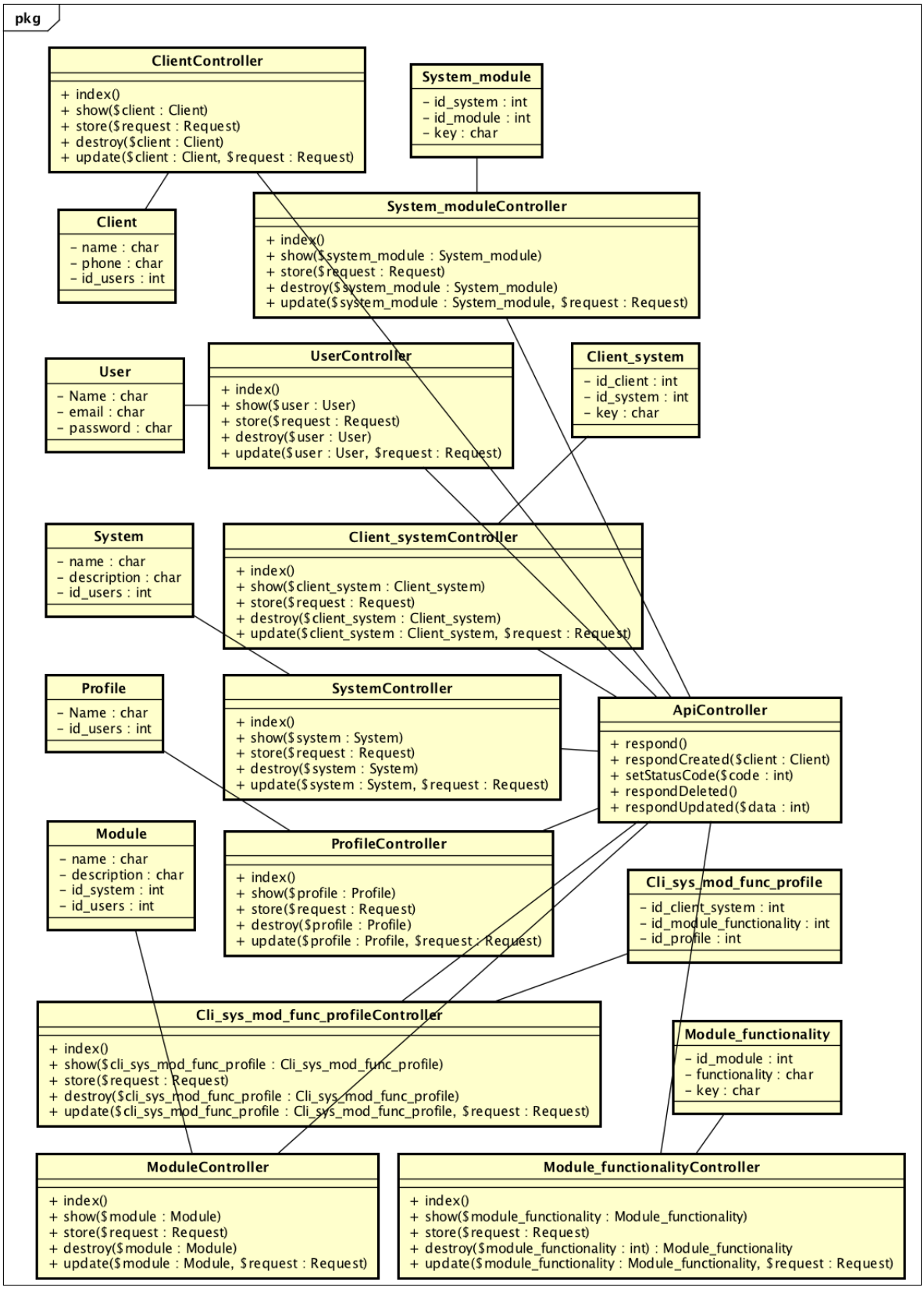

Figura 3.3 Diagrama de classe

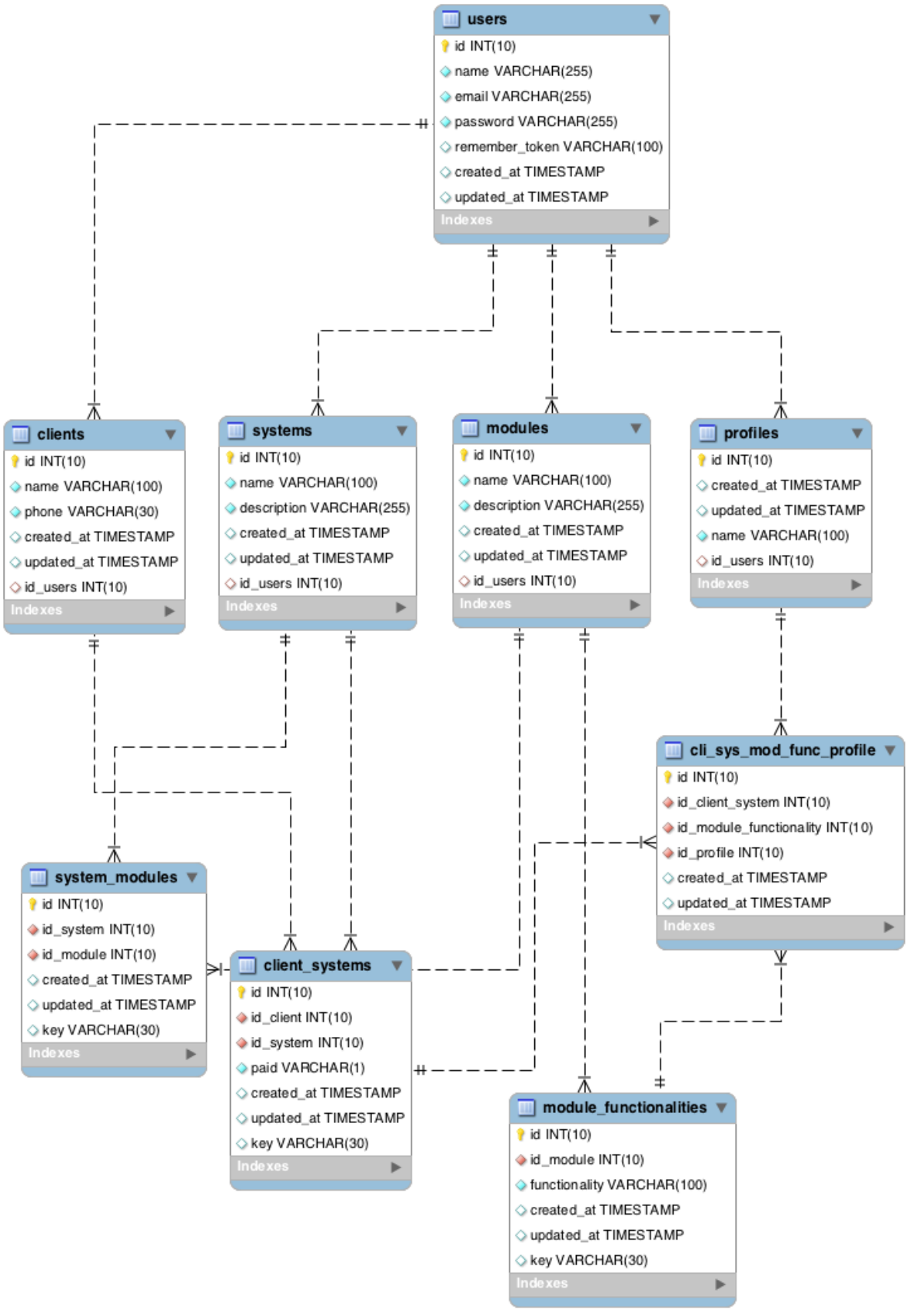

Figura 3.4 Diagrama entidade relacionamento

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.

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 figura 3.2.

- 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 figura 3.2.

- 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 figura 3.2.

- 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 figura 3.2.

- 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 figura 3.2.

- 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 figura 3.2.

- 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 figura 3.2.

- 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 figura 3.2.

- 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 figura 3.2.

- RF 11: Endpoint

  Descrição: O sistema deve disponibilizar a URL http://api.modman.ga/api/endpoint para receber uma requisição via post, que 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 figura 3.2.

Para melhor compreensão do sistema proposto, a Figura 3.5 apresenta uma visão geral do processo implementado na ferramenta, utilizando, para tal, a notação BPMN (Business Process Model and Notation).

3.3 Resumo do capítulo

Seguindo a norma de padronização de código, com o ModMan 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 proposta trata de uma aplicabilidade diferente do controle de permissão dos perfis, que não foi detectada em sistemas disponíveis no estado da prática.

Os artefatos de software aqui apresentados servem para embasar qualquer usuário do ModMan, pois será possível tomar conhecimento sobre a razão da escolha das tecnologias utilizadas, da arquitetura, mapeando vantagens e desvantagens e também tomar conhecimento da proposta de cada funcionalidade do sistema através dos requisitos funcionais. Complementando o entendimento, os diagramas UML fornecem melhor visão de determinados aspectos, a exemplo do diagrama de classe que exibe a organização interna das classes do back-end do software.

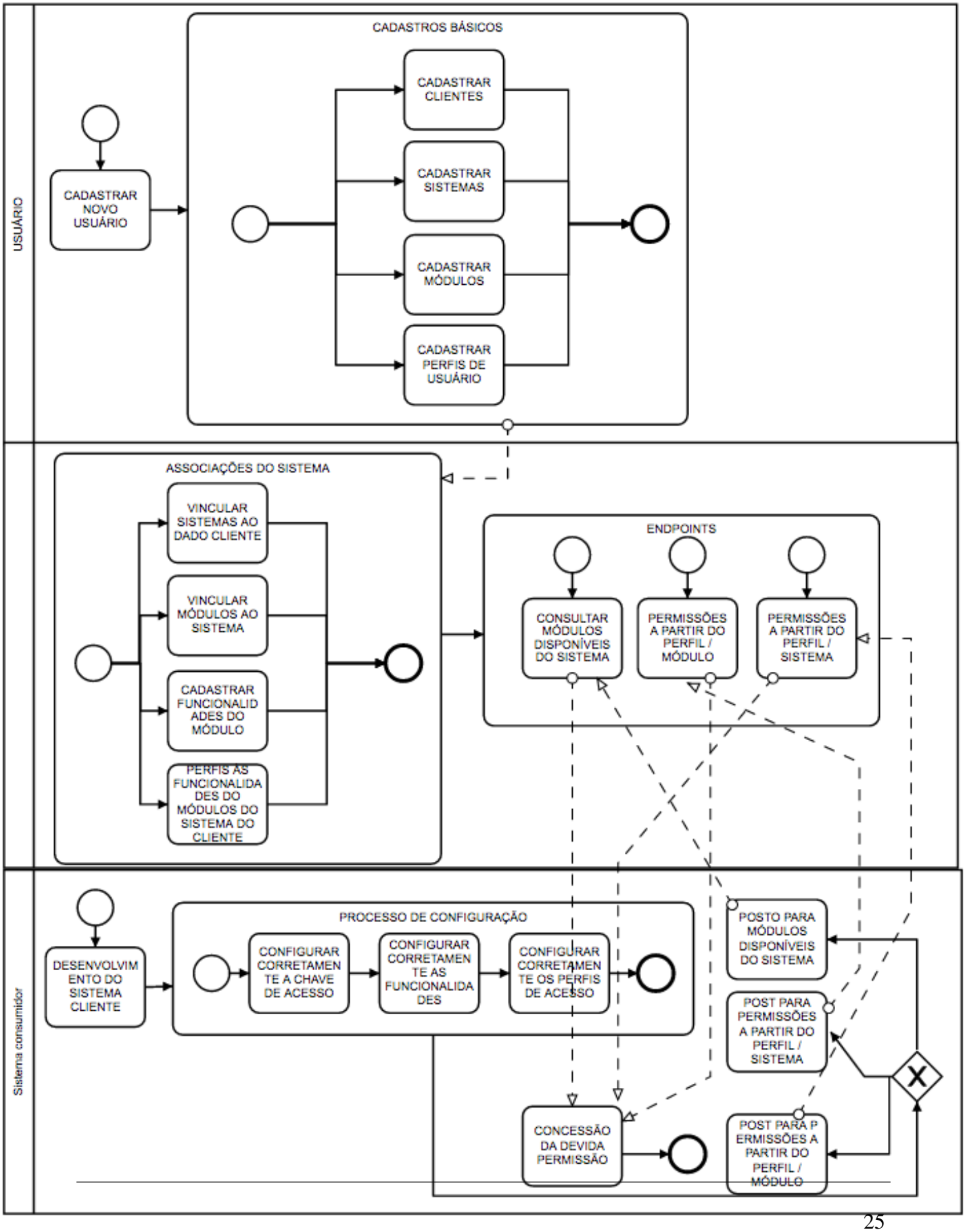

Figura 3.5 Diagrama do processo do ModMan

4

Avaliação empírica

A fim de realizar a validação da ideia do ModMan, o capítulo corrente aborda a prova de conceito do sistema e é composto por cinco seções. A Seção 4.1 traz os elementos sobre a prova de conceito, a Seção 4.2 traz o questionário aplicado às pessoas que testarem o sistema e a Seção 4.3 discorre sobre as avaliações que foram realizadas. A Seção 4.4 apresenta algumas considerações relevantes sobre a avaliação realizada. Por fim, a Seção 4.5 traz as possibilidades que podem influenciar negativamente na validação da proposta.

4.1 Prova de conceito

A prova de conceito, tradução do conhecido termo em inglês PoC (Proof of Concept) é utilizado para desenvolver na prática, artefatos que possam mostrar o funcionamento de alguma coisa que ainda se encontre em fase teórica. Segundo Basili [2], a definição de *prova de conceito* pode ser resumida como um estudo pequeno em que finalidade é a validação da funcionalidade da aplicação da nova ideia. Além disso, a definição do estudo piloto pode ser resumida como um pequeno estudo desenvolvido após o estudo de prova de conceito, e cujo objetivo é elaborar e provar uma protótipo de solução final em um pequeno caso real. Desta forma, a prova de conceito será utilizada para obter artefatos de softwaes que possam comprovar que o trabalho proposto realmente pode funcionar e, a partir disso, discutir a viabilidade da implementação do mesmo perante outros softwares.

O ModMan pode ser imaginado em diversas situações e por exemplo, é possível integrá-lo junto a um sistema cliente feito em plataforma Desktop, Web ou mobile. Ainda assim, tem a situação do sistema cliente ser um projeto novo ou em andamento. Desta forma, a prova de conceito é aplicada testando a viabilidade em distintos cenários buscando embasar o funcionamento.

Neste trabalho, a prova de conceito busca por meio de dois desenvolvedores, construir

aplicações que consumam a API disponibilizada pelo ModMan. Para isso, fica a cargo de cada um, realizar a integração a problemas particulares. A fim de diversificar as provas de conceito, as mesmas contemplaram sistemas Web e aplicativos mobile, alternando em cenários novos e existentes. Em seguida, aplicou-se o questionário da Seção 4.2 buscando obter a percepção dos desenvolvedores quanto ao uso da abordagem proposta.

4.2 Questionário

Com o objetivo de contextualizar e obter detalhes sobre o teste realizado pelos desenvolvedores, os mesmos responderam a um questionário para fornecer dados sobre a eficácia do trabalho proposto e também sobre as suas características. O questionário foi aplicado após a construção da aplicação que consome o ModMan e a análise possibilitou tecer conclusões pertinentes à abordagem proposta. O questionário é dividido em duas seções: Background, que coleta características pessoais dos testadores e Aspecos da implementação, que coleta dados relevantes de cada teste realizado.

Background:

- **Questão 1:** Qual o seu tempo de experiência com desenvolvimento de software?

  Opções de resposta:

  - 1 a 2 anos
  - 3 a 5 anos
  - Mais de 5 anos

- **Questão 2:** Qual o seu tempo de experiência com desenvolvimento de sistemas Web?

  Opções de resposta:

  - 1 a 2 anos
  - 3 a 5 anos
  - Mais de 5 anos

- **Questão 3:** Experiência com o uso de frameworks Web (prover reuso de software)

  Opções de resposta: Escala de 1 a 5 (1=Nenhuma, 5=Expert)

- **Questão 4:** Qual(s) framework(s) de desenvolvimento Web você costuma utilizar?

  Opção de resposta: Texto curto

- **Questão 5:** Em linhas gerais, qual o nível de preocupação das equipes em prover o reuso de software?

  Opções de resposta: Escala de 1 a 5 (1=Muito baixo, 5=Muito alto)

- **Questão 6:** Você costuma utilizar o módulo de controle de acesso e permissão fornecido pela ferramenta que trabalha ou adota evoluções para um padrão próprio?

  Opções de resposta:

  - Adoto o do framework que trabalho sem alterações
  - Adoto o do framework que trabalho mas costumo realizar melhorias
  - Costumo desenvolver um modelo próprio

- **Questão 7:** Numa empresa que trabalha com diversas tecnologias de desenvolvimento de software, em qual grau você classifica como desvantagem a adoção de diferentes formas de controle dos módulos e perfis devido à variação das tecnologias?

  Opções de resposta:

  - Irrelevante
  - Atrapalha a manutenção mas pode ser mediado sem problema
  - Traz prejuízo para a manutenção

- **Questão 8:** Em qual grau você classifica a importância de uma empresa padronizar o módulo de permissão de acesso dos sistemas por ela desenvolvidos?

  Opções de resposta: Escala de 1 a 5 (1=Muito baixo, 5=Muito alto)

- **Questão 9:** Com a disponibilidade de um serviço de controle de permissão de acesso, você migraria sistemas que já estão em produção a fim de padronizar os seus sistemas ?

  Opções de resposta: Sim, Depende da complexidade da mudança e Não realizaria esse tipo de mudança num sistema em produção

- **Questão 10:** No caso do desenvolvimento de novos sistemas, você adotaria a ferramenta proposta de serviço de permissão de acesso?

  Opções de resposta: Sim, Talvez, dependendo da complexidade para aplicar e Não, usaria o modelo da própria ferramenta

- **Questão 11:** Você achou a proposta válida? Voltaria a utilizar? Comente a experiência/-percepção de uso da abordagem proposta.

  Opções de resposta: Texto livre

Aspectos da implementação:

- **Questão 12:** Quantas linhas de código foram inseridas para implementar o uso do sistema de permissões como serviço?

  Opções de resposta: Valor livre

- **Questão 13:** E quantas linhas foram alteradas?

  Opções de resposta: Valor livre

- **Questão 14:** Qual o nível de dificuldade de integração numa escala de 1 a 10 ?

  Opções de resposta: Escala de 1 a 10 (1=Muito fácil, 10=Muito difícil)

As questões tem aspectos técnicos, para coletar dados específicos relevantes ao processo de construção de um software, a fim de obter conhecimento sobre o impacto da implantação do sistema de permissões proposto. E para avaliar os itens técnicos, também é composto por perguntas que ilustram a experiência do desenvolvedor.

4.3 Avaliações

Esta seção detalha os testes realizados pelos desenvolvedores para realizar a avaliação e apresenta detalhes sobre a implementação como, por exemplo, as tecnologias utilizadas e o procedimento adotado para realizar a integração.

4.3.1 Teste 1

O teste 1 foi realizado pelo Desenvolvedor A numa aplicação existente do Ionic framework com integração ao Laravel Framework. Trata-se de um aplicativo de acompanhamento escolar e o teste realizado foi aplicado no menu do aplicativo, a fim de avaliar o comportamento do processo de integração com um aplicativo mobile já existente. O código fonte deste projeto não está disponível para consulta, pois trata-se de um projeto privado de uma empresa. A Figura 4.1 mostra o trecho de código que foi necessário ser acrescentado no software existente, de forma a realizar a integração. A responsabilidade deste trecho, é buscar os dados que foram cadastrados no sistema de permissões e organizá-los de maneira que a integração com a interface seja simples.

```
//Teste para sistema de controle de permissão como serviço.
//O restante da implementação, encontra-se em e-Grafite.html, no ng-show dos campos.
$scope.perfil = "Cliente";
$scope.permissoes = new Array();

dados = {};
dados.key = "M11";
//"http://modman.dev/api/endpoint/", M189
$http({
    url: "http://api.modman.ga/api/endpoint",
    dataType: "json",
    method: "POST",
    data: dados,
    headers: {
        "Content-Type" : "application/json"
    }
}).success(function(data) {

    angular.forEach(data, function(permissao, key){

        if (permissao['profile'] == "Administrador") {
            //console.log(permissao);
            $scope.permissoes[permissao['functionality']] = true;
        }
    });
    console.log($scope.permissoes);

}).error(function(data){
    console.log(data);
});
```

Figura 4.1 Código do controller do teste 1

A Figura 4.2 mostra como tal integração ocorreu, bastando acrescentar uma verificação nas tags dos itens do menu. O funcionamento da integração do teste 1 ocorreu com sucesso pois as permissões foram devidamente aplicadas.

4.3.2 Teste 2

O teste 2 foi realizado numa nova aplicação Web em PHP desenvolvida exclusivamente para testar o ModMan. O objetivo foi parametrizar a disponibilidade de ações num formulário utilizando somente o Laravel Framework. A aplicação foi desenvolvida pelo mesmo Desenvolvedor A que realizou o Teste 1 e encontra-se disponível no Github no link https://github.com/guiassemany/modman-concept. A Figura 4.3 mostra a Trait (Grupo de métodos para incluir em outra classe) CheckPermissionTrait, o qual contém o método hasPermission() que realiza a requisição POST para consultar as permissões. Assim, as classes importam a CheckPermissionTrait, tratando de acordo com a necessidade e aplicam na view, conforme mostrado na Figura 4.4. De acordo com

```
<ion-side-menu side="left">
    <ion-header-bar class="bar-stable">
        <div class="title">Menu</div>
    </ion-header-bar>
    <ion-content padding="false" class="side-menu-left has-header">
        <img src="img/logo1.png" style="display: block; margin: 0 auto;">
        <ion-list>
            <ion-item ng-if="permissoes['Faltas']" menu-close="" href="#/menu/inicio"> <i class="ion-home"> </i>  Início
            </ion-item>
            <ion-item ng-if="permissoes['Dependentes']" menu-close="" ng-if="tipo=='HQXs'" href="#/menu/dependentes"> <i
             class="ion-university"> </i>  Dependentes</ion-item>
            <ion-item ng-if="permissoes['Requerimentos']" menu-close="" href="#/menu/requerimento_cons"> <i class="
            ion-document-text"> </i>  Requerimentos</ion-item>
            <ion-item ng-if="permissoes['Notas']" menu-close="" href="#/menu/disciplinas_notas_cons"> <i class="
            ion-ios-compose"> </i>  Notas</ion-item>
            <ion-item ng-if="permissoes['Livros']" menu-close="" href="#/menu/emprestimos_livros"> <i class="
            ion-ios-book"> </i>  Livros</ion-item>
            <ion-item ng-if="permissoes['Faltas']" menu-close="" href="#/menu/disciplinas_falta_cons"> <i class="
            ion-android-calendar"> </i>  Faltas</ion-item>
            <ion-item ng-if="permissoes['Atendimentos médicos']" menu-close="" href="#/menu/atendimentos_medicos"> <i
            class="ion-medkit"> </i>  Atendimentos Médicos</ion-item>
            <ion-item ng-if="permissoes['Ocorrências']" menu-close="" href="#/menu/ocorrencias_disciplinares"><i class="
            ion-ios-folder"> </i>  Ocorrências</ion-item>
            <ion-item ng-if="permissoes['Matrículas']" menu-close="" href="#/menu/graduacoes_aluno"> <i class="
            ion-university"> </i>  Matrículas</ion-item>
            <ion-item ng-if="permissoes['']" menu-close="" ng-click="menu.sair();"> <i class="ion-android-exit"> </i>
            Sair</ion-item>
        </ion-list>
    </ion-content>
</ion-side-menu>
```

Figura 4.2 Código da view do teste 1

a estrutura adotada, o teste 2 foi executado com sucesso pois as permissões foram devidamente aplicadas.

4.3.3 Teste 3

O teste 3 foi realizado pelo Desenvolvedor B numa aplicação Web existente, desenvolvida com PHP no back-end e Javascript no front-end com arquitetura de Single page Application, a fim de avaliar o comportamento da integração e possíveis complicação durante o processo. O sistema do teste 3 é de gerenciamento de oficina mecânica e foi desenvolvido com o Laravel Framework e o AngularJs e não contém o código fonte disponível para demonstração por se tratar de um software proprietário. O objetivo do teste foi aplicar as devidas permissões para a dashboard do sistema, representada na Figura 4.5 que contém acessos rápidos aos usuários e deve ter os itens disponíveis somente para o funcionário que trata do correspondente setor da oficina. Para realizar a integração, o código da Figura 4.6 trata do back-end, realizando uma requisição POST que envia o devido parâmetro "key", tratando o JSON retornado para utilizá-lo na view, que apenas verifica a disponibilidade da variável da permissão para exibir ou não o determinado item da dashboard. Utilizando a estrutura apresentada, o teste 3 ocorreu com sucesso.

```php
<?php
namespace App\Modman;
use GuzzleHttp\Client;
use Illuminate\Support\Facades\Auth;
trait CheckPermissionTrait {
    public function hasPermission()
    {
        $client = new Client(['base_uri' => 'http://api.modman.ga/api/']);
        $response = $client->request('POST', 'endpoint', ['json' => ['key' => $this->key]]);
        $permsArray = [];
        $retorno = json_decode($response->getBody());
        foreach ($retorno as $permission){
            $permsArray[] = strtoupper($permission->profile);
        }
        if(!empty($permsArray)){
            return in_array(strtoupper(Auth::user()->role), $permsArray);
        }
        return false;
    }
}
```

Figura 4.3 Trait desenvolvida no teste 2

```blade
<div class="col-md-4">
    @if($employeeModule)
        <div class="panel panel-default">
            <div class="panel-body">
                Funcionários
            </div>
            <div class="panel-footer">
                <a href="{{route('employee.home')}}">Acessar <i class="glyphicon
                glyphicon-arrow-right"></i></a>
            </div>
        </div>
    @endif
</div>
```

Figura 4.4 Código da view do teste 2

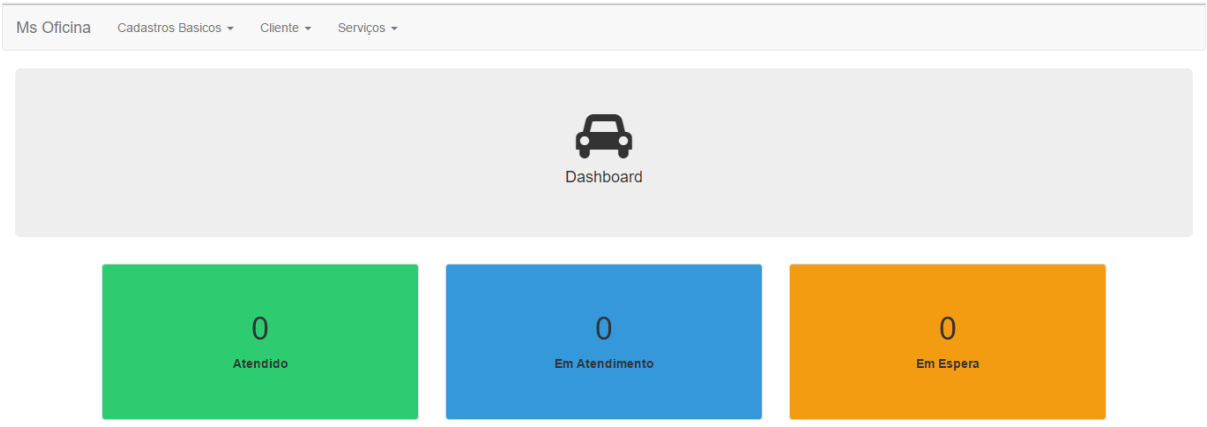

Figura 4.5 Dashboard do sistema do teste 3

dashboard.html × dashboard.js ×

```
7         $scope.em_atendimento    = 0;
8         $scope.atendido          = 0;
9         $scope.em_espera         = 0;
10
11
12
13        $scope.perfil = "Administrador";
14        $scope.permissoes = new Array();
15
16        dados = {};
17        dados.key = "M22";
18
19        $http({
20            url: "http://modman.dev/api/endpoint/",
21            dataType: "json",
22            method: "POST",
23            data: dados,
24            headers: {
25                "Content-Type" : "application/json"
26            }
27        }).success(function(data) {
28
29        angular.forEach(data, function(permissao, key){
30
31            if (permissao['profile'] == "Cliente") {
32                //console.log(permissao);
33                $scope.permissoes[permissao['functionality']] = true;
34            }
35        });
36        console.log($scope.permissoes);
37        console.log($scope.permissoes['Faltas']);
38
39        }).error(function(data){
40            console.log(data);
41        });
```

Figura 4.6 Controller de integração do teste 3

4.4 Considerações

Esta seção traz uma análise sobre as avaliações realizadas discorrendo acerca do questionário que os avaliadores responderam. Para prover a discussão, a Tabela 4.4 contém as respostas que se referem ao background dos avaliadores e a 4.4 contém os aspectos da implementação. Com base nas resposta fornecidas ao formulário é possível notar que os avaliadores têm experiência considerável em tecnologias utilizadas no desenvolvimento Web, o que pode ter colaborado para a integração simples e de baixo impacto, de acordo com as respostas de ambos. A constatação de facilidade na integração do sistema proposto com sistemas clientes é baseada no feedback de edição e acréscimo de poucas linhas de código, e a facilidade ocorreu nos ambientes Web e mobile, o que evidencia a interoperabilidade.

Ambos avaliadores responderam que costumam desenvolver um modelo próprio de permissão nos softwares, o que favorece a adoção deste projeto pois seria possível reduzir este esforço. De acordo com o resposta do Desenvolvedor A na questão 14, tem-se o feedback de que o integração do projeto é simples e funcional, entretanto, houve a sugestão de possibilitar a ativação/inativação dos cadastros no sistema. Essa sugestão é bastante pertinente pois possibilita maior flexibilidade sem necessidade de codificação, o que facilita a manutenção e abrange mais a possibilidade da administração das permissões por meio de pessoas que não tem conhecimento de desenvolvimento de software. Os avaliadores responderam que usariam este trabalho, entretanto, vale ressaltar que de acordo com eles, a viabilidade de integração depende da complexidade da mudança no dado contexto, que nem sempre poderá ser favorável como os aqui apresentados.

4.5 Ameaças

Conforme apresentado na prova de conceito do presente capítulo, este trabalho mostrou-se viável. Entretanto, é importante citar que apesar da obtenção do sucesso na integração, alguns fatores podem ter influenciado para o sucesso. Dentre esses fatores, cita-se a realização da avaliação com softwares pequenos e de arquitetura semelhante, o que deixa em aberto a análise de viabilidade em sistemas de maior porte. O mesmo vale para a arquitetura do software cliente, pois é possível integrar com diversas linguagens, inclusive de plataforma Desktop, mas não foram desenvolvidas provas para esses cenários. Também, os avaliadores detém forte conhecimento no contexto, o

Tabela 4.1 Respostas fornecidas no formulário - Background

Questão	Guilherme Assemany (Desenvolvedor A)	Matheus Marques (Desenvolvedor B)
1	Mais de cinco anos	3 a 5 anos
2	Mais de cinco anos	3 a 5 anos
3	3	3
4	Laravel, Slim, Angular, React	Laravel
5	4	3
6	Costumo desenvolver um modelo próprio	Costumo desenvolver um modelo próprio
7	Irrelevante	Traz prejuízo para a manutenção
8	5	5
9	Depende da complexidade da mudança	Depende da complexidade da mudança
10	Sim	Sim
11	Sim. O projeto proposto é bastante eficiente e verdadeiramente utilizável em aplicações. Além de ser fácil de implementar, dá uma possibilidade de enxergar um sistema de forma mais modular e ter mais controle sob suas funcionalidades. Como uma melhoria, seria bom se houvesse algum marcador de ativo / inativo no gerenciamento das regras dos módulos e das funcionalidades, dessa forma seria mais agradável para o utilizar controlar o status de cada pequena parte do sistema.	Sem resposta

Tabela 4.2 Respostas fornecidas no formulário - Referente à implementação

Questão	Teste 1	Teste 2	Teste 3
12	40 a 60	22	30
13	5	3	4
14	2	4	3

que naturalmente tornou o procedimento descomplidado. Possivelmente, um programador que não tem experiência sólida no desenvolvimento Web encontre dificuldade na integração e aponte deficiências ainda não detectadas.

5 Conclusão

Diante da proposta deste trabalho de conclusão de curso e da avaliação realizada, tem-se que a proposta teve um funcionamento correto. Entretanto, é necessário realizar uma discussão acerca do trabalho desenvolvido, comentando aspectos conclusivos. Assim, o capítulo corrente é composto por três seções. A seção 5.1 traz uma discussão sobre a percepção em caráter geral sobre o contexo em que o trabalho foi aplicado. A seção 5.2 traz trabalhos relacionados ao aqui apresentado, provendo comparação sobre o modelo de funcionamento da ideia. E por fim, a seção 5.3 discorre sobre possíveis caminhos para a evolução do trabalho aqui desenvolvido.

5.1 Percepção geral

Diante dos resultados apresentados no capítulo anterior, através da prova de conceito realizada, tem-se que este trabalho pode colaborar com a redução do esforço para desenvolver um software e propor a padronização da parte de controle de acesso. Também, foi constatada a aplicabilidade deste trabalho em softwares de diferentes plataformas (Web e mobile), exemplificando que é possível expandir ainda mais a padronização de desenvolvimento de software, gerando componentes universais que facilitem o entendimento e acelerem o desenvolvimento dos softwares. Também foi constatado que é possível aplicar a padronização em softwares novos e legados. Devido ao baixo esforço necessário para a integração deste serviço, conforme o feedback das avaliações realizadas, é sempre possível pensar em novas ideias que facilitem as atividades realizadas.

5.2 Trabalhos relacionados

Conforme já apresentado, este trabalho de conclusão de curso propõe um software como serviço que atende ao controle de acesso do software cliente. Essa situação tem o mesmo fundamento que

o OAuth[1], que fornece esquema de autenticação para os sites/softwares clientes. Com o OAuth, é possível integrar ao software cliente a autenticação via e-mail, Facebook, Google, Github, Twitter e etc, atendendo a diversas plataformas. O processo de autenticação do OAuth é seguro e útil aos clientes, pois os mesmos não precisam realizar novamente extensos cadastros nos sites, porque o site correspondente à forma escolhida de autenticação (Via Facebook, Google...) já dispõe de tais informações dos usuários.

O Firebase[2], que pertence ao Google, também tem abordagem semelhante ao ModMan. O Firebase disponibiliza uma série de serviços que podem ser integrados ao projetos de terceiros mediante correta configuração. Entre os serviços disponibilizados, tem-se notificações Push, banco de dados em tempo real e Analytics. O foco é atender aos dispositivos móveis Android e iOS e também sistemas Web. Devido à interoperabilidade, o Firebase tem suporte à comunicação com diversas linguagens, deixando que o desenvolvedor decida a melhor escolha para o seu projeto. A principal diferença do ModMan para os serviços supracitados se dá pelo tipo de serviço que é fornecido, pois o controle de permissão que o ModMan fornece não é disponibilizado pelo OAuth e Firebase.

5.3 Direções futuras

Conforme foi mencionado no feedback das avaliações realizadas sobre o trabalho proposto, existem pontos que notoriamente podem ser evoluidos no mesmo. Um item essencial, é disponibilizar a possibilidade de deixar uma configuração de permissão como ativa ou intaiva, pois atualmente para "inativar"uma configuração cadastrada, é necessário deletar o correspondente cadastro. Essa procedimento gera transtorno para quem administra as permissões, tornando pertinente a evolução dessa funcionalidade. Outra possibilidade de evolução ao sistema, é a replicação das permissões configuradas a um perfil, sistema ou módulo. Dessa forma, caso esteja configurando as permissões de um sistema complexo, o processo de cadastro tornaria-se mais rápido e eficiente.

[1] https://oauth.net/
[2] https://firebase.google.com/?hl=pt-br

Referências Bibliográficas

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

[2] Basili, V. R. and Rombach, H. D. (1991). Support for comprehensive reuse. *Softw. Eng. J.*, **6**(5), 303–316.

[3] Chen, H. (2016). 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)*, pages 382–385.

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

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

[6] Keswani, R., Joshi, S., and Jatain, A. (2014). Software reuse in practice. In *2014 Fourth International Conference on Advanced Computing Communication Technologies*, pages 159–162.

[7] Kumar, S. R., Sharma, R., and Gupta, K. (2016). Strategies for web application development methodologies. In *2016 International Conference on Computing, Communication and Automation (ICCCA)*, pages 160–165.

[8] La, H. and Kim, S. (2009). A Systematic Process for Developing High Quality SaaS Cloud Services. In M. Jaatun, G. Zhao, and C. Rong, editors, *Cloud Computing*, volume 5931 of *Lecture Notes in Computer Science*, chapter 25, pages 278–289. Springer Berlin / Heidelberg, Berlin, Heidelberg.

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

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

[11] RedMonk (2016). Gráfico de linguanges - redmonk. http://goo.gl/vDSGLx.

[12] Salvadori, I. and Siqueira, F. (2015). A maturity model for semantic restful web apis. In *2015 IEEE International Conference on Web Services*, pages 703–710.

[13] Tesarik, J., Dolezal, L., and Kollmann, C. (2008). 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)*, pages 223–228.

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

[15] Yuping, J. (2014). Research and application of ajax technology in web development. In *2014 IEEE Workshop on Electronics, Computer and Applications*, pages 256–260.

Apêndice