



“Sistema de gerenciamento de módulos”

Por

Victor de Azevedo Nunes

Trabalho de Graduação



Universidade Federal da Bahia
ceapgmt@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	xvii
Lista de Tabelas	xix
Lista de Acrônimos	xxi
1 Introdução	1
1.1 Motivação	1
1.2 Problema	1
1.3 Objetivos	2
1.4 Metodologia	2
1.5 Resultados Esperados	5
1.6 Fora de Escopo	5
1.7 Estrutura do Trabalho	6
2 Referencial Teórico	7
2.1 Software como serviço	7
2.2 Reuso de software	9
2.3 Modularização	11
2.4 Aplicações web	12
3 Proposta	13
3.1 Arquitetura da aplicação	13
3.1.1 Tecnologias utilizadas	15
Laravel framework	15
AngularJs	15
Bootstrap	16
3.1.2 Diagramas UML	16
3.1.3 Funcionalidades	16
Requisitos não funcionais	16
Requisitos funcionais	19
3.1.4 Processo	21
3.1.5 O que há de novo ?	21
Apêndice	25

Lista de Figuras

1.1	Ranking das linguagens de programação no Stack Overflow e Github . . .	3
1.2	Infográfico da WebhostFace, exibindo a popularidade dos Frameworks PHP em 2015	4
1.3	Gráfico do Google Trends exibindo as pesquisas por ferramentas front-end	5
3.1	Gráfico do Google Trends exibindo comparação entre as pesquisas por SPA e MVC	15
3.2	Diagrama de caso de uso	17
3.3	Diagrama entidade relacionamento	18
3.4	Diagrama do processo(BPMN) do software em questão	22

Lista de Tabelas

Lista de Acrônimos

ALM Application Lifecycle Management

1

Introdução

Numa organização de desenvolvimento de software, normalmente encontra-se problema relacionado ao cumprimento dos prazos de entrega. Tal situação pode ocorrer devido a diversos fatores, 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 diversos fatores, entretanto, existem elementos universais que podem colaborar para a melhoria, como o reuso de software. Assim, este trabalho de conclusão de curso busca padronizar a parte de controle de acesso dos softwares, tornando-a universal para tornar possível que seja reutilizada como um serviço em qualquer circunstância de desenvolvimento, gerando assim uma redução no tempo de desenvolvimento e manutenção.

1.1 Motivação

Otimizar o desenvolvimento de um software é uma tarefa árdua e bastante estudada. A área de Engenharia de software estuda maneiras de melhorar o desenvolvimento de software através de bons 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 agradar desde programador, com um trabalho facilitado, até o cliente final com redução de custo.

1.2 Problema

A repetição de código durante o desenvolvimento de um software, acaba por aumentar a sua complexidade e tempo de desenvolvimento. O impacto dessa má prática atinge fortemente a manutenibilidade do software, gerando grande transtorno diante de uma

alteração que poderia ser simples caso estivesse bem projetado desde o começo. Assim, o reuso de código sempre é buscado num projeto. 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 padrões que mantenham o componente de código na melhor forma possível para ser acoplado em outros softwares sem transtorno.
- **Objetivo específico:** Reutilizar o módulo de controle de acesso de um sistema, já que o mesmo está presente sem desvios significativos 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 controle de acesso que seria um projeto à parte. Tal situação também tem a vantagem de ter interoperabilidade entre linguagens de programação, já que realizaria a comunicação através de JSON, seguindo padrões atuais. Assim, teria-se uma constância no módulo de acesso independentemente da possível necessidade de migração de tecnologia.

1.4 Metodologia

Baseando-se nas tecnologias gratuitas em alta no cenário atual do desenvolvimento web, 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. Como linguagem de programação, adota-se o PHP. A escolha é fundamentada de acordo com a 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 com o gráfico da pesquisa citada, na qual o PHP é apresentado na terceira colocação, apenas atrás do líder JavaScript e do segundo colocado, o Java.

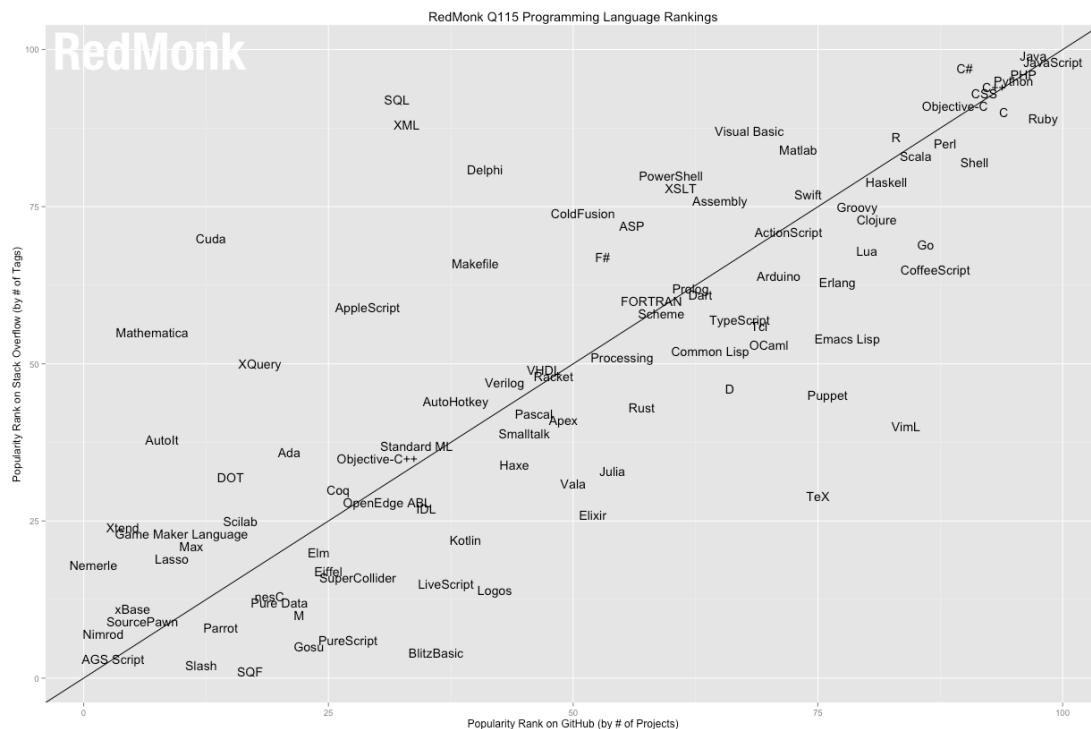


Figura 1.1 Ranking das linguagens de programação no Stack Overflow e Github

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 seguinte infográfico:

Como pode ser verificado no gráfico [2], 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 3, que propõe comparações entre termos pesquisados, revela a popularidade do AngularJs diante de alguns dos principais concorrentes. O gráfico abaixo evidencia o cenário.

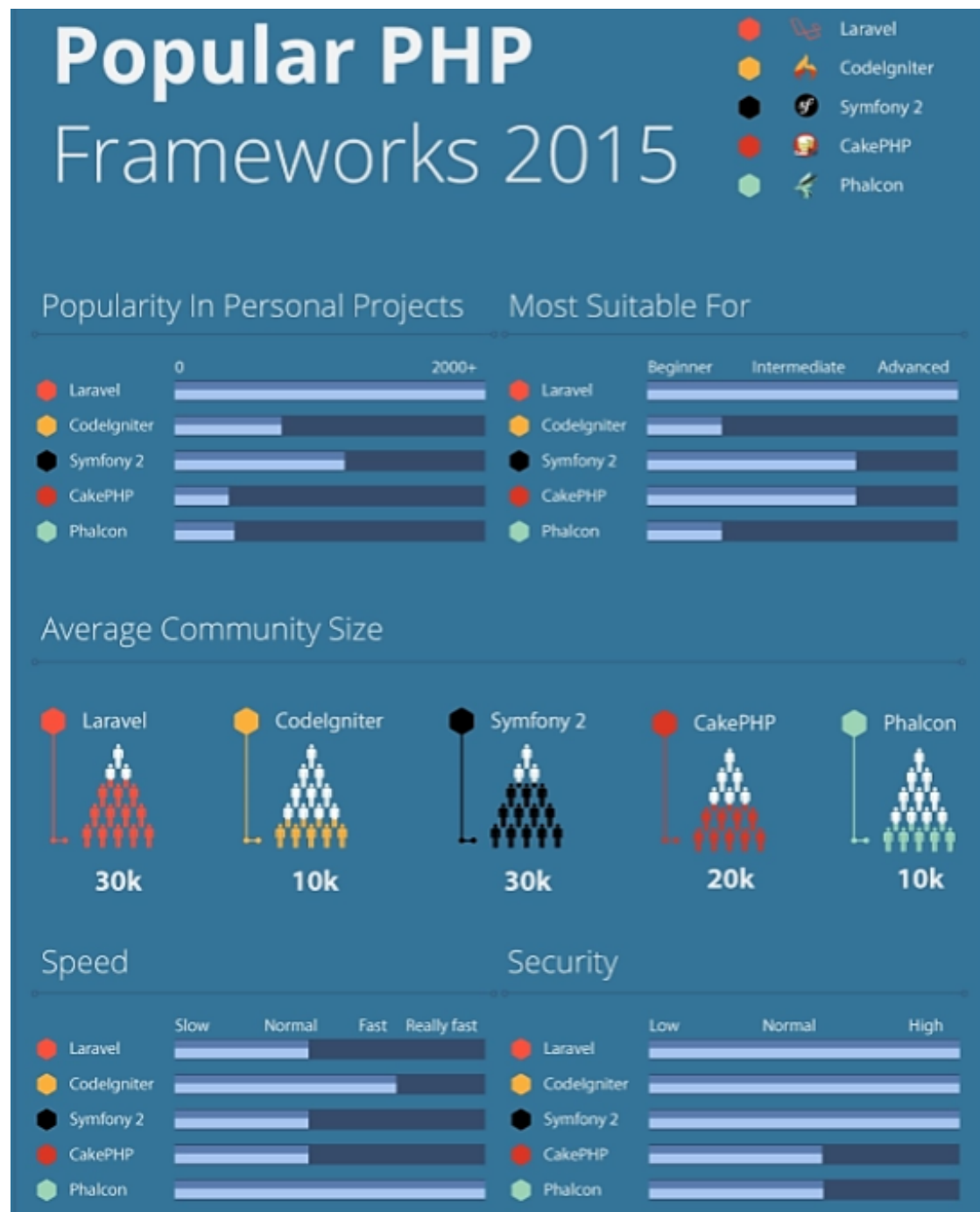


Figura 1.2 Infográfico da WebhostFace, exibindo a popularidade dos Frameworks PHP em 2015

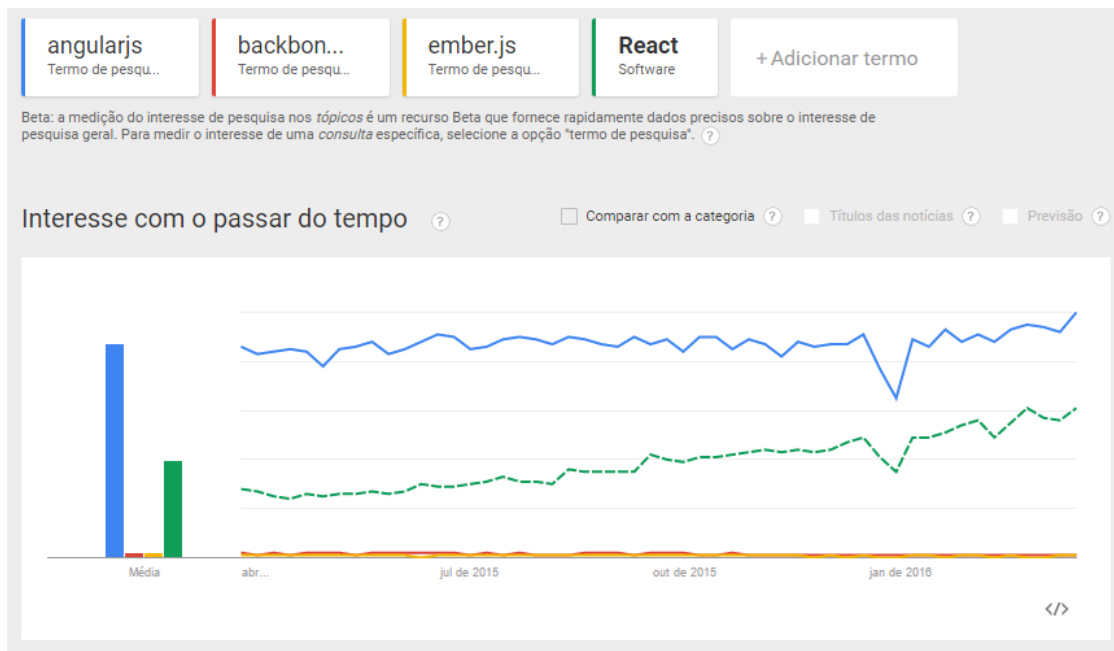


Figura 1.3 Gráfico do Google Trends exibindo as pesquisas por ferramentas front-end

1.5 Resultados Esperados

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

- Economia no tempo de desenvolvimento
- Economia do custo do projeto
- Redução da complexidade de código
- Maior facilidade para manutenção

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, fazendo a sinalização de que o usuário está 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:
- Capítulo 5:

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 nesta seção, uma breve discussão sobre alguns elementos de projeto e arquitetura de software, a fim de contextualizar este trabalho de conclusão de curso.

2.1 Software como serviço

A definição de SaaS encontra-se muito bem elaborada em um dos trabalhos listados na literatura. Segundo La e Chun [4], o princípio da definição de Software como um Serviço (Software as a Service - SaaS) é um serviço complementar para aplicações da computação em nuvem (cloud computing). As duas áreas estão interligadas, no entanto, não se confundem, pois o 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 propõe vantagens, software como serviço é uma tendência forte, isso graças à evolução da web. Diversos fatores podem ser favoráveis para a adoção de um SaaS, como custo e manutenção dentre outros fatores aplicáveis a determinados contextos. Um trabalho recente realizado por Lechessa et al. [5] apresenta uma pesquisa qualitativa sobre os fatores determinantes para adoção ou não de um SaaS voltado para ERP na África do Sul. Esses autores indicam que 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. O artigo de Rai et al. [6] orienta como 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. [6], 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 muito importante para qualquer trabalho de ponta na área de SaaS.

A arquitetura de armazenamento de dados de um SaaS pode variar de acordo com a necessidade do contexto. O artigo recente de Huixin [7], exemplifica possíveis modelagens para utilizar. Tal 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. Tal 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 diferente conceito de obtenção de software, tanto pela visão do cliente como pela visão do vendedor, é necessário tomar conhecimento dos diversos fatores que podem ser relevantes ao orçar um SaaS. O recente trabalho de T. Kaur et al. [8] 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.

2.2 Reuso de software

Para Peter Freeman, o reuso é a utilização de qualquer informação que um desenvolvedor pode necessitar no processo de criação de software (Ezran et al., 2002). Basili e Rombach definem reutilização de software como o uso de tudo o que está associado a um projeto de conhecimento (Basili e Rombach, 1991). 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. [9], 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:

- **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 menos esforços de teste e também economiza análise e design, gerando economia em diversos escopos 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 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 menos desenvolvimento, tempo e custo associado.
- **Documentação:** Embora a documentação seja muito importante para a manutenção de um sistema, muitas vezes é negligenciada. A reutilização de componentes

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 entrar 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 melhores comunicações e aumento da produtividade.

Apesar dos benefícios da reutilização de software, a mesma não é amplamente praticado como imagina-se. 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:

- **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 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 reutilizáveis componentes do sistema. 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.

2.3 Modularização

Conforme é frisado no trabalho de Wickramarachchi e Lai [10], 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 Wickramarachchi e Lai [10], 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 e a qualidade do software. Tal conceito encontra-se bastante difundido e está incorporado em linguagens de programação e ferramentas de software. O trabalho proposto 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.

2.4 Aplicações web

A popularidade da aplicação web aumentou exponencialmente na última década e todos os dias cresce o número de pessoas usuárias de aplicações web. E seguindo o padrão de desenvolvimento de software, Kumar et al. [11] 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 diagramas UML. 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. [12] propõem o uso do modelo MVC (Model, View e Controller) no atual 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 no artigo de Yuping [13], 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.

3

Proposta

Seguir padrão em desenvolvimento de software é uma premissa básica, pois ao trabalhar com variações dentro de uma organização, a tendência é gerar confusão. Assim, o objetivo deste trabalho é formular 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 bastante comum encontrar frameworks de desenvolvimento de software que automaticamente geram um esquema de configuração de módulos com os perfis de acesso. Entretanto, apesar de ser um facilitador, tal política pode se tornar confusa em alguns 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, podendo gerar uma confusão.

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

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 usada em aplicações web e mobile. SPA basicamente significa codificar menos no server-side e mais no client-side. Assim,

boa parte da aplicação passa a ser processada no cliente (dentro do navegador Web).

Vantagens da SPA:

- Balanceamento da responsabilidade da execução entre cliente e servidor.
- Menos código do servidor, e mais responsabilidade no cliente;
- Melhorar a experiência ao usuário (UX) criando interface com usabilidade moderna e de fácil entendimento do usuário;
- Menor consumo de banda, pois as cargas de dados são feitas por demanda e por AJAX.

O grande ator de app SPA é o código Javascript executado no cliente. Toda a aplicação pode ser construída simplesmente manipulando 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. Dentre as diversas bibliotecas Javascript disponíveis no momento, as mais difundidas na comunidade de programadores estão: AngularJs, VueJs, Backbone, ReactJs, Ember e outras.

No lado servidor, temos a execução das linguagens tradicionais como PHP(Adotada nesse projeto), ASP.NET, JSP, etc, trabalhando também de forma tradicional, servindo arquivos, acessando a banco de dados, tratando as regras de negócios que não podem estar no código JS por questões de segurança. E é neste lado (servidor) que podemos utilizar a arquitetura REST – Representational State Transfer para fornecer serviços do servidor para nossa aplicação SPA. É comum encontrar aplicação SPA utilizando serviços RESTFul. Uma aplicação no servidor que utiliza a arquitetura REST para servir 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.

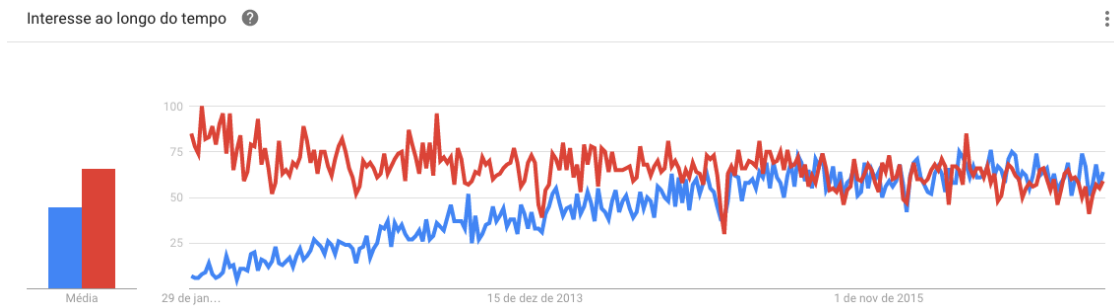


Figura 3.1 Gráfico do Google Trends exibindo comparação entre as pesquisas por SPA e MVC

3.1.1 Tecnologias utilizadas

Laravel framework

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.

AngularJs

AngularJS é um framework JavaScript open-source, mantido pelo Google, que auxilia na execução de single-page applications. Seu objetivo é aumentar aplicativos que podem ser acessados por um navegador web e foi construído sob o padrão model-view-view-model (MVVM).

A biblioteca 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 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, carrousel, slideshow, entre outros com facilidade, apenas acrescentando algumas configurações no código.

3.1.2 Diagramas UML

FALTA INSERIR O DIAGRAMA DE CLASSE

3.1.3 Funcionalidades

A fim de apresentar as funcionalidades do sistema proposto, nesta seção serão apresentados os requisitos de acordo com a engenharia de software.

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 no banco de dados.

- RN03: Alta disponibilidade

Descrição: O sistema deverá ter disponibilidade de aproximadamente 99 por cento do tempo.

- RN04: Desempenho

Descrição: Os tempos de resposta das consultas não devem ultrapassar 3 segundos.

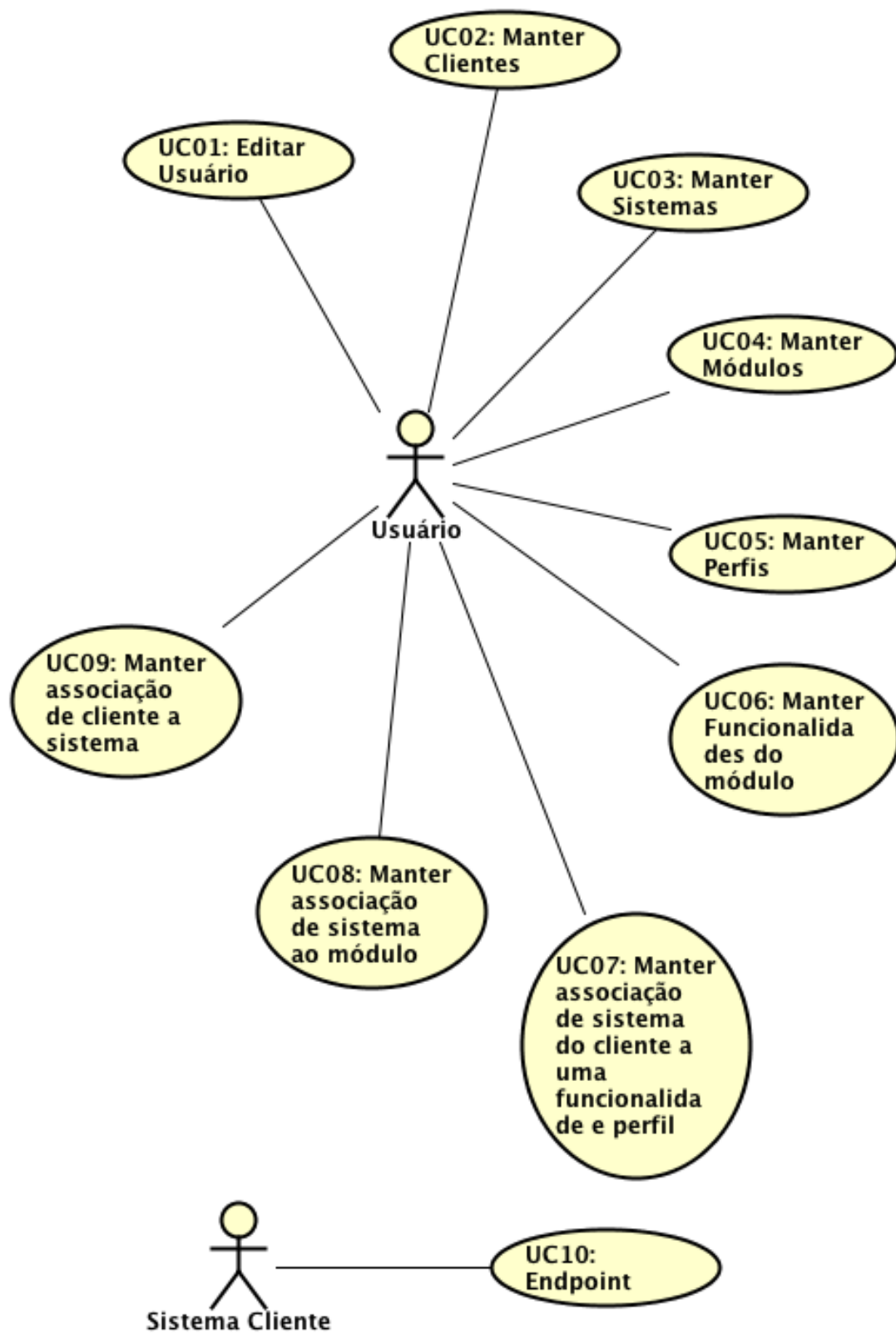


Figura 3.2 Diagrama de caso de uso

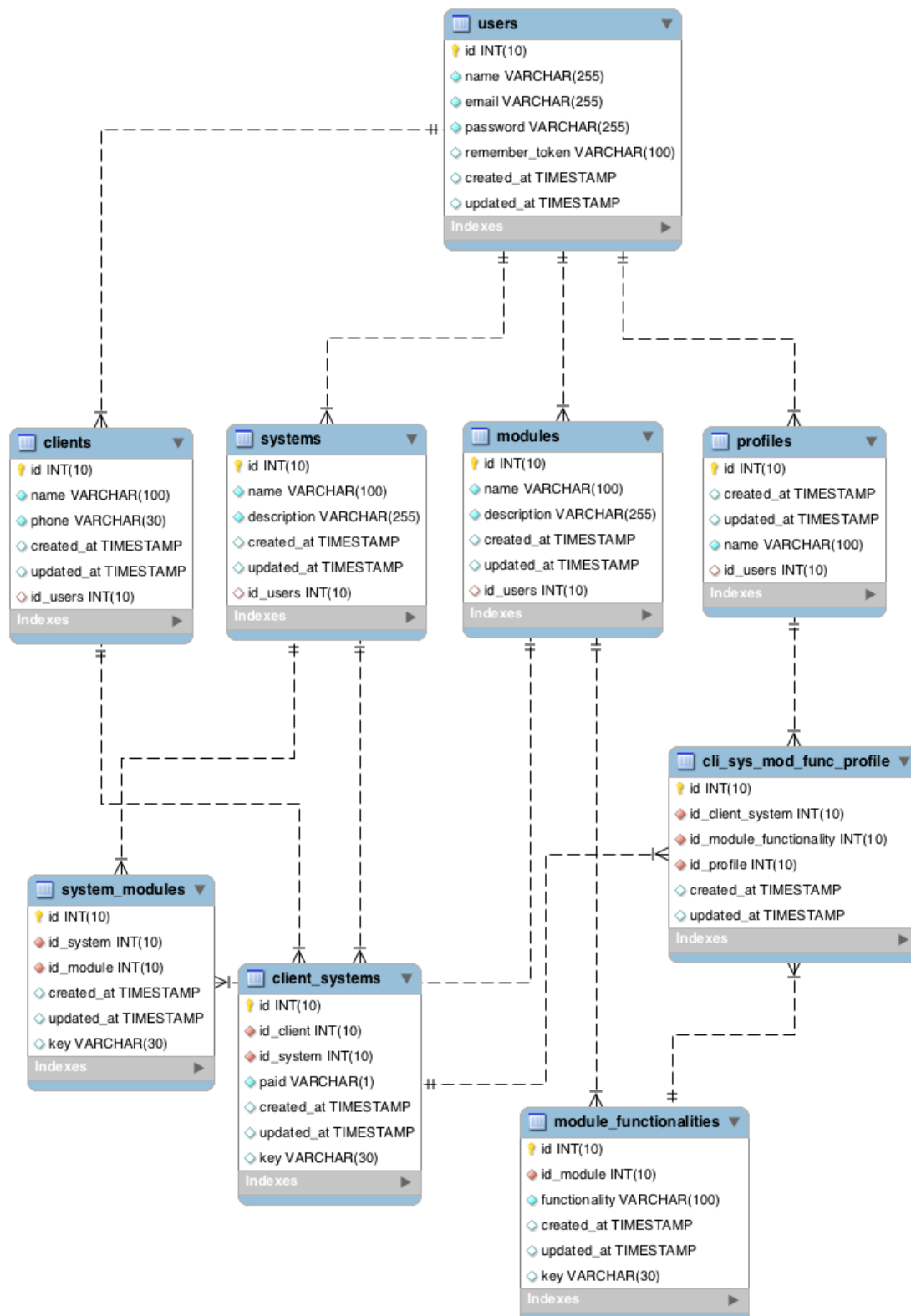


Figura 3.3 Diagrama entidade relacionamento

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 caso de uso está relacionado ao UC01.

- 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 caso de uso está relacionado ao UC02.

- 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 caso de uso está relacionado ao UC03.

- 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 caso de uso está relacionado ao UC04.

- 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 perfis cadastrados para o usuário da sessão. Este caso de uso está relacionado ao UC05.

- 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 caso de uso está relacionado ao UC09.

- 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 caso de uso está relacionado ao UC08.

- 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 caso de uso está relacionado ao UC06.

- 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 caso de uso está relacionado ao UC07.

- 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 requisição enviando a "key", o sistema deve retornar um texto no formato Json, contendo os dados das

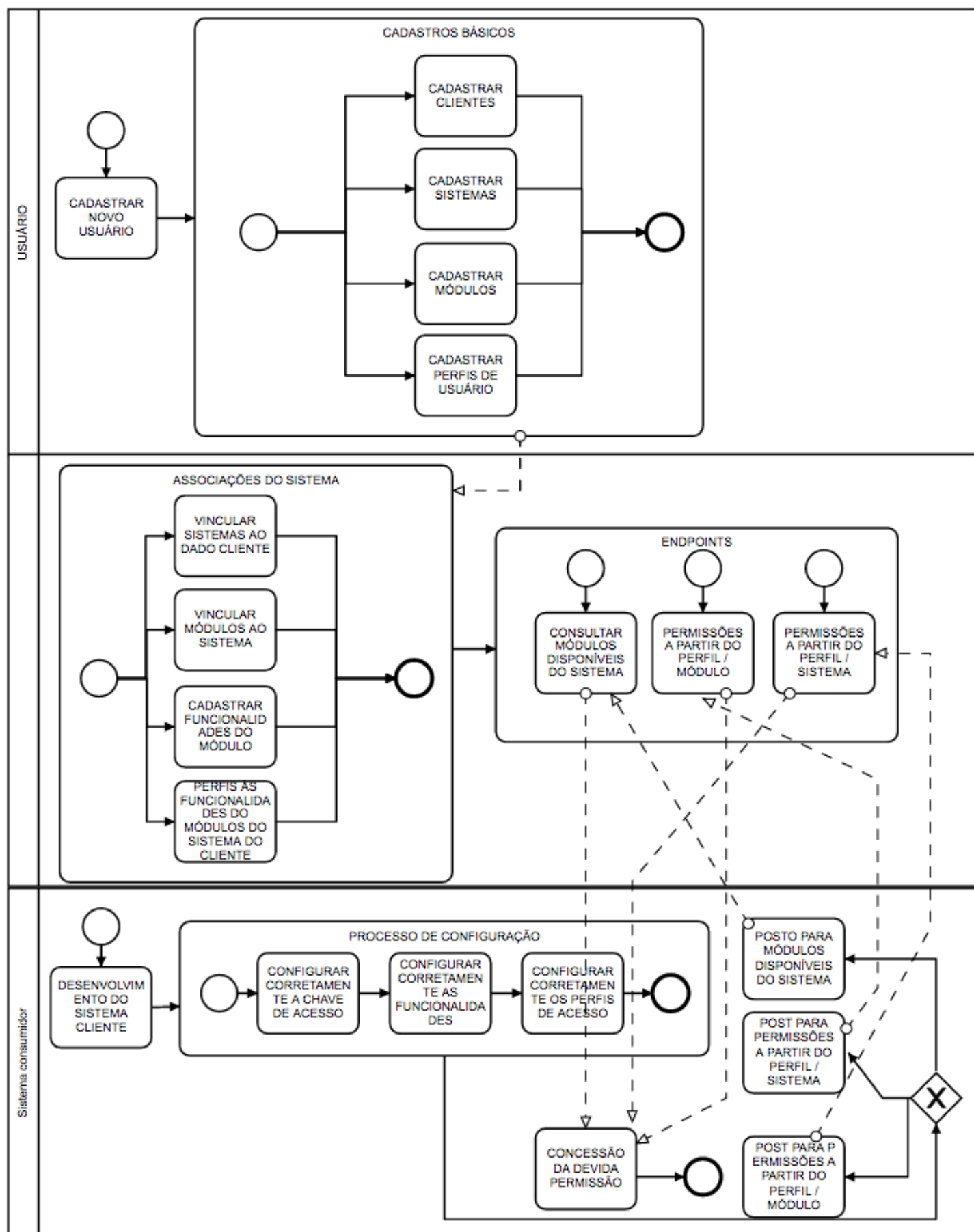
permissões que foram compostas no sistema pelo usuário. A chave enviada pelo usuário pode ser de qualquer um dos níveis de configuração realizado no sistema e o endpoint deve retornar a resposta sempre num mesmo formato, viabilizando um tratamento único da resposta pelo cliente. Este caso de uso está relacionado ao UC10.

3.1.4 Processo

Para melhor compreensão do sistema proposto, abaixo temos um diagrama BPMN do processo do software.

3.1.5 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.



Referências Bibliográficas

- [1] RedMonk, “Gráfico de linguagens - redmonk.” <http://goo.gl/vDSGLx>, 2016.
- [2] W. Face, “Popular frameworks php 2015.” <https://goo.gl/DjUREE>, 2016.
- [3] Google, “Googletrends - frameworks front-end mais buscados em 2015.” <https://goo.gl/VUhIrv>, 2016.
- [4] 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.
- [5] 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.
- [6] R. Rai, G. Sahoo, and S. Mehruz, “Securing software as a service model of cloud computing: Issues and solutions,” *CoRR*, vol. abs/1309.2426, 2013.
- [7] 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.
- [8] 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.
- [9] 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.
- [10] 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.

REFERÊNCIAS BIBLIOGRÁFICAS

- [11] 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.
- [12] 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.
- [13] 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.

Apêndice

