

**UNIVERSIDADE ESTADUAL DO TOCANTINS**  
**CURSO DE SISTEMAS DE INFORMAÇÃO**

**QRAuth:**  
**SISTEMA DE GERENCIAMENTO DE IDENTIDADE UTILIZANDO QR**  
**CODE**

**CRISTIAN DEAN ABREU REGO**

Palmas – TO

2016

**UNIVERSIDADE ESTADUAL DO TOCANTINS**  
**CURSO DE SISTEMAS DE INFORMAÇÃO**

**QRAuth:**  
**SISTEMA DE GERENCIAMENTO DE IDENTIDADE UTILIZANDO QR  
CODE**

**Autor: CRISTIAN DEAN ABREU REGO**

Trabalho de Conclusão de Curso de  
Sistemas de Informação da Fundação  
Universidade do Tocantins, apresentado  
como parte dos requisitos para a  
obtenção do grau de Bacharel em  
Sistemas de Informação.

Palmas – TO

2016

## **AGRADECIMENTOS**

Primeiramente agradeço a Deus por ter me proporcionado a oportunidade de ter chegado até aqui e ter me acompanhado em cada momento dentro e fora da minha vida universitária.

Agradeço toda a minha família, em especial meus pais e minha irmã que por muitas vezes foram pacientes e me ajudaram sempre me dando a força que eu necessitava para passar por todos os obstáculos.

Aos meus amigos, meus mais sinceros agradecimentos por sempre entenderem o meu “Não posso, tenho prova amanhã”.

Agradeço também ao meu orientador Alex Coelho, toda a equipe de professores da UNITINS e meus colegas e amigos que sempre acreditaram no meu potencial.

*“Não há nenhuma maneira fácil de contornar isso. Não importa o quão talentoso você é seu talento vai falhar se você não está qualificado, se não tiver habilidade, se você não estudar, se você não trabalhar duro e dedicar-se a ser melhor a cada dia” (Will Smith).*

## **LISTA DE ABREVIATURAS E SIGLAS**

API	Application Programming Interface
QR CODE	Quick Response Code
ACID	Atomicidade, Consistência, Isolamento e Durabilidade
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TSL	Transport Layer Security
REST	Representational State Transfer
SOAP	Simple Object Access Protocol
WSDL	Web Services Description Language
HTTP	Hypertext Transfer Protocol
SSO	Single Sign-On

## LISTA DE ILUSTRAÇÕES

<i>Figura 1- Incidentes reportados ao CERT.br por ano desde 1999.....</i>	<i>15</i>
<i>Figura 2 - Serviços de segurança relativos à mensagem ou à entidade .....</i>	<i>16</i>
<i>Figura 3 - Verificação de Integridade .....</i>	<i>18</i>
<i>Figura 4 – Comparação do protocolo TCP/IP e o TCP/IP com TSL/SSL .....</i>	<i>20</i>
<i>Figura 5 - Relatório gráfico de smartphones vendidos em 2014 .....</i>	<i>22</i>
<i>Figura 6 - Camadas da arquitetura Android.....</i>	<i>24</i>
<i>Figura 7 - Códigos de Barras 1D e 2D .....</i>	<i>26</i>
<i>Figura 8 - Áreas do QR Code.....</i>	<i>28</i>
<i>Figura 9 - Composição do protocolo SOAP.....</i>	<i>29</i>
<i>Figura 10 - Ciclo de requisição de identidade em um SGI Fonte: do autor, 2016 .....</i>	<i>33</i>
<i>Figura 11 - Esquematização de um sistema que mantém o controle de identidade de usuários integrado .....</i>	<i>33</i>
<i>Figura 12 - Modelo triangular do Single Sign-On Fonte: (SUN, POSPISIL, et al., 2012) .....</i>	<i>34</i>
<i>Figura 13 - Fluxo abstrato do protocolo OAuth 2.0 Fonte (D. HARDT, 2012) .....</i>	<i>37</i>
<i>Figura 14 - Desenho metodológico da pesquisa.....</i>	<i>40</i>
<i>Figura 15 - Solução proposta para o trabalho.....</i>	<i>42</i>
<i>Figura 16 - Diagrama de classes.....</i>	<i>45</i>
<i>Figura 17 - Tela de login .....</i>	<i>46</i>
<i>Figura 18 - Página inicial da aplicação QRAuth.....</i>	<i>51</i>
<i>Figura 19 - Página de alteração de aplicações .....</i>	<i>52</i>
<i>Figura 20 - Página de alteração de dispositivos .....</i>	<i>53</i>
<i>Figura 21 - Interface de login da aplicação mobile .....</i>	<i>53</i>
<i>Figura 22 - Interface inicial da aplicação mobile.....</i>	<i>54</i>
<i>Figura 23 - Smartphone realizando a leitura do QR Code na interface de login .....</i>	<i>54</i>
<i>Figura 24 - Página de solicitação de vínculo do usuário.....</i>	<i>55</i>

# SUMÁRIO

<b>1 INTRODUÇÃO .....</b>	<b>10</b>
<b>1.1 OBJETIVOS .....</b>	<b>11</b>
1.1.1 OBJETIVO GERAL .....	11
1.1.2 OBJETIVOS ESPECÍFICOS .....	11
<b>2 REFERENCIAL TEÓRICO.....</b>	<b>13</b>
<b>2.1 SEGURANÇA.....</b>	<b>13</b>
2.1.1 SEGURANÇA DA INFORMAÇÃO.....	13
<b>2.2 ELEMENTOS DE SEGURANÇA .....</b>	<b>15</b>
2.2.1 SEGURANÇA DE REDE .....	15
2.2.2 HASH E SENHAS .....	17
2.2.3 CRIPTOGRAFIA.....	19
<b>2.3 DISPOSITIVOS MÓVEIS .....</b>	<b>21</b>
<b>2.4 CÓDIGO DE BARRAS .....</b>	<b>25</b>
2.4.1 QR CODE.....	26
<b>2.5 WEB SERVICES.....</b>	<b>28</b>
<b>2.6 SISTEMAS DE GERENCIAMENTO DE IDENTIDADE.....</b>	<b>32</b>
2.6.1 FERRAMENTAS E TRABALHOS CORRELATOS.....	36
<b>3 METODOLOGIA.....</b>	<b>39</b>
<b>3.1 DESENHO DE ESTUDO.....</b>	<b>39</b>
<b>3.2 DESENHO METODOLÓGICO .....</b>	<b>39</b>
<b>3.3 MATERIAIS E MÉTODOS .....</b>	<b>41</b>
3.3.1 BASE DE DADOS .....	42
3.3.2 IDENTITY PROVIDER (IdP).....	43
3.3.3 RELYING PARTY (RP) .....	44
3.3.4 COMPUTADOR / SMARTPHONE DO USUÁRIO .....	44
<b>4 RESULTADOS .....</b>	<b>45</b>
<b>4.1 MODELAGEM.....</b>	<b>45</b>
<b>4.2 APRESENTAÇÃO DA APLICAÇÃO .....</b>	<b>46</b>
4.2.1 QR CODE .....	46
4.2.1 WEB SERVICES .....	48

4.2.1 SOFTWARE WEB E MOBILE.....	51
4.3 TESTES.....	55
<b>5 CONCLUSÃO .....</b>	<b>61</b>
<b>4 REFERÊNCIAS.....</b>	<b>63</b>



## **RESUMO**

A necessidade de identificação do usuário perante as mais diversas aplicações existentes na web pode trazer uma série de problemas ligados a questões relacionadas à segurança, visto que senhas textuais são inseguras, além de dificuldades de memorização. Assim, este trabalho teve por objetivo o desenvolvimento de um software que visa gerenciar a identidade do usuário e permitir sua autenticação em plataformas de terceiros por meio de seu smartphone utilizando estruturas gráficas conhecidas como QR Code.

**Palavras-chave:** Gerenciamento de Identidade, Senhas, QR Code.

## **ABSTRACT**

The need to identify the user before the various existing web applications can bring a lot of problems related to security issues, seeing that textual passwords are insecure, besides memorizing difficulties. This work aimed to development of a software that aims to manage the user's identity and allow your login on third party platforms through your smartphone using known graphic structures, as QR Code.

**Keywords:** Identity Management, Passwords, QR Code.

## 1 INTRODUÇÃO

Nos últimos anos a internet vem se tornando cada vez mais presente na vida dos brasileiros. Quase metade da população, cerca de 48%, a utiliza para algum fim. Nesse rumo, o uso de aparelhos celulares como meio para acessar a internet já compete com computadores e notebooks, 66% e 71%, respectivamente. Esse resultado é decorrente da influência das redes sociais que conectam cerca de 92% dos internautas (BRASIL, 2015).

Uma das consequências desse crescimento é a necessidade do usuário identificar-se, frequentemente, através das mais diversas aplicações disponíveis na rede mundial de computadores. Embora sejam passíveis de esquecimento e possuam problemas de vulnerabilidade, a utilização de senhas textuais ainda consiste no modelo de autenticação mais utilizado (SILVA, 2007).

Visando encontrar soluções para os problemas de utilização de senhas relacionados à memória dos usuários, a utilização de sistemas de gerenciamento de identidade torna-se cada vez mais frequente.

O gerenciamento de identidade consiste no processo em que entidades são representadas e reconhecidas como identidades digitais. Em um modelo centralizado, é possível que diversas aplicações utilizem um único provedor de identidade para realizar a autenticação dos usuários, carecendo apenas de uma relação de confiança entre ambos (JøSANG, FABRE, *et al.*, 2005).

É comum encontrar *websites* que fazem o uso de ferramentas de terceiros para permitir a autenticação do usuário na plataforma. Dentre as ferramentas mais conhecidas, destacam-se as de redes sociais, como o *Facebook Login*<sup>1</sup>, *Google Sign-In for Google+*<sup>2</sup> e *Twitter Sign in*<sup>3</sup>. Mesmo se tratando de *plugins* seguros, deixam a desejar quando o assunto é privacidade, isso pelo fato de muitos deles utilizarem técnicas de *web tracking*. O termo *web tracking* diz respeito à prática onde *sites* coletam informações sobre usuários e suas atividades de navegação em diferentes *web sites* (ROESNER, KOHNO e WETHERALL, 2012).

---

<sup>1</sup> Facebook for developers. Disponível em: <<https://goo.gl/ie8yIL>> Acesso em 23 mar. 2016.

<sup>2</sup> Google Sign-In for Google+. Disponível em: <<https://goo.gl/eGD4tB>> Acesso em 23 mar. 2016.

<sup>3</sup> Twitter Sign in. Disponível em: <<https://goo.gl/zfESMu>> Acesso em 23 mar. 2016.

Este trabalho teve como propósito a elaboração de uma solução que visa centralizar as credenciais do usuário e permitir seu acesso em diversas plataformas e aplicações desenvolvidas por terceiros.

Como diferencial de tantas outras ferramentas existentes no mercado, o software desenvolvido simplifica o processo de autenticação, permitindo que o usuário seja identificado por determinada aplicação dispensando o uso de senhas textuais. Todo o processo será realizado de forma automática através da leitura de um código de barras 2D denominado *QR Code* (descrito na Sessão 2.4.1 *QR CODE* do Referencial Teórico) por meio do smartphone do usuário.

Com o intuito de permitir a comunicação entre as aplicações envolvidas utilizou-se de diversos conceitos de *Service-Oriented Architecture (SOA)* que serão abordados posteriormente. Além disso, a necessidade de se atentar a questões de segurança foi essencial para o andamento deste trabalho.

## **1.1 OBJETIVOS**

Este tópico visa estabelecer os objetivos geral e específicos propostos para trabalho.

### **1.1.1 OBJETIVO GERAL**

Desenvolvimento de uma aplicação que tem como foco gerir informações básicas de identificação do usuário de forma centralizada, garantindo uma melhor experiência na autenticação em *softwares* de terceiros. A ferramenta também possibilitará através de um módulo *mobile* a leitura de elementos *QR Code* para a liberação imediata das credenciais para as aplicações requisitantes, evitando assim, a utilização de senhas textuais.

### **1.1.2 OBJETIVOS ESPECÍFICOS**

São objetivos específicos da proposta:

- Desenvolver aptidão tecnológica quanto aos principais componentes e estruturas para o desenvolvimento *Service-Oriented Architecture (SOA)* e para dispositivos móveis;
- Promover a criação de um *web service* para a realização das requisições e consumo dos dados;
- Desenvolver uma funcionalidade que permita o usuário administrar e revogar suas credenciais disponibilizadas a sistemas de terceiros;
- Permitir que o usuário possa se autenticar uma única vez através da leitura de um *QR Code* por meio de dispositivos móveis e possibilitar que sua sessão fique disponível para todas as aplicações, nas quais o mesmo possua um vínculo ativo.

Para que fosse possível o desenvolvimento deste trabalho de conclusão, fez-se necessário a utilização de uma fundamentação teórica, disposta no próximo tópico, que abordará conceitos e técnicas acerca dos temas aqui evidenciados.

No decorrer deste trabalho serão apresentados o Referencial Teórico (2), Metodologia (3), Resultados (4) e Conclusão (5).

## **2 REFERENCIAL TEÓRICO**

O objetivo deste capítulo é apresentar o referencial teórico, descrevendo em tópicos e subtópicos os principais conceitos e definições dos assuntos envolvidos neste trabalho.

### **2.1 SEGURANÇA**

Sistemas e métodos de segurança podem ser descritos como fortes e fracos. Uma analogia que pode ser feita é utilizando o número de cartões de créditos, que oferecem por si só apenas uma defesa fraca, contra repúdio. No entanto, o cartão de crédito pode ter uma defesa considerada forte, se levarmos em consideração a utilização da senha, permitindo o usuário deixar evidência da sua presença por meio de uma assinatura (O'GORMAN, 2004).

Diversas características podem ser verificadas quando se fala de segurança e da manipulação de recursos computacionais e tecnológicos. Devido a proporção das atividades desempenhadas, tais aspectos tomam maior visibilidade. Por isso, uma categoria em específico tem crescido com o decorrer dos anos. Trata-se, neste caso, da segurança da informação abordada a seguir.

#### **2.1.1 SEGURANÇA DA INFORMAÇÃO**

Para que se possa entender a importância da Segurança da Informação, deve-se, necessariamente, compreender o verdadeiro sentido do termo Informação. A questão é que não há um conceito absoluto quando se trata de informação (WEIZSÄCKER, 1985 apud CAPURRO, 2007).

Le Coadic (1996) aponta que a informação é um conhecimento que pode ser inscrito de alguma forma, seja ela escrita, oral ou mesmo audiovisual. Para Dretske (1981) a informação não necessita de um processo de interpretação, porém esse processo é necessário para tornar a aprendizagem possível e assim atingir o conhecimento.

Na sociedade pós-industrial em que vivemos, a informação vem sendo considerada um capital precioso. O sucesso desejado pelas organizações está diretamente ligado às informações que as mesmas possuem em conjunto com a forma de gerir e aproveitá-las (MORESI, 2000).

Em todas as etapas de existência de uma empresa, independente de seu ramo, *core business* e porte, elas sempre gozaram da informação na intenção de obter uma melhor produtividade, diminuir custos, ganho de *market share*, aumento de agilidade, competitividade e uma base para a tomada de decisões (SÊMOLA, 2014).

Percebe-se que a segurança da informação deve ser tratada com seriedade, independente do cenário, mas quando as informações estão presentes na rede o cuidado deve ser maior. Na última década a utilização da internet cresceu consideravelmente, de 15% os valores hoje ultrapassam 40% da população mundial. Todo esse processo de digitalização das informações consiste em uma oportunidade tentadora para que os ladrões virtuais possam atacar (MCAFEE, 2015).

Segundo Stallings (2010, p. 8, tradução nossa): “O campo de segurança de rede e de internet consiste de medidas para cessar, prevenir, detectar e corrigir violações de segurança que envolvam a transmissão de informação”.

Dentre as diversas normas existentes sobre segurança, uma se destaca, a ISO 17799 - “Código de boas práticas para a gestão da segurança da informação”, que propõe uma série de recomendações destinadas às pessoas responsáveis pela definição, implementação ou manutenção da segurança da informação. Esta norma define a **segurança da informação** como a proteção da **disponibilidade**, **integridade** e **confidenciabilidade** da informação escrita, falada ou em formato digital - (TASHI e GHERNAOUTI-HÉLIE , 2007).

A partir do exposto, pode se considerar tamanha importância da informação assim como sua segurança. No próximo tópico serão abordados elementos de segurança, para que se possa compreender os cuidados e as técnicas mais adequadas para armazenar e trafegar informações.

## 2.2 ELEMENTOS DE SEGURANÇA

Importante ponto da discussão passa pela apresentação dos conceitos sobre elementos de segurança, assim como sua relevância ao se desenvolver aplicações. Desse modo, é fundamental a realização de considerações acerca de alguns destes conceitos, iniciando pela segurança de rede.

### 2.2.1 SEGURANÇA DE REDE

Com a difusão da internet iniciou-se uma grande preocupação com a área de segurança de redes. De acordo com CERT.br (2015), os anos de 2014 e 2015 tiveram o maior número de incidentes reportados na história. Juntos chegaram a ultrapassar a marca de 1,7 milhões, como apresenta a Figura 1, lembrando que são apenas os ataques notificados espontaneamente ao CERT.br.

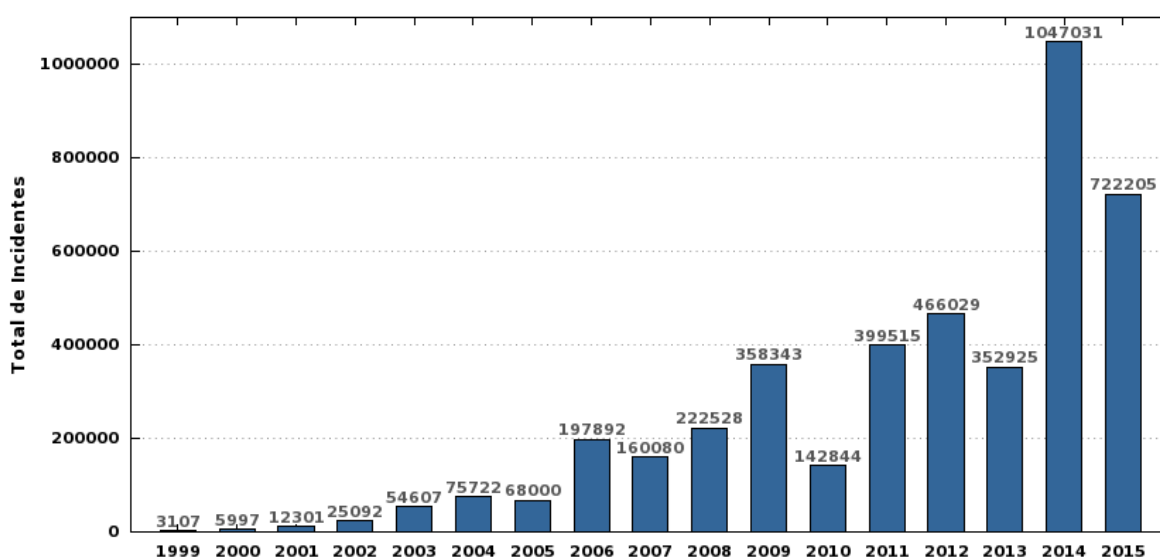


Figura 1- Incidentes reportados ao CERT.br por ano desde 1999

Fonte: (CERT.br, 2015)

Segundo Forouzan (2008), são cinco os serviços que podem ser fornecidos pela segurança de rede (apresentados pela Figura 2). Quatro deles estão relacionados à troca de mensagens pela rede: Confidencialidade, Integridade, Autenticação e Não Repúdio, o quinto relaciona-se a Identificação de Entidades.

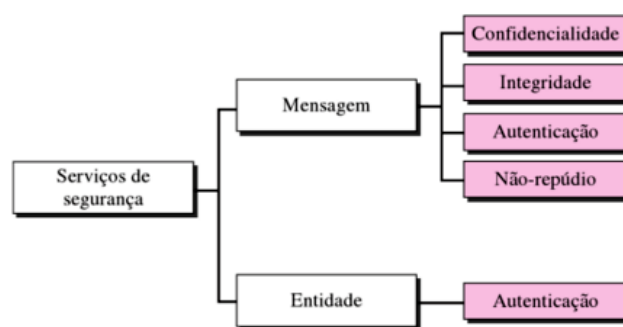


Figura 2 - Serviços de segurança relativos à mensagem ou à entidade  
Fonte: (FOROUZAN, 2008)

A seguir as descrições de cada um dos serviços (FOROUZAN, 2008):

- **Confidencialidade da Mensagem:** Tanto o emissor quanto o receptor da mensagem esperam sigilo, ou seja, a mensagem deve fazer sentido apenas para o receptor pretendido.
- **Integridade da Mensagem:** A mensagem não deve sofrer nenhuma alteração durante a transmissão, seja acidental ou mal-intencionada.
- **Autenticação de Mensagens:** O receptor precisa estar certo da identidade do emissor para evitar o recebimento de mensagens de impostores.
- **Não repúdio de Mensagens:** É uma garantia de que o emissor não pode recusar uma mensagem que ele, de fato, enviou.
- **Autenticação de Entidades:** Qualquer usuário, para ter acesso aos recursos do sistema, deve ter sua identidade verificada previamente.

Devido a extrema facilidade de conexão e o benefício de dispensar o uso de cabos, a utilização de redes sem fio como meio de acesso à internet tem se tornado cada vez mais comum. Embora os lados positivos das redes sem fio sejam muitos, há uma enorme facilidade de obter acesso às cópias dos pacotes trafegados por este meio (KUROSE e ROSS, 2009).

Por sistemas *wireless* muitas vezes alcançarem espaços além do perímetro necessário, intrusos podem se beneficiar de ferramentas de análise de redes, popularmente conhecidas como *sniffers*, para se apoderar de dados confidenciais



até mesmo fora do ambiente da organização, como em estacionamentos ou locais próximos (AMARAL e MAESTRELLI, 2004).

Na linha refletida, a necessidade da utilização de técnicas de autenticação se torna indispensável para a segurança do usuário. Assim, na próxima seção serão abordados conceitos acerca de *Hash* e Senhas que são dois atributos essenciais para a validação da autenticidade de usuários.

### 2.2.2 HASH E SENHAS

O termo senha é usado para se referir a palavras, frases ou *PINs*<sup>4</sup> que são mantidos estritamente em segredo para realizar autenticação. O principal problema da utilização de senhas está associado ao fato de que geralmente, as de fácil memorização podem ser descobertas por *hackers* sem muito esforço, todavia, a utilização de senhas mais complexas faz com que o usuário tenha mais dificuldade de decorá-las (O’GORMAN, 2004).

Na tradicional autenticação de senhas, cada usuário possui um identificador (ID) e uma senha (PW<sup>5</sup>) armazenados normalmente em uma tabela no servidor. Sempre que um usuário desejar se autenticar na aplicação, deverá submeter seu ID e PW para o servidor. Uma vez encontrado na tabela um registro correspondente ao par (ID e PW) indicado pelo usuário, o mesmo terá acesso as funcionalidades da aplicação (LIAO, LEE e HWANG, 2005).

Segundo Lamport (1981) há três maneiras de um intruso aprender a senha do usuário e se passar por ele interagindo com o sistema, estas serão descritas a seguir:

- Obtendo acesso a informações armazenadas no sistema.
- Através da interceptação dos dados do usuário provenientes da comunicação com o sistema.
- Pela divulgação acidental da senha do usuário.

---

<sup>4</sup> *Personal Identification Number* ou Números de identificação pessoal.

<sup>5</sup> Acrônimo de *password*.

O primeiro item da lista pode ter sua segurança aperfeiçoada utilizando os conceitos de *hash*. Para Garbe (2015, p. 85) *hash* é uma operação de via única, ou seja, que não pode ser desfeita, pelo menos não facilmente. De maneira mais técnica, Lamport (1981) entende *hash* como sendo uma função de mão única, de modo que possa ser mapeada por  $F$  onde:

1. Dada uma palavra qualquer  $x$ , esta é facilmente computada por  $F(x)$ ;
2. Dada uma palavra  $y$ , não é viável calcular  $x$ , tal que  $y = F(x)$ .

O armazenamento do *DIGEST*, termo que diz respeito ao retorno de uma função *hash*, ao invés da senha propriamente dita, pode dificultar o trabalho de um possível invasor que venha a ter acesso as senhas armazenadas na aplicação (o esquema pode ser conferido na Figura 3). Contudo, na internet é possível encontrar listas contendo senhas conhecidas e seus respectivos *digests*, estas são conhecidas como *Rainbow Tables*. Todavia, este problema tem uma solução relativamente simples, que é adição de um *salt* no cálculo do *hash*, que nada mais é que uma *string*, normalmente aleatória, que tem como objetivo aumentar a complexidade da senha (GARBE, 2015).

A Figura 3 representa uma exemplificação do conceito de *hash*, onde um emissor (Alice) utiliza uma função para gerar um *digest* de uma mensagem ou documento. Do outro lado Bob utiliza da mesma função para gerar outro *digest* de uma cópia da mensagem ou documento que o mesmo tem armazenado. A partir daí é possível realizar a comparação dos *digests* e verificar sua autenticidade.

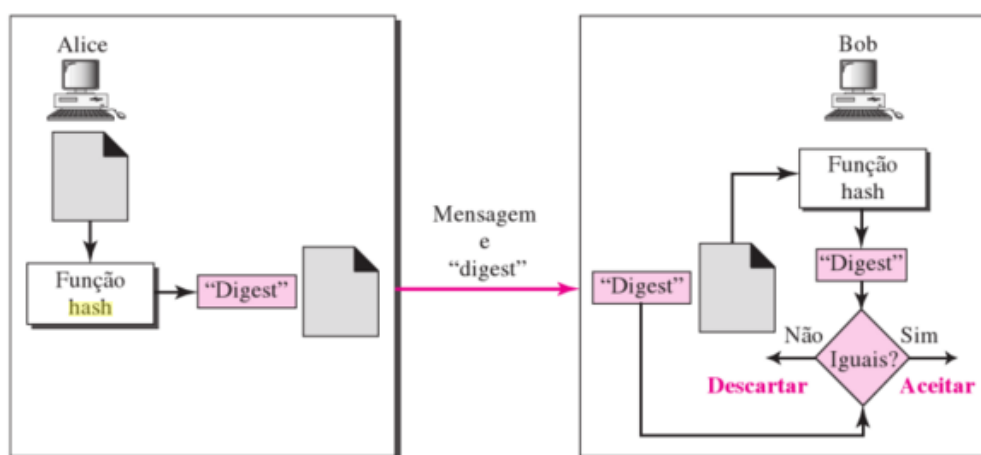


Figura 3 - Verificação de Integridade  
Fonte: (FOROUZAN, 2008)

É importante frisar que funções *hash* não podem ser consideradas criptografia. Isso se deve ao fato de seu processo ser irreversível. Tendo em vista estes conceitos aqui abordados, pode-se afirmar que o *hash* trata-se apenas de uma assinatura. Portanto, sua principal função é prover a Integridade (GARBE, 2015).

Em razão da importância de utilização de senhas para se comprovar a autenticidade e das funções *hash* para a confirmação da integridade, faz-se necessário, por complementação, a utilização de técnicas que visam a proteção dos dados de forma que apenas usuários autorizados possam ter acesso ao conteúdo. Assim sendo, na próxima seção serão abordadas técnicas e conceitos acerca de criptografia.

### 2.2.3 CRIPTOGRAFIA

A criptografia está diretamente ligada a área de segurança da informação. De acordo com Kurose e Ross (2009, p. 495):

Técnicas criptográficas permitem que um remetente disfarce os dados de modo que um intruso não consiga obter nenhuma informação dos dados interceptados. O destinatário, é claro, deve estar habilitado a recuperar os dados originais a partir dos dados disfarçados.

Embora a criptografia se enquadre na categoria de requisito não funcional, por comumente não estar associada às finalidades da aplicação, o crescente aumento do uso destas para realizar a troca de informações fez com que a procura por serviços de segurança tenha aumentado consideravelmente (BRAGA, 1999).

Os modelos de criptografia são diferenciados pelo conjunto de técnicas que utilizam. Essas técnicas são baseadas em chaves criptografadas. As mais comuns são as simétricas e assimétricas, descritas a seguir (GALVÃO, 2015).

- Chave simétrica: Neste modelo também conhecido como criptografia de chave única, a mesma chave é utilizada para cifrar e decifrar informações, como exemplo temos os algoritmos: *DES*, *RC*, *AES*.
- Chave assimétrica: Também conhecido como criptografia de chave pública, utiliza-se duas chaves distintas, uma delas é usada para cifrar e outra para decifrar os dados. O algoritmo ainda permite a divulgação

da chave pública para que qualquer um possa cifrar os dados, contudo, apenas o detentor da chave privada terá acesso aos mesmos.

Dentre os métodos de criptografia utilizando chave assimétrica, o mais conhecido e utilizado é o *RSA*. Para implementá-lo são necessários dois números primos  $p$  e  $q$ , onde o produto destes serão utilizados para se chegar em  $n$ , que é a chave pública. A segurança do *RSA* depende de que os números primos (chave privada), sejam mantidos em segredo e que somente o receptor da mensagem tenha acesso. Cada um dos caracteres da mensagem são convertidos em seu respectivo valor numérico da tabela *ASCII*, que em seguida é dividido em blocos menores do que o valor da chave pública  $n$  (SILVEIRA, 2009).

A utilização de protocolos de segurança na internet é de grande importância. Sobretudo, quando os dados trafegados exigem que a privacidade do usuário não seja violada, como é o caso de compras online, mensagens privadas entre outros.

O *Secure Sockets Layer (SSL)* e o *Transport Layer Security (TLS)* são dois protocolos que visam proporcionar a privacidade e a confiabilidade entre dois aplicativos que se comunicam. O *TSL* trata-se de uma evolução do *SSL*, ambos são protocolos que atuam entre a camada de transporte e aplicação do protocolo *TCP*, como pode ser conferido na Figura 4 (IETF, 2008).

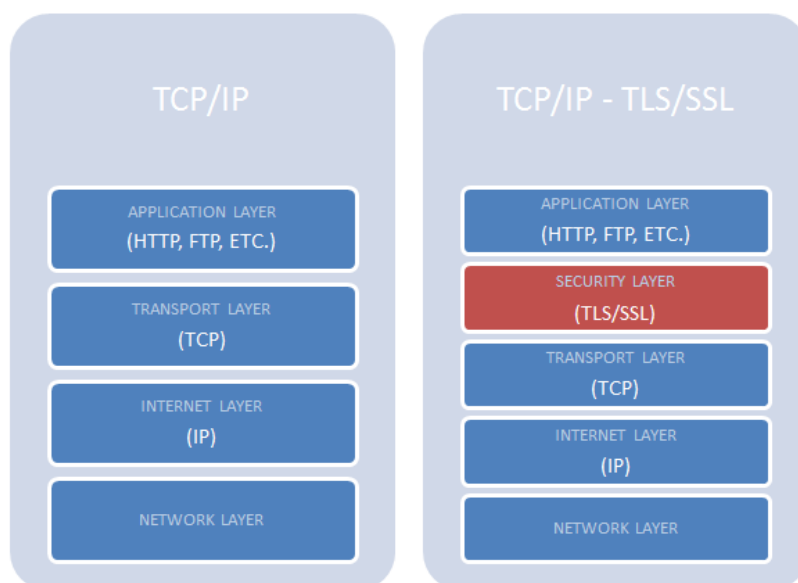


Figura 4 – Comparação do protocolo TCP/IP e o TCP/IP com TSL/SSL  
Fonte: (SLAVIERO, 2011)

A vantagem da utilização do *TLS/SSL* é o fato do protocolo ser projetado para combinar perfeitamente com a arquitetura *TCP/IP*, mantendo os princípios de design baseado em camadas de *IP* (SLAVIERO, 2011).

Os protocolos *SSL* e *TLS* visam um conjunto de metas. Estas estão listadas abaixo em ordem de prioridade (IETF, 2008):

1. Segurança criptográfica: Uma conexão segura deve ser estabelecida entre as duas partes.
2. Interoperabilidade: Programadores independentes devem ser capazes de desenvolver aplicações utilizando *SSL/TLS* de forma que possam trocar parâmetros de criptografia sem a necessidade de conhecer o código do outro.
3. Extensibilidade: O *SSL/TSL* deve prover uma estrutura de forma que possam ser incorporados novos métodos de criptografia sem a necessidade de implementar uma nova biblioteca de segurança.
4. Eficiência Relativa: Devido ao uso intensivo de *CPU* que operações criptográficas necessitam. O protocolo *SSL/TSL* incorporou um sistema de cache de sessão opcional, que visa reduzir o número de ligações estabelecidas.

Embora as metas tenham como objetivo assegurar diversos benefícios ao *TLS/SSL*, uma camada de segurança extra possui desvantagens. No quesito desempenho da aplicação, a utilização de criptografia possui uma influência significativa. Em vista disso, a meta Eficiência Relativa visa reduzir esse gargalo.

Considerando os indicadores das abordagens acerca de criptografia e segurança realizadas neste tópico, é importante salientar que a utilização destas técnicas em aplicações *mobile* é muitas vezes necessária. Desse modo, na próxima seção serão abordados conceitos e estatísticas acerca de dispositivos móveis.

## 2.3 DISPOSITIVOS MÓVEIS

A utilização de dispositivos móveis para o acesso à internet vem crescendo cada vez mais ao longo dos anos. Segundo o IBGE (2016), o percentual entre os domicílios que acessaram a internet entre 2013 e 2014 por meio de um

microcomputador recuou de 88,4% para 76,6%. Enquanto isso, aqueles que utilizavam o celular como meio de acesso subiu de 53,6% para 80,4%.

O aumento da utilização destes dispositivos, faz com que a concorrência entre os sistemas operacionais mobile acabem aumentando. O IDC (2015) relata que em 2014, o sistema operacional *Android* estava presente em 81,5% dos dispositivos móveis, seguido do iOS com 14,8% e outros menos utilizados como pode ser observado no gráfico presente na Figura 5.

Neste trabalho o foco será dado aos dispositivos *Android*, devido a sua considerável liderança em números de usuários.

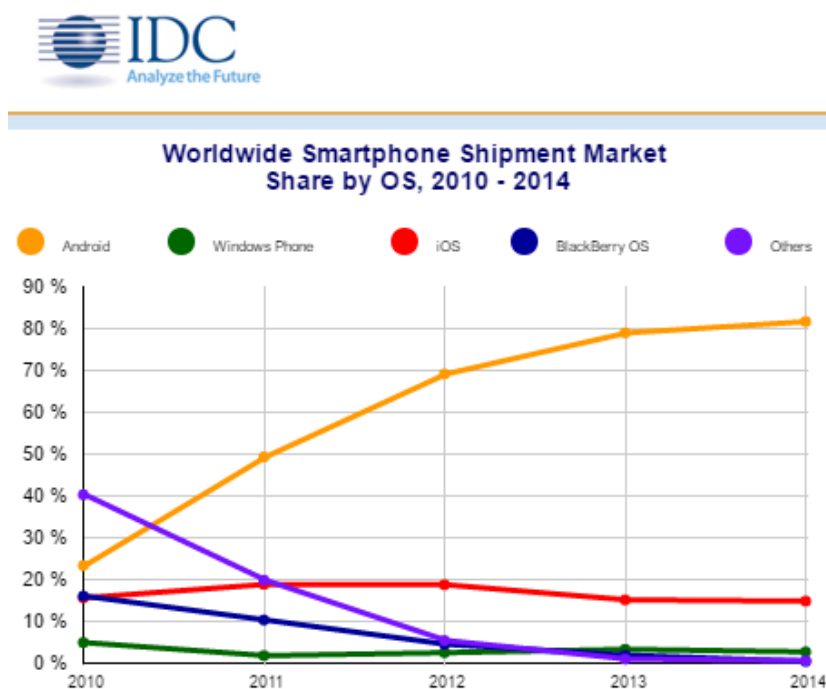


Figura 5 - Relatório gráfico de smartphones vendidos em 2014  
Fonte: (IDC, 2015)

O *Android* é um *Software Stack*<sup>6</sup> de código aberto criado por um grupo de empresas conhecido como *Open Handset Alliance*, liderada pelo Google. A arquitetura *Android* é baseada em camadas, como pode ser visto na Figura 6 (ANDROID, 2016).

<sup>6</sup> Conjunto de aplicações que trabalham em conjunto para produzir um resultado em comum.

A seguir, observa-se a descrição de cada uma das camadas da arquitetura *Android* (ANDROID, 2016):

- *Linux Kernel*: É nesta primeira camada onde se encontram os *drivers* do dispositivo. O *Android* utiliza uma versão do *kernel* do Linux com algumas modificações especiais, como o controle de memória mais agressivo, permitindo uma melhor performance em dispositivos com um *hardware* mais modestos.
- *Hardware abstraction layer (HAL)*: Nesta camada de abstração, é definida uma interface padrão para fornecedores de *hardware*. Além disso, a *HAL* permite que sejam implementadas funcionalidades sem interferir no sistema de nível superior.
- *Android System Services*: Nesta camada se localizam as *Application Programming Interface (APIs)* responsáveis pela comunicação com os serviços do sistema e acesso ao *hardware* subjacente.
- *Binder IPC*: Este é nível responsável por promover a comunicação entre os aplicativos e os processos no *Android*, além de permitir a chamada de serviços do sistema.
- *Application Framework*: Geralmente a camada mais utilizada pelos desenvolvedores e onde ocorre as interações com o usuário. Nesta camada que encontramos os aplicativos, tais como: Alarme, navegador, calculadora, calendário, entre outros.

Conforme pôde ser visto, a utilização de diferentes níveis de abstração na plataforma favorece uma significativa organização no funcionamento do sistema como um todo. Para uma melhor compreensão acerca da arquitetura *Android*, a Figura 6 representa sua disposição em camadas.



Figura 6 - Camadas da arquitetura *Android*  
 Fonte: (ANDROID, 2016)

Para COSTA e FILHO (2013), a plataforma *Android* possui uma série de vantagens, dentre elas:

1. Uma plataforma adaptada para diversos tamanhos de tela.
2. Suporte nativo a banco de dados, no caso o *SQLite*.
3. Suporte a diversas tecnologias de conectividade.
4. O navegador tem como base o *WebKit*, que é um *framework* de código aberto.



5. As aplicações são desenvolvidas utilizando java e compiladas em *bytecodes Dalvik*. O Dalvik permite que independente do processador utilizado, as aplicações funcionem corretamente.
6. Suporta recursos de *hardware* adicionais, tais como: câmeras de vídeo, acelerômetros, *Global Positioning System (GPS)*, telas *touchscreen* e aceleração de gráficos 3D.
7. O *Android* possui um *Software Development Kit*<sup>7</sup> (SDK) bastante completo, com recursos como: ferramentas para *debugging*, memória e análise de desempenho.

Além das vantagens, Costa e Filho (2013) reforça algumas desvantagens sobre a plataforma, enumeradas a seguir:

1. As aplicações pagas são mais caras.
2. A carência de atualizações assim como os problemas gerados por elas.
3. Vírus na loja de aplicativos.
4. Devido ao grande número de versões do *Android* no mercado, a incompatibilidade entre as versões chega a ser um problema.

Desse modo, verifica-se que a utilização da plataforma *Android* possui mais vantagens do que desvantagens. Dentre as diversas utilidades dos *smartphones*, alguns conceitos de automação de tarefas, como a leitura de código de barras (abordados no próximo tópico) nestes dispositivos, podem facilitar a vida de muitos usuários.

## 2.4 CÓDIGO DE BARRAS

Na sociedade moderna, a automação acabou se tornando um processo fundamental e irreversível (SOARES e VASCONCELLOS, 1991).

Silveira e Lima (2003) definem a automação como:

[...] um conjunto de técnicas destinadas a tornar automáticas a realização de tarefas, substituindo o gasto de bio-energia humana, com esforço muscular e mental, por elementos eletromecânicos computáveis. [...]. Os

---

<sup>7</sup> Nome dado ao conjunto de ferramentas necessárias para o desenvolvimento de softwares/aplicativos para determinada arquitetura.

benefícios para qualquer processo automação são nítidos: eficiência, segurança, menor custo, maior produção, etc.

Os códigos de barras podem ser considerados como um dos sinais visíveis da automação. Estes estão cada vez mais presentes na sociedade e permitem que diversos processos, antes manuais, sejam agilizados (SOARES e VASCONCELLOS, 1991).

Por definição, os códigos de barras podem ser classificados em dois tipos: o unidimensional (1D), que utilizam um segmento de linhas de largura diferentes e espaços na representação dos dados, e o bidimensional (2D), utilizando símbolos empilhados e matrizes. Alguns exemplos podem ser conferidos na Figura 7 (CHUANG, HU e KO, 2010).

1D barcodes	<b>Code 39</b>  123456	<b>Code 128</b>  123456	<b>EAN-13</b>  1 234567 890128	<b>ISBN</b>  9 781234 567897
2D barcodes	<b>QR Code</b> 	<b>PDF417</b> 	<b>DataMatrix</b> 	<b>Maxi Code</b> 

Figura 7 - Códigos de Barras 1D e 2D  
Fonte: (CHUANG, HU e KO, 2010)

Como o foco deste trabalho é a utilização do *QR Code* para a resolução do problema em questão, faz-se necessário o estudo de suas vantagens e desvantagens perante seus concorrentes. Dessa forma, este será o tema abordado na seção seguinte.

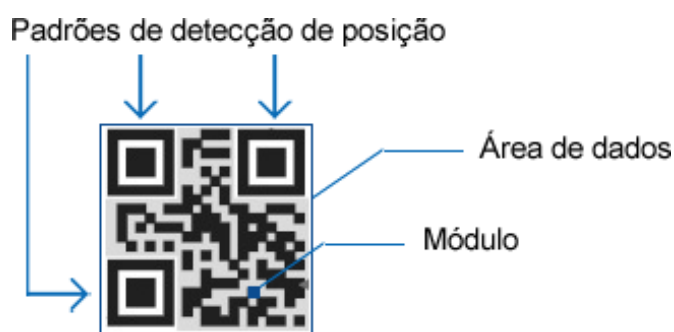
#### 2.4.1 QR CODE

O *QR Code* (*Quick Response*) trata-se de um código bidimensional (2D) com alta capacidade de armazenamento de informações codificadas (DENSO WAVE INCORPORATED, 2014).

Segundo Yabe (2011) a utilização do *QR Code* no Brasil só não se propagou tanto quanto no Japão devido a popularização tardia dos *smartphones*. Porém, isso tende a mudar com o decorrer do tempo.

A utilização do *QR Code* traz uma série de vantagens quando comparada a seus concorrentes. A seguir serão demonstradas algumas destas vantagens (DENSO WAVE INCORPORATED, 2014):

- Alta capacidade de codificação: Um *QR Code* pode guardar centena de vezes mais caracteres que um código de barras convencional.
- Pequeno tamanho de impressão: Como as informações estão dispostas em duas dimensões (vertical e horizontal), o *QR Code* é capaz de codificar dados em cerca de um décimo do espaço de um código de barras tradicional.
- Capacidade de codificação de caracteres: Com o *QR Code* é possível guardar caracteres de outros idiomas como o Kanji<sup>8</sup> e Kana<sup>9</sup> de forma eficiente.
- Resistência a danos e sujeira: O *QR Code* tem a capacidade de correção de erros, garantindo a restauração parcial ou total das informações mesmo que o código esteja danificado ou ilegível em até 30%.
- Legível em 360º: Possibilita a leitura independente de sua orientação devido aos padrões de detecção de posição (evidenciados na Figura 8).
- Recurso de incremento estruturado: Há um recurso que permite que determinada informação seja dividida e armazenada em vários *QR Code*.



<sup>8</sup> Ideogramas chineses utilizados por japoneses

<sup>9</sup> Caracteres de escrita silábica japonesa

Figura 8 - Áreas do QR Code  
Adaptado: (DENSO WAVE INCORPORATED, 2014)

Há três elementos que caracterizam um *QR Code* (evidenciados na Figura 8). Os padrões de posição garantem a leitura do código independente de sua orientação em relação ao leitor. Os módulos são cada um dos quadradinhos dispostos na área de dados (DENSO WAVE INCORPORATED, 2014).

A leitura de *QR Code* pode ser implementada em *smartphones* utilizando diversos aplicativos e bibliotecas, como: BeeTagg<sup>10</sup>, Kaywa<sup>11</sup>, Zxing<sup>12</sup> entre outras.

Como pôde ser visto neste capítulo, são muitos os benefícios do *QR Code*. Sua associação com outras tecnologias, como o *Web Service* pode ser muito satisfatória, e em vista disso, no próximo tópico serão estudados estes conceitos.

## 2.5 WEB SERVICES

Para MENÉNDEZ (2002) um *Web Service* tem a função de permitir que aplicações interajam entre si sem a necessidade da intervenção humana. KAO (2001, tradução nossa) sustenta que *Web Service* “[...] é uma aplicação que aceita requisições de outros sistemas através da internet ou intranet [...]”.

Algumas tecnologias anteriores como *Remote Method Invocation (RMI)*, *Common Object Request Broker Architecture (CORBA)* e *Distributed Component Object Model (DCOM)* já foram bastante utilizadas na integração de sistemas distribuídos, o problema é que essas tecnologias dependem de uma certa compatibilidade entre o cliente e o servidor (MUMBAIKAR e PADIYA, 2013).

Os *Web Services* surgiram com o objetivo de buscar a interoperabilidade entre diferentes sistemas (MARTINS, 2007). Para que isso ocorra é necessário que haja padronização, e em vista disso houve a necessidade da criação de protocolos, como o *SOAP* e o *REST*.

<sup>10</sup> QR Generator by BeeTag. Disponível em: <<http://goo.gl/vPo0UC>>. Acesso em 23 mai. 2016.

<sup>11</sup> Free QR Code Generator, Coupon, Contact & Design QR Codes & Tracking. Disponível em: <<https://goo.gl/VvI21q>>. Acesso em 23 mai. 2016.

<sup>12</sup> Disponível Oficial ZXing ("Zebra Crossing") project home. GitHub. Disponível em: <<https://goo.gl/N5mIzB>>. Acesso em 23 mai. 2016.

O *SOAP* ou *Simple Object Access Protocol* é um tipo de extensão do protocolo *HTTP* que permite a comunicação entre um emissor e receptor através de um conjunto de padrões em comum. Todo o tráfego dos dados é realizado no formato *XML* (*Extensible Markup Language*) e necessita seguir a descrição *WSDL*<sup>13</sup> (NEWCOMER, 2002).

As requisições no protocolo *SOAP* precisam estar contidas em uma estrutura denominada Mensagem e inseridas dentro de um Envelope (demonstrado na Figura 9), que por sua vez deve ser formalizado de acordo com a inspeção *WSDL* do serviço.

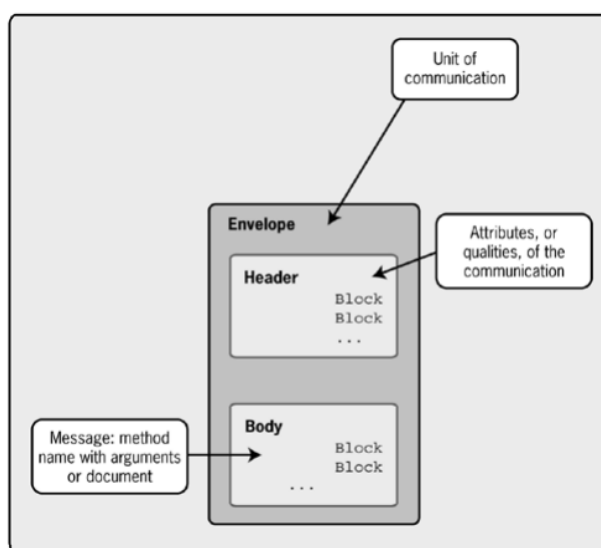


Figura 9 - Composição do protocolo SOAP  
Fonte: (NEWCOMER, 2002)

Cada mensagem *SOAP* pode ser dividida em três partes, sendo elas: *Envelope*, *Header* e *Body*, descritas a seguir (SILVA e GONÇALVES, 2012):

- *Envelope*: Refere-se à *tag* que envolve toda a mensagem, delimitando seu início e fim.
- *Header*: Esta é a área responsável por guardar informações adicionais, tais como: dados de autenticação, assinatura digital, autorização, etc.

<sup>13</sup> *WSDL* ou *Web Services Description Language* é um documento que tem como função descrever um determinado *web service*, especificando o local e os métodos utilizados pelo serviço. XML WSDL Disponível em: <<http://goo.gl/CcEKJW>>. Acesso em 28 abr. 2016.

- *Body*: Seção onde fica armazenado o corpo da mensagem, conforme descrito no *WSDL*. Além disso, o *body* também é responsável por guardar o nome da operação solicitada e seus respectivos parâmetros.

Embora o *SOAP* seja um padrão bem estabelecido no mercado e conte com o apoio de diversas empresas do ramo tecnológico tais como *IBM*, *Oracle* e *Microsoft*, por se tratar de um protocolo extenso e com diversas especificações, é praticamente impossível sua utilização sem o apoio de ferramentas auxiliares (SILVA e GONÇALVES, 2012).

Conforme foi apresentado, não há questionamentos quanto ao poderio do *SOAP* em aplicações distribuídas. Todavia, há uma alternativa bastante popular que conta com algumas vantagens, o *REST*.

Para Fielding (2000, tradução nossa) o conceito de *REST* (*Representational State Transfer*) aplica-se a:

[...] uma abstração da arquitetura de elementos dentro de um sistema de distribuído de hipermídia. O *REST* ignora os detalhes da implementação do componente e da sintaxe do protocolo a fim de se concentrar nos papéis dos componentes, nas restrições sobre a sua interação com outros componentes e sua interpretação de elementos de dados significativos.

A arquitetura *REST* foi construída com o intuito de se obter um modelo para a construção de sistemas de hipermídia distribuídos em larga escala. Por se tratar de um conceito bastante abstrato, os princípios do *REST* foram utilizados para explicar a escalabilidade de protocolos como o *HTTP* 1.0 e *HTTP* 1.1 (PUTASSO, ZIMMERMANN e LEYMAN, 2008).

Segundo FIELDING (2000), a arquitetura *REST* pode ser definida por um conjunto de quatro princípios, listados na Tabela 1 - Restrições de interface da arquitetura *REST*.

Tabela 1 - Restrições de interface da arquitetura *REST*

Restrição	Descrição
<i>Identification of resources</i>	O principal objetivo da identificação de recursos na arquitetura <i>REST</i> é identificar determinado recurso

	envolvido em uma interação entre componentes. Para isso o <i>REST</i> utiliza o padrão <i>URI</i> <sup>14</sup> .
<i>Manipulation of resources through these representations</i>	Com os recursos já identificados, o <i>REST</i> denomina que se deve realizar a manipulação destes através da utilização de métodos <i>HTTP</i> <sup>15</sup> .
<i>Self-descriptive messages</i>	A descrição das mensagens através de metadados implica diretamente na facilidade da utilização de recursos, dispensando um contrato preestabelecido.
<i>Hypermedia as the engine of application state</i>	O <i>HATEOAS</i> está vinculado ao conceito de <i>Uniform Interface</i> <sup>16</sup> que é uma das principais características que diferencia o <i>REST</i> de qualquer outra arquitetura baseada em estilos. O <i>HATEOAS</i> deve garantir que aplicações que utilizem <i>REST</i> (também conhecidas como <i>RESTful</i> ) utilizem identificadores padronizados.

**Adaptado:** (FIELDING, 2000)

De acordo com o que foi apresentado, é possível concluir que ambas as tecnologias, *REST* e *SOAP*, são viáveis para a construção de aplicações que necessitam de comunicação entre si. Em vista disso, cabe realizar um estudo para se identificação de tecnologias que se utilizar em uma determinada aplicação, levando em consideração as vantagens e desvantagens de cada uma.

No caso do trabalho em questão, a tecnologia *REST* pode ser melhor aplicada, por trabalhar da forma mais transparente possível e permitir a criação de padrões sem um conjunto de regras bem estabelecidas.

É importante salientar que a tecnologia *SOAP* não deixa de ser uma opção viável, uma vez que é possível a utilização da mesma para a realização dos mesmos procedimentos que serão implementados utilizando a arquitetura *REST*.

Os *web services* podem ser utilizados para diversas finalidades, uma delas é na troca de informações de identidade do usuário entre aplicações distintas. Diante

<sup>14</sup> *URIs* ou *Uniform Resource Identifier* são curtas sequências que identificam recursos na web, tais como: documentos, imagens, arquivos para download, serviços etc. Disponível em: <<https://goo.gl/mMsm9E>>. Acesso em 28 abr. 2016.

<sup>15</sup> Os métodos *HTTP* suportados pelo *REST* são: *GET*, *PUT*, *DELETE* e *POST*. Disponível em: <<http://goo.gl/QY1SWp>>. Acesso em 28 abr. 2016.

<sup>16</sup> A interface uniforme garante a simplicidade da arquitetura global do sistema e uma melhora da visibilidade das interações, de forma que se obtenha operações genéricas e bem definidas.

disso, o próximo tópico visa discutir conceitos acerca de Sistemas de Gerenciamento de Identidade (SGIs).

## 2.6 SISTEMAS DE GERENCIAMENTO DE IDENTIDADE

Esta seção tem como objetivo apresentar opções relevantes de provedores de identidade para um melhor embasamento teórico acerca dos Sistemas de Gerenciamento de Identidade (SGIs).

Para BHARGAV-SPANTZEL, CAMENISCH, *et al.* (2007) um sistema de gerenciamento de identidade consiste em uma série de componentes de *softwares* e protocolos que lidam com a identidade de indivíduos durante todo o ciclo de vida da identidade. Isso significa que sempre que for necessário resgatar informações sobre a identidade de determinado usuário, ou mesmo manipulá-las de alguma forma, o SGI servirá como intermediário nessas transações.

Para uma melhor compreensão acerca de SGIs, o diagrama da Figura 100 demonstra o ciclo de transações desde a solicitação da identidade realizada pela *Relying Party*, por meio de uma requisição do usuário até a resposta do serviço pela mesma.

A Figura 10 retrata um cenário no qual o usuário (1) solicita um serviço (4) disponibilizado pela *Relying Party* (3) que exige a autenticação de (1). Assim que (3) recebe a solicitação, o mesmo verifica (5) com o *Identity Provider* (2) se (1) já está autenticado. Tendo em vista que seja o primeiro acesso de (1), a resposta de (2) para (3) é negativa (6), ou seja, não há uma sessão ativa para (1). Desta maneira (3) manda uma requisição (7) para (1) solicitando que o mesmo se identifique (8) através de (2). Após a identificação de (1) a partir de suas credenciais, (2) envia as informações necessárias para possibilitar a troca de informações (9) para (3). De forma que todo o processo tenha ocorrido com êxito. Assim, (3) recebe as informações (9) referentes a (1), que as verifica e libera o serviço para o usuário (10).



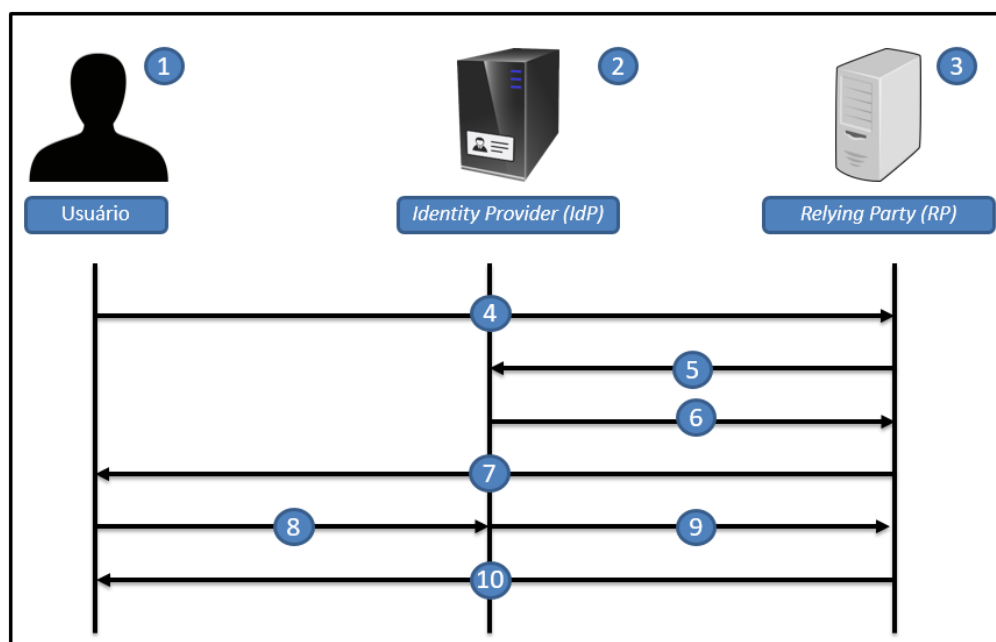


Figura 10 - Ciclo de requisição de identidade em um SGI  
Fonte: do autor, 2016

Conforme pode ser observado, um ambiente que utiliza SGI centralizado vem a ser bem mais complexo que sistemas que mantêm o controle de identidade de seus usuários de forma integrada, conforme o esquema apresentado na Figura 111.

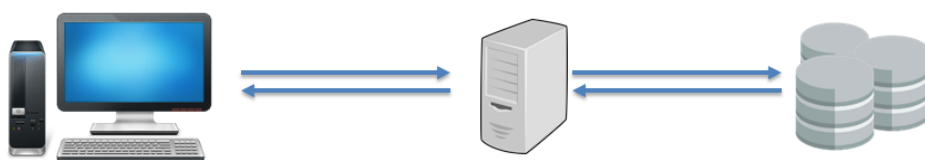


Figura 11 - Esquematização de um sistema que mantém o controle de identidade de usuários integrado  
Fonte: do autor, 2016

Em uma pesquisa realizada pela *Microsoft Research*, foi estimado que um usuário comum possui cerca de 25 senhas na internet (FLORENCIO e HERLEY, 2007). Levando em consideração alguns elementos das boas práticas na elaboração de senhas, segundo a cartilha disponibilizada pelo CERT.BR (2012), não se recomenda a utilização da mesma senha para serviços distintos e sugere alterá-la regularmente, fica claro a dificuldade do usuário seguir todos os princípios impostos.

Ao se tratar de Sistemas de Gerenciamento de Identidade (SGIs), um termo bastante comum é o *Single Sign-On (SSO)*. O método de controle de acesso SSO permite que usuários utilizem suas credenciais apenas uma vez para iniciar sua sessão, a partir daí, é possível ter acesso a múltiplas aplicações sem ter que se autenticar novamente. A arquitetura SSO (apresentada na Figura 122) pode ser dividida em cinco entidades lógicas, são elas (SUN, POSPISIL, *et al.*, 2012):

- *Identity Provider (IdP)*: É a entidade que administra e autentica os usuários.
- *Relying Party (RP)*: A terceira parte é a aplicação que fornece serviços aos utilizadores finais.
- *User*: O usuário é quem assume determinada identidade provida de um *IdP* para acessar os recursos fornecidos pelas *RPs*.
- *User Agent*: O Agente de usuário faz referência a uma aplicação ou software que está em execução no dispositivo de acesso do usuário e que interage com o *IdP* e *RP* em nome do utilizador.
- *Protocol*: O Protocolo é um formato de mensagens e o meio de transporte entre *IdPs*, *RPs* e *User Agents*, que possibilitam a troca de informações.

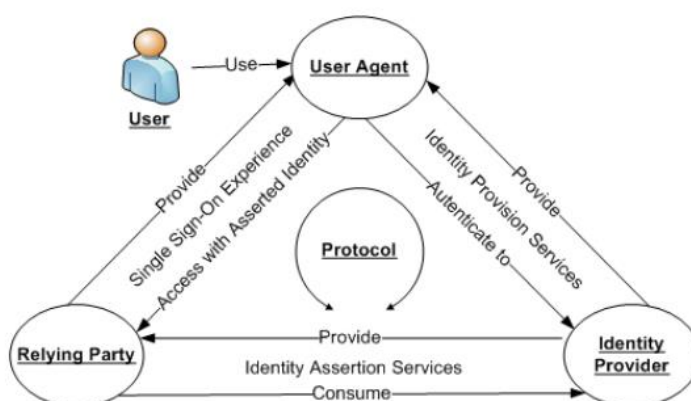


Figura 12 - Modelo triangular do *Single Sign-On*  
 Fonte: (SUN, POSPISIL, *et al.*, 2012)

A Figura 122 representa a arquitetura *Single Sign-On* em seu modelo triangular onde o usuário, a partir do *User Agent*, envia determinada requisição através de um protocolo pré estipulado para a *Relying Party*, que por sua vez realiza

uma verificação com o *Identity Provider* acerca da autenticidade do usuário. Caso o *Identity Provider* não consiga identificar o usuário, a autenticação é solicitada. A partir do momento que o usuário comprova sua autenticidade, o *Identity Provider* libera o acesso para que a *Relying Party* consuma os dados do usuário.

Segundo ARAÚJO (2008), a utilização de SSO traz uma série de benefícios, como o uso de credenciais únicas para a memorização evitando problemas de esquecimento de senhas, dados do usuário e autenticação centralizada em um único servidor além da capacidade de utilização de uma única política de autenticação. Como desvantagem, o autor cita a dificuldade de implementação e possíveis problemas de segurança, quando o sistema não é bem projetado e configurado.

Dessa forma, visando a segurança de aplicações que necessitam de alguma forma identificar o usuário, O'GORMAN (2004) faz uma comparação entre três tipos de autenticação, enumeradas a seguir:

1. *Knowledge-Based*: A autenticação baseada em conhecimento ("O que você sabe"), tem como característica o sigilo. Este tipo confirma a autenticidade do usuário através de uma senha, geralmente memorizada pelo mesmo.
2. *Object-Based*: Autenticação baseada em objetos ("O que você tem"), é fundamentada sob a perspectiva de o usuário possuir um objeto, também chamado de *token*, que permita a sua identificação perante o *IdP*.
3. *ID-Based*: Autenticação baseada em identidade ("Quem você é"), caracteriza-se por identificar determinado usuário através da biometria. Para isso, pode-se utilizar a leitura de impressões digitais, técnicas de reconhecimento vocal, leitura ocular ou até mesmo a assinatura.

Vale ressaltar que os tipos de autenticação propostos anteriormente podem ser agrupados, fazendo com que a segurança das aplicações sejam incrementadas.

### 2.6.1 FERRAMENTAS E TRABALHOS CORRELATOS

Diversas empresas investem em *APIs* que permitem a terceiros utilizarem suas bases de usuários com o objetivo de prover a identidade e autenticidade aos internautas, é o caso do *Facebook*<sup>17</sup>, *Twitter*<sup>18</sup>, *Google*<sup>19</sup>, *Microsoft*<sup>20</sup>, entre outras.

Ao utilizar essas ferramentas, o usuário estará sujeito a ter sua privacidade comprometida, como pode ser observado na Política de Privacidade do Facebook: “Coletamos informações quando você acessa ou usa sites e aplicativos de terceiros que utilizam nossos Serviços” (FACEBOOK, 2015).

Além do Facebook, o *Google* também deixa claro em seus termos de uso a utilização de técnicas para coleta de dados do usuário:

Nós, juntamente com nossos parceiros, usamos várias tecnologias para coletar e armazenar informações quando o usuário visita um serviço da Google. Tais informações podem incluir o uso de cookies ou tecnologias semelhantes para identificação do navegador ou dispositivo do usuário. [...].

Embora existam diversas ferramentas e padrões que possibilitam o compartilhamento de dados de forma segura para promover a autenticação, como por exemplo a *Security Assertion Markup Language (SAML)*, *OpenID* e *API Tokens*, um protocolo que vem se destacando cada vez mais é o *OAuth* (BILBIE, 2013).

O *OAuth* é um protocolo de código aberto amplamente utilizado para promover de forma simples a autenticação de usuários. O protocolo em sua versão 2.0 possui suporte a diversas linguagens de programação, como *Java*, *PHP*, *Python*, *Ruby*, *.NET*, entre outras. Além de fornecer uma infinidade de bibliotecas, é suportado por serviços de grandes sites, tais como: *Google*, *Facebook*, *Dropbox*, *Foursquare*, *Windows Live*, e outros (OAUTH, 2012). Segundo D. HARDT (2012, p. 1 tradução nossa) o framework de autorização OAuth 2.0:

permite que aplicações de terceiros obtenham acesso limitado a um serviço HTTP, seja em nome do proprietário de um recurso ao orquestrar uma aprovação de uma interação entre o proprietário do recurso e o serviço

<sup>17</sup> Login no Facebook. Disponível em: <<https://goo.gl/BzhFCO>>. Acesso em: 19 abr. 2016.

<sup>18</sup> Sign in with Twitter. Disponível em: <<https://goo.gl/Cv1Eto>>. Acesso em: 19 abr. 2016

<sup>19</sup> Google+ Platform for WEB. Disponível em: <<https://goo.gl/q082j7>>. Acesso em: 19 abr. 2016.

<sup>20</sup> Windows Live ID Web Authentication. Disponível em: <<https://goo.gl/jMLYBo>>. Acesso em: 19 abr. 2016.

HTTP, ou permitir que a aplicação de terceiros obtenham acesso em seu próprio nome.

O fluxo da aplicação utilizando o protocolo OAuth 2.0, demonstrado na Figura 133, se dá da seguinte forma:

- (A) O *Client* envia uma requisição contendo seu identificador único (*cliente\_id*) para o *Resource Owner* solicitando autorização.
- (B) O *Resource Owner* verifica se o *Client* possui ou não autorização e o responde.
- (C) Caso o *Client* possua autorização para a utilização de recursos, o mesmo solicita ao *Authorization Server* um *token* de acesso (*access\_token*).
- (D) O *Authorization Server* remete ao cliente o *token*.
- (E) Com o *token* em mãos, o *Client* pode acessar recursos do *Resource Server* utilizando seu identificador único, o *token* de acesso e seu *token* secreto (*secret\_token*).

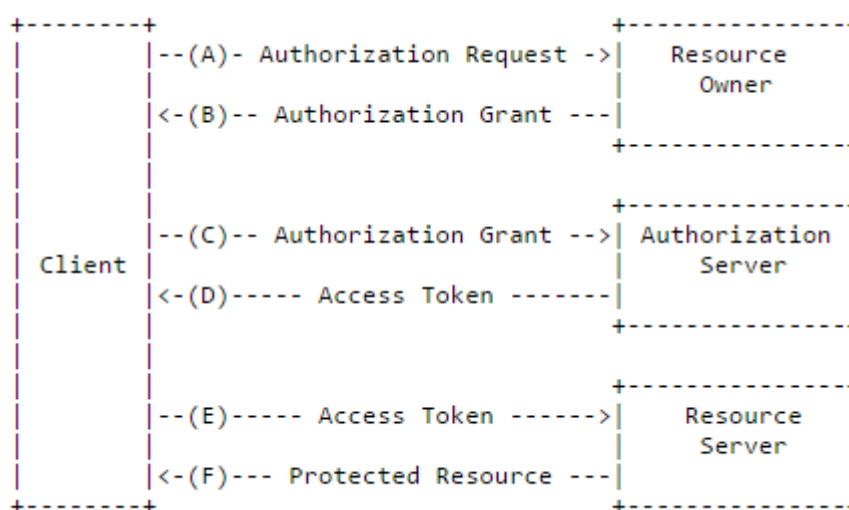


Figura 13 - Fluxo abstrato do protocolo OAuth 2.0  
Fonte (D. HARDT, 2012)

O padrão *SAML* 2.0 é outro detentor de diversas vantagens. Além de possibilitar o modelo de gerenciamento de identidade centralizado, também permite a utilização de forma federada, onde determinada aplicação pode utilizar diversos provedores de identidade para se autenticar. Outro benefício do *SAML* 2.0 é sua diversidade de *frameworks* disponíveis, como o SimpleSAMLphp que permite uma

rápida implementação de soluções de identidade (SOUTO, DOMENECH e WANGHAM, 2014).

Diversas organizações que mantêm o foco em segurança, como é o caso dos bancos, estão cada vez mais investindo em soluções para evitar fraudes e facilitar a vida dos usuários. Este é o caso do Banco do Brasil que permite que seus clientes acessem suas contas através de seus celulares utilizando a impressão digital ou a leitura de *QR Code* (abordada no capítulo 2.4.1 *QR CODE*) para confirmar transações (PICCOLO, 2016).

Cabe citar que diferentes empresas vêm buscando cada vez mais a abolição das senhas baseadas em conhecimento. Um dos fatores que sustentam esta afirmação é um aplicativo lançado pelo *Facebook* chamado *Account Kit*, que permite que desenvolvedores disponibilizem uma solução para seus usuários, de forma que eles possam se autenticar utilizando seu número de telefone ou endereço de e-mail (KUZNETSOVA, 2016).

Nesta seção foram apresentadas diversas soluções e conceitos que podem ser incorporados na criação de sistemas viabilizando a identificação do usuário. O próximo capítulo abordará a metodologia que será utilizada para a realização deste trabalho.

### 3 METODOLOGIA

Esta seção apresentará a metodologia que foi utilizada no decorrer do desenvolvimento deste trabalho de conclusão de curso, considerando tecnologias, ferramentas e sua relação.

#### 3.1 DESENHO DE ESTUDO

O presente trabalho caracteriza-se como uma pesquisa **bibliográfica e aplicada**. Bibliográfica, pois sua aplicação é essencial para a realização de estudos em que se busca domínio sobre determinado tema a partir de uma fundamentação teórica (CERVO, BERVIAN e SILVA, 2006).

Para GIL (2002) a pesquisa bibliográfica:

[...] é desenvolvida com base em material já elaborado, constituído principalmente de livros e artigos científicos. Embora em quase todos os estudos seja exigido algum tipo de trabalho dessa natureza, há pesquisas desenvolvidas exclusivamente a partir de fontes bibliográficas. Boa parte dos estudos exploratórios pode ser definida como pesquisas bibliográficas.

A pesquisa em questão qualifica-se, também, no critério de aplicada, uma vez que há a necessidade de produção de conhecimento tendo como objetivo a contribuição para fins práticos, ou seja, buscar soluções para problemas concretos (BARROS e LEHFELD, 2000).

Para demonstrar metodologicamente a maneira que o projeto foi realizado, o tópico a seguir tem como objetivo descrever cada uma das fases de desenvolvimento deste trabalho.

#### 3.2 DESENHO METODOLÓGICO

Este tópico visa tratar do conjunto de procedimentos que foram adotados para a realização deste trabalho.

Para representar de maneira mais clara e objetiva a metodologia utilizada, o diagrama presente na Figura 14 - Desenho metodológico da pesquisa<sup>4</sup> demonstra todos os passos, desde a coleta bibliográfica até as considerações finais.

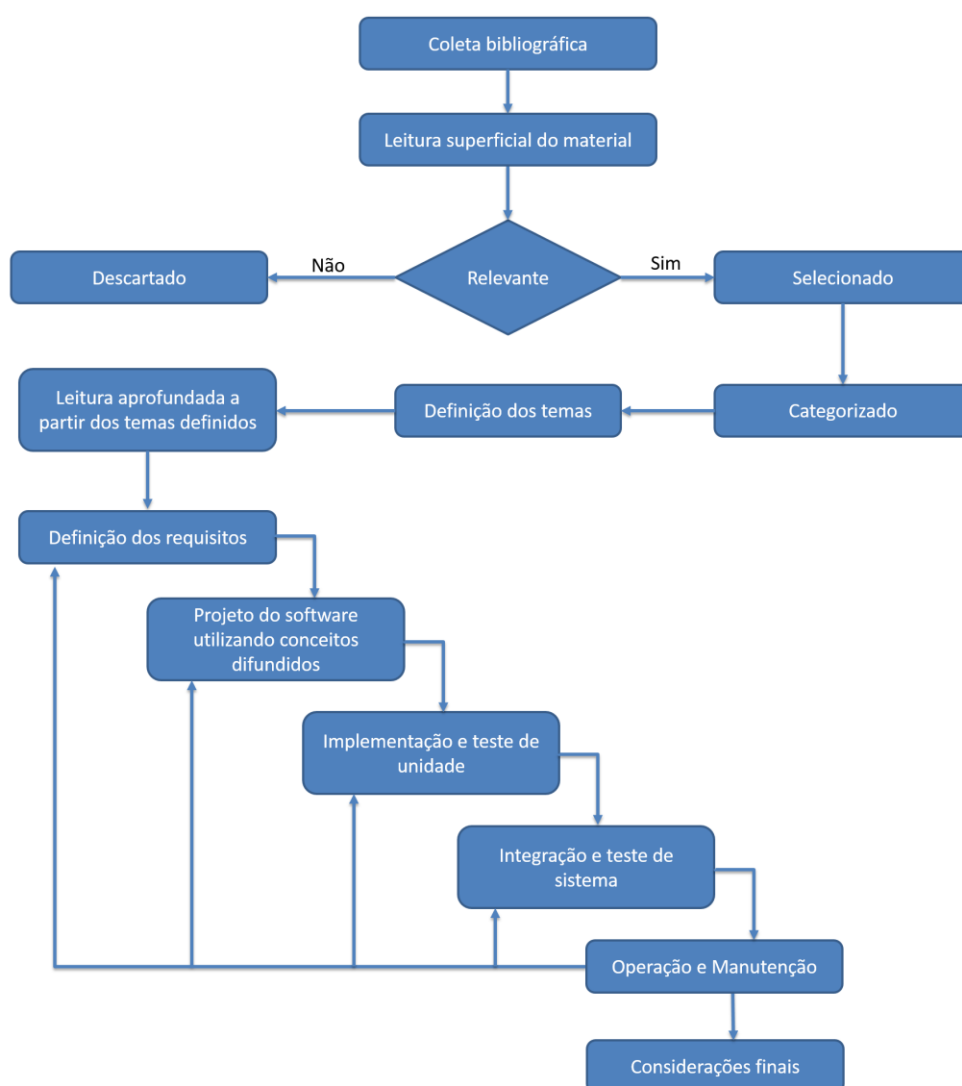


Figura 14 - Desenho metodológico da pesquisa  
Fonte: do autor, 2016

Como pode ser observado no desenho metodológico exposto na Figura 14, na Coleta bibliográfica, na condição de primeira fase deste trabalho, foi realizada a busca de todos os materiais que serviriam de base teórica. Em seguida foi efetivada uma leitura superficial em todos os livros, revistas, links, artigos e dissertações. Os materiais que trouxeram consigo um conteúdo irrelevante quanto ao tema apresentado foram imediatamente descartados, enquanto os outros foram selecionados e categorizados.

O próximo passo caracterizou-se pela definição dos temas abordados no capítulo 2, o do REFERENCIAL TEÓRICO, tomando como base nas categorias



definidas na fase anterior. Posteriormente foi realizada uma leitura aprofundada dos temas definidos.

As cinco próximas etapas: definição dos requisitos, projeto de *software*, implementação de testes de unidade, integração e teste de sistema e Operação e Manutenção referem-se aos cinco estágios do desenvolvimento de *software*. A seguir, faz-se uma breve descrição sobre cada um deles (SOMMERVILLE, 2007).

- Análise e Definição de Requisitos: Fase em que todos os serviços, restrições e objetivos são definidos e servem como uma especificação do sistema.
- Projeto de sistema e *software*: Envolve toda a identificação e descrição das abstrações do *software*.
- Implementação e teste de unidade: Envolve o desenvolvimento de todas as unidades do sistema e a realização dos testes de cada uma das unidades separadamente.
- Integração e teste de sistema: Todas as unidades individuais são integradas em seguida testadas no sistema como um todo.
- Operação e Manutenção: Geralmente fase mais longa do ciclo de vida, onde o sistema é colocado em operação. A manutenção envolve a correção de erros não detectados anteriormente, aprimoramentos na aplicação e implementação de novas unidades a medida que se identifica novos requisitos.

Como pode ser observado, é imprescindível a utilização de um modelo para a construção de uma aplicação. A escolha do modelo cascata foi realizada devido aos seus estágios serem bem definidos e a partir da fase de manutenção, ser possível retornar para cada um dos estágios anteriores.

### 3.3 MATERIAIS E MÉTODOS

Com a finalidade de solucionar a questão da utilização de senhas complexas e de seus possíveis esquecimentos por parte do usuário, este trabalho visa utilizar uma alternativa já citada na seção **Erro! Fonte de referência não encontrada.**, a autenticação *Object-Based*.

Para a obtenção de uma melhor compreensão acerca do objetivo deste trabalho, a Figura 155 demonstra de forma simplificada os principais dispositivos envolvidos e suas relações entre si.

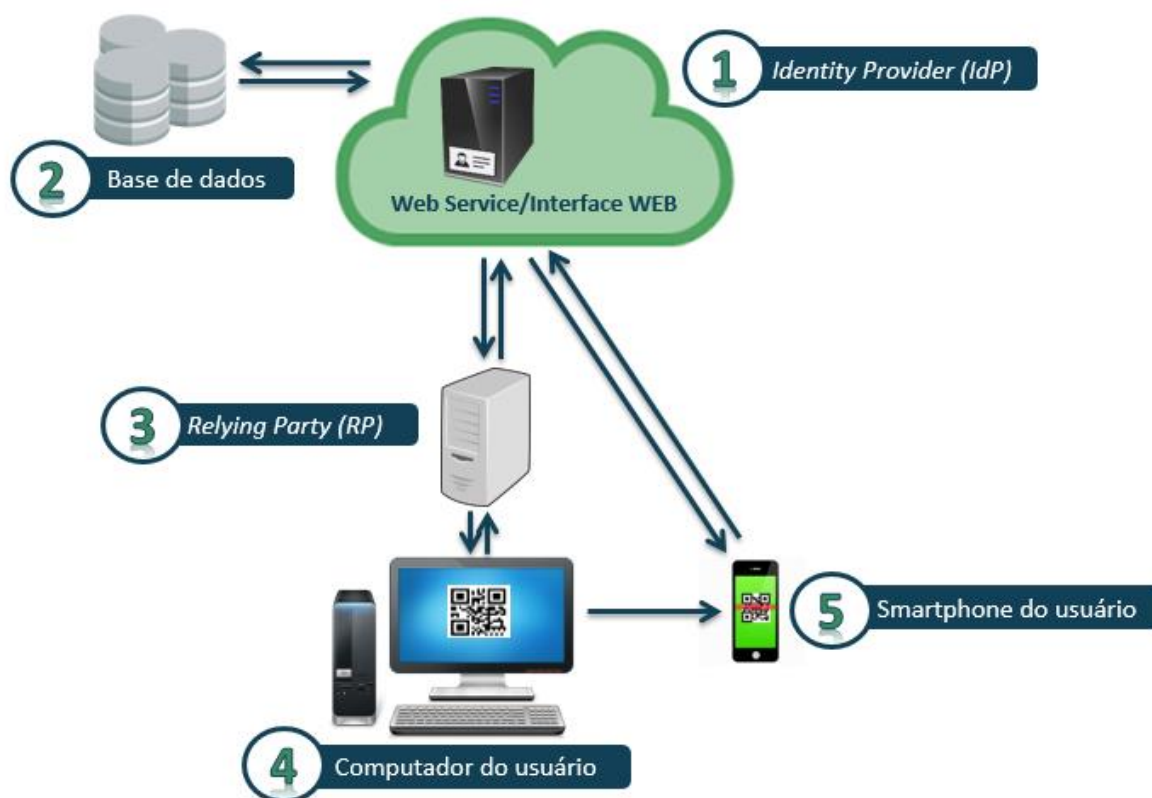


Figura 15 - Solução proposta para o trabalho  
Fonte: do autor, 2016

O objeto utilizado como detentor da capacidade de autenticação dos usuários foram seus próprios smartphones, os quais deverão possuir um aplicativo instalado que realizará a leitura de um *QR Code* fornecendo de forma imediata os dados de autenticação do usuário para a aplicação requisitante.

A seguir uma descrição de cada uma das partes envolvidas.

### 3.3.1 BASE DE DADOS

A base de dados possui papel fundamental para prover a persistência das informações e integridade dos dados. Para a realização deste trabalho foi utilizado o

sistema de gerenciamento de banco de dados objeto-relacional *PostgreSQL* na versão 9.5.

Todas as informações armazenadas por parte do usuário e aplicações envolvidas serão armazenadas na base de dados. Além disso, o acesso direto dessas informações só serão possíveis através do *Identity Provider*, que será apresentado a seguir.

### 3.3.2 IDENTITY PROVIDER (IdP)

Denomina-se *Identity Provider* a entidade responsável por fornecer a identidade dos usuários para as aplicações autorizadas. Para a realização deste trabalho o *IdP* foi segmentado em duas camadas, sendo uma delas de serviços e outra de gerenciamento, dispostas a seguir:

A **Camada de Serviços** é responsável por possibilitar o acesso a informações do *IdP* através de um *Web Service* utilizando o protocolo OAuth na versão 2.0. Este acesso é utilizado tanto para as *Relying Party*, que serão definidas no próximo tópico, quanto para a aplicação *mobile*.

A **Camada de Gerenciamento** é a interface web que tem como papel permitir que o usuário possa gerenciar todas as informações de sua conta. Além disso, o cadastro dos dispositivos (*smartphones*), autorização e revogação de aplicações serão realizados nesta camada.

A utilização da senha propriamente dita ainda será necessária para que o usuário possa se autenticar na camada de gerenciamento, caso o mesmo não esteja com seu smartphone disponível no momento da autenticação.

A linguagem de programação definida para o desenvolvimento do *IdP* foi *Java*. Será utilizado o conjunto de especificações *Java Enterprise Edition 8* em conjunto com o *Apache Tomcat 8* como servidor de aplicação. Como ambiente de desenvolvimento foi utilizado o software *IntelliJ IDEA 15.0 Student*.

### 3.3.3 RELYING PARTY (RP)

Toda e qualquer aplicação que utilizará dos serviços disponibilizados pelo *IdP* será denominada *Relying Party (RP)*. As *RPs* deverão utilizar o protocolo OAuth para se comunicar com *Identity Provider* e obter acesso às credenciais dos usuários.

Por ser designado para trabalhar com requisições sob o protocolo *HTTP* é possível utilizar o OAuth em qualquer aplicação que tenha suporte para o tal, independente de implementação ou da linguagem.

### 3.3.4 COMPUTADOR / SMARTPHONE DO USUÁRIO

Sempre que o usuário necessitar se identificar perante alguma *RP*, o mesmo será redirecionado para o *IdP* que verificará se há alguma sessão ativa. Caso o usuário não tenha iniciado uma sessão, ele aponta seu *smartphone* com a aplicação QRAuth instalada e realiza a leitura de um *QR Code* na página de login do *IdP*. Assim que se verifica a autenticidade do usuário e se constata que a aplicação requisitante (*RP*) possui permissão para acessar seus dados, o fluxo continua enviando um token de acesso para a *RP*.

A aplicação *mobile* será desenvolvida utilizando Java, já citado anteriormente, em conjunto com o *Android SDK*.

A próxima seção abordará os resultados obtidos no desenvolvimento deste trabalho.

## 4 RESULTADOS

Os resultados adquiridos no decorrer deste trabalho, os processos de modelagem do *software*, as telas desenvolvidas, entre outros artefatos estarão dispostos nas próximas seções.

### 4.1 MODELAGEM

Visando obter uma melhor compreensão acerca da estrutura do *software* desenvolvido, o diagrama de classes proposto na Figura 16 - Diagrama de classes traz consigo uma visão geral das entidades envolvidas e seus respectivos atributos.

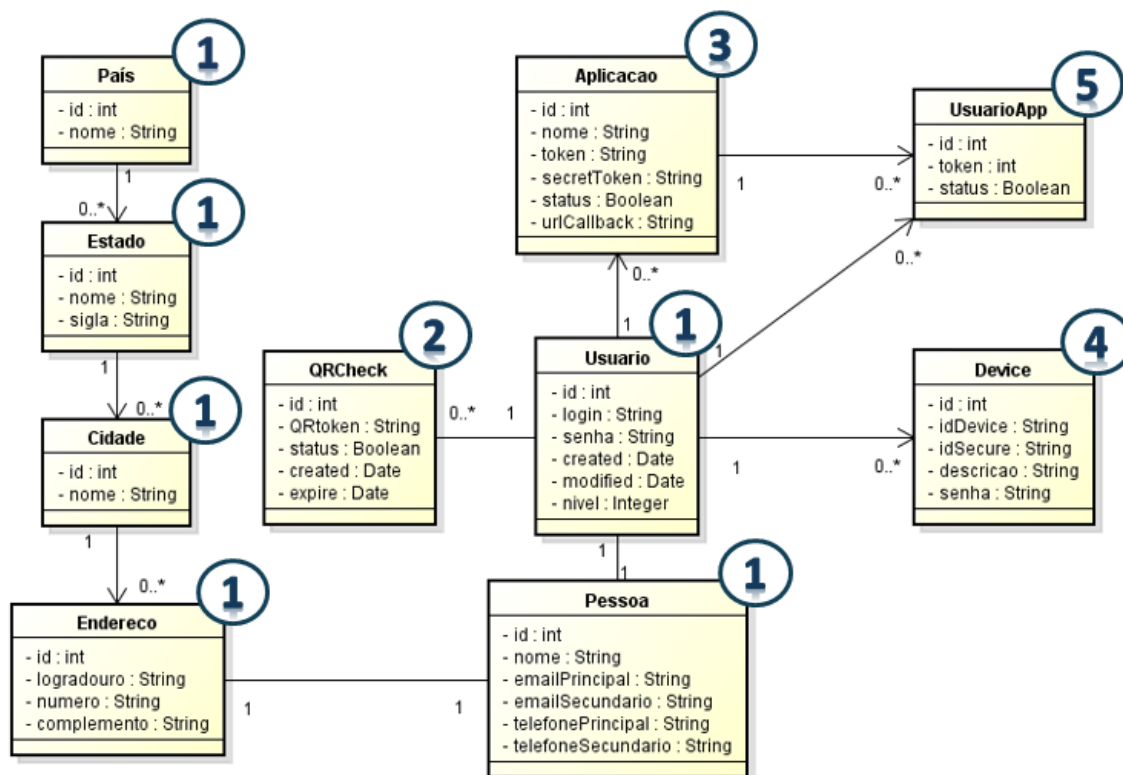


Figura 16 - Diagrama de classes  
Fonte: Do autor, 2016

Na Figura 16 - Diagrama de classes, as classes demarcadas com 1 representam as informações individuais de cada usuário cadastrado no sistema. A classe QRCheck (2) é responsável por iniciar a sessão do usuário guardando o *token* da sessão (QRToken) e a data que o mesmo irá expirar. A finalidade da classe

Aplicação (3) é armazenar os dados das *Relying Party*, os quais são: o nome da aplicação, o *token*, o *token* secreto e a *url* de redirecionamento. Os dispositivos móveis dos usuários são representados na entidade *Device* (4). Cada um deles trará um identificador único (*idDevice*) e um identificador de segurança (*idSecure*), além de uma descrição e uma senha para acesso. A última entidade, *usuárioApp*, terá como papel estabelecer o vínculo das *Relying Party* e dos Usuários, fornecendo um *token* que permitirá as aplicações terem acesso aos dados dos usuários vinculados.

Na próxima seção serão discutidos e apresentados os resultados obtidos a partir do *software* desenvolvido para este trabalho.

## 4.2 APRESENTAÇÃO DA APLICAÇÃO

Conforme apresentado na metodologia, diversos conceitos foram utilizados para a construção da aplicação proposta. As próximas seções demonstrarão onde e como tais conceitos foram empregados, iniciando pelo elemento *QR Code*.

### 4.2.1 QR CODE

Como a biblioteca de componentes visuais *Primefaces* permite a concepção de diversos tipos de código de barras, inclusive do tipo *Quick Response Code*, optou-se por utilizá-la em todas as situações em que houvesse a necessidade deste componente na interface *web*, conforme pode ser conferido na tela de *login* da aplicação, representada na Figura 17 - Tela de login.



Figura 17 - Tela de login

A página de *login* (disposta na Figura 17 - Tela de login) realiza de tempos em tempos requisições para o servidor verificando se o usuário conectou-se através da leitura do QR Code na aplicação *mobile*. Sempre que a autenticação é realizada desta forma, um registro da entidade **QRCheck** (que pode ser conferida no diagrama de classes na Figura 16 - Diagrama de classes) é criado, relacionando o código com o usuário. O método responsável por verificar se o usuário realizou a leitura do QR Code está representado na

Figura 18).

```

1 public void isLoggedInByQrCode() {
2     QRCheckRepository qrCheckRepository = new QRCheckRepository(
3         getEm());
4     QRCheck qrCheck = qrCheckRepository.buscarQrToken(
5         SessionAdministrativo.getAuthenticationToken());
6
7     if (qrCheck != null &&
8         qrCheck.getUsuario() != null &&
9         qrCheck.getUsuario().getId() != null
10    ){
11         SessionAdministrativo.setUsuarioLogado(qrCheck.getUsuario());
12         SessionAdministrativo.updateUsuario();
13         Util.redirect(getSuccessUrl());
14     }
15 }

```

Figura 18 - Método para verificar se o usuário realizou a leitura do QR Code na tela de login

O método disposto na

Figura 18 realiza uma busca na base de dados de usuários autenticados (**QRCheck**) passando como parâmetro o *digest*, representado no QR Code da página de *Login*. Assim que um registro é encontrado, constata-se a existência de um usuário vinculado, armazena-o na sessão e em seguida o usuário é redirecionado para uma das seguintes opções:

- Para a página inicial do sistema: Caso o usuário esteja apenas tentando se autenticar na interface web do QRAuth.
- Para a página de confirmação de vínculo com a *relying party*: Quando determinada aplicação de terceiro ainda não tenha a devida autorização do usuário para obter seus dados.

- Para o endereço designado pelo administrador da *Relying Party*: Na situação em que uma *relying party*, já autorizada pelo usuário, necessita autenticá-lo.

Outro ponto crucial para o desenvolvimento deste trabalho foram os *Web Services*, que serão apresentados na próxima sessão.

#### 4.2.1 WEB SERVICES

Com a utilização do protocolo OAuth 2.0 houve a necessidade da implementação de um *web service* para possibilitar a comunicação entre as diferentes plataformas, permitindo assim a interoperabilidade do sistema.

Desse modo, tamanha a importância da utilização de um *framework* para gerir os serviços web, para este trabalho, utilizou-se da biblioteca de código aberto *RESTful Jersey*, que fornece uma série de métodos que auxiliam na manipulação de dados aplicando o padrão *REST* (retratados na

Figura 19 - Métodos de acesso do Web Service).

1	@POST
2	@Path("/authDevice/")
3	@Produces(MediaType.APPLICATION_JSON)
4	public String authDevice(@FormParam("secureId") String secureId,
5	@FormParam("deviceId") String deviceId,
6	@FormParam("devicePassword") String devicePassword
7	){...}
8	
9	
10	@GET
11	@Path("/user/{appToken}/{secretToken}/{userAppToken}")
12	@Produces(MediaType.APPLICATION_JSON)
13	public String qrCheck(@PathParam("appToken") String appToken,
14	@PathParam("secretToken") String secretToken,
15	@PathParam("userAppToken") String userAppToken
16	){...}
17	
18	



```

19  @GET
20  @Path("/token/{appToken}/{secretToken}/{qrToken}")
21  @Produces(MediaType.APPLICATION_JSON)
22  public String getToken(@PathParam("appToken") String appToken,
23                        @PathParam("secretToken") String secretToken,
24                        @PathParam("qrToken") String qrToken
25                        ) {...}
26
27
28  @GET
29  @Path("/login/{qrToken}/{idDevice}")
30  @Produces(MediaType.APPLICATION_JSON)
31  public String addQRToken(@PathParam("qrToken") String qrToken,
32                          @PathParam("idDevice") String idDevice
33                          ) {...}

```

Figura 19 - Métodos de acesso do Web Service

Conforme apresentado na

Figura 19 - Métodos de acesso do Web Service, foram utilizados quatro métodos de acesso ao *Web Service*. O primeiro deles **authDevice** (linha 4) é responsável por autenticar o *smartphone* do usuário, com a passagem de três parâmetros. São eles:

- *secureId*: Verificando-se da necessidade de aumentar a segurança na autenticação, o parâmetro **secureId** é uma identificação única de cada *smartphone* denominada *International Mobile Equipment Identity (IMEI)*. O armazenamento do *IMEI* possibilita verificar a autenticidade do dispositivo no momento que o mesmo tenta realizar o login na aplicação.
- *deviceId*: Durante o cadastro do dispositivo, uma identificação única no padrão: **deviceId:<Digest aleatório>** é gerada para que seja estabelecido um vínculo do *smartphone* com o usuário.
- *devicePassword*: Cada dispositivo cadastrado possui uma senha numérica de 6 dígitos, através desta é possível atestar a autenticidade do acesso do usuário através de seu *smartphone*.

Os métodos **qrCheck** e **getToken**, descritos nas linhas 13 e 22, respectivamente na Figura 17 - Tela de login, efetuam papéis similares. O primeiro deles pode ser utilizado pelas *Relying Parties* a qualquer momento, desde que possuam o *token* do usuário para a aplicação e consequentemente autorização para obter suas informações. O segundo método, utilizado para o mesmo propósito, é empregado no momento em que o usuário possui uma sessão estabelecida e a *Relying Party* necessita buscar seus dados. A seguir a descrição de cada um dos parâmetros destes métodos.

- *appToken*: Cada *Relying Party* possui uma identificação única denominada *appToken*. Este é um dado público e é repassado ao QRAuth via *URL* no momento em que o usuário necessita se autenticar em determinada aplicação de terceiro.
- *secretToken*: Para que se possa verificar a autenticidade determinada *Relying Party* em suas requisições, cada uma possui um senha de 32 caracteres denominada *secretToken*. Esta não pode ser alterada, todavia, pode ser gerada novamente na página de Alteração da Aplicação.
- *userAppToken*: Para cada aplicação que o usuário é vinculado é gerado um token de identificação exclusivo designado *userAppToken*.
- *qrToken*: Representa o *digest* aleatório que está contido no *QR Code* da página de *login* (como verifica-se na Figura 17 - Tela de login). Assim que o usuário é autenticado, o *qrToken* passa a ser o identificador da sessão, permitindo que outras aplicações, a partir dele, consumam seus dados através do *web service*.

O último método, representado na linha 31 da

Figura 19 - Métodos de acesso do Web Service, designado **addQRToken** é utilizado pelo *smartphone* para armazenar um registro na tabela QRCheck na base de dados, permitindo que se identifique a autenticação do usuário via dispositivo na interface *web* do sistema. Dois parâmetros são enviados para o servidor, o *qrToken*

que já foi conceituado anteriormente e o *idDevice* que representa a identificação única do dispositivo.

Como pôde ser observado, o *Web Service* criado teve um papel fundamental para a realização deste trabalho. Na próxima seção será apresentada a aplicação *mobile* desenvolvida.

#### **4.2.1 SOFTWARE WEB E MOBILE**

As aplicações *Web* e *Mobile*, presentes neste trabalho, podem ser consideradas interfaces através das quais o usuário final tem uma interação direta. Além disso, é através destas que serão controladas as ações do sistema.

A tela de *login* do sistema, já apresentada na Figura 17 - Tela de login, possibilita o usuário se autenticar informando seu nome de usuário ou *e-mail* e sua senha de acesso. É possível também se identificar utilizando o módulo *mobile* do *QRAuth* instalado em um *smartphone* e apontando-o para o *QR Code*, como será visto posteriormente.

A página inicial do sistema, apresentada na Figura 20, fornece uma série de atalhos (3) para facilitar a interação do usuário. Na barra lateral está disposto o menu de navegação (1) e de desenvolvedores (2). Além disso, o *software* traz consigo o recurso de responsividade, permitindo adaptar-se em diferentes tamanhos de telas e dispositivos.

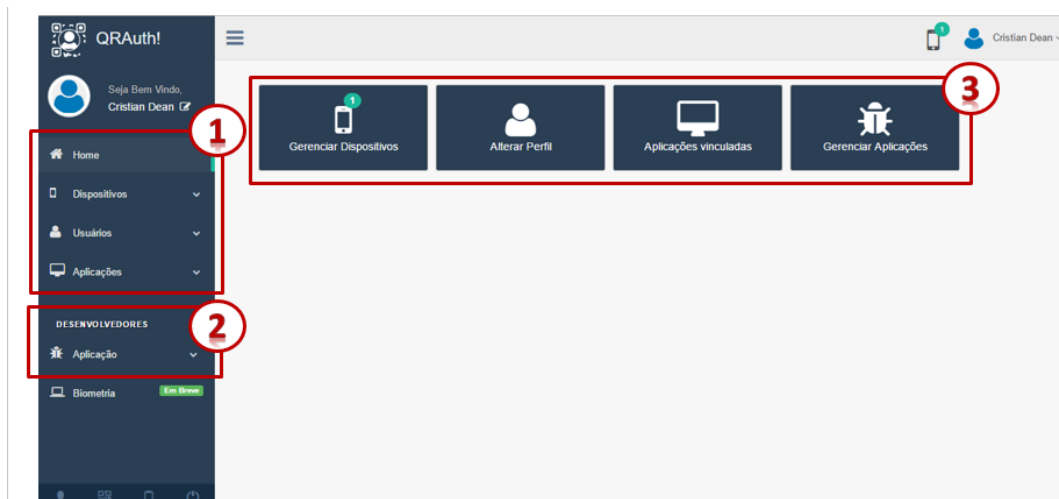


Figura 20 - Página inicial da aplicação QRAuth

O primeiro passo para a utilização do sistema é a criação de uma *Relying Party*, que neste trabalho será tratada pelo termo **aplicação**. Para que determinado usuário seja capaz de criar uma aplicação, é necessário que o mesmo esteja autenticado com o perfil de Desenvolvedor.

O cadastro de uma aplicação exige apenas um nome, para identificá-la, e uma *Url de Callback* para redirecionar o usuário depois de autenticado. Uma vez cadastrada, a aplicação fornece dois dados cruciais para o desenvolvedor, o *Token* (também chamado de *appToken*) e o *Secret Token*, já conceituados no capítulo anterior, como pode ser visto na Figura 21.

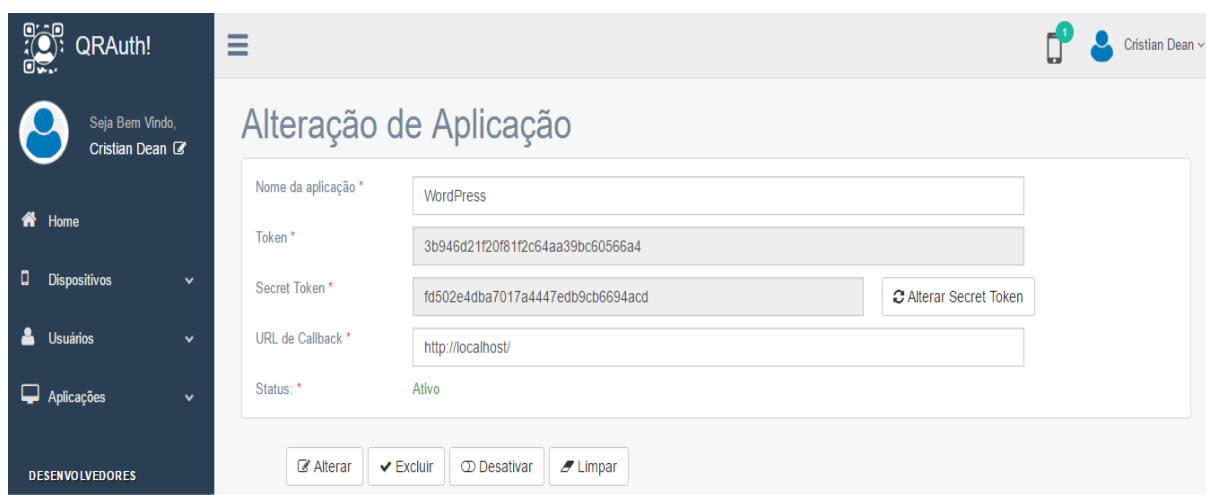


Figura 21 - Página de alteração de aplicações

Caso o desenvolvedor, por algum motivo necessite gerar um novo *Secret Token*, esta opção está disponível na página de alteração da aplicação (Figura 21).

O cadastro dos dispositivos (*smartphones* e *tablets*) pode ser efetuado através do menu Dispositivos no item Cadastrar Dispositivo (1), como pode ser observado na Figura 21. A descrição (2) e uma senha de 6 dígitos numéricos (3) são obrigatórias.



Figura 22 - Página de alteração de dispositivos

Assim que realizado o cadastro do dispositivo móvel, o próximo passo é vinculá-lo ao aparelho. Para realizar esta operação é necessário efetuar a instalação do módulo *mobile* no *smartphone/tablet* e através da interface de *login* (apresentada na Figura 23) pressionar o botão indicado com o número 1 para realizar a leitura do QR Code na página de alteração de dispositivos (apontado como 4 na Figura 22).

Uma vez detectado o código do dispositivo móvel, deve-se informar no teclado aleatório do aparelho a senha numérica cadastrada, e pressionar o botão Entrar (indicados como número 2 e 3 respectivamente na Figura 23). A utilização de um teclado numérico randômico, como utilizando em terminais de autoatendimento bancário, fornece uma segurança extra para o usuário ao inserir sua senha em locais públicos.



Figura 23 - Interface de login da aplicação *mobile*

Uma vez realizada a autenticação na aplicação *mobile*, o usuário é direcionado para a tela inicial do aplicativo, onde há a descrição do dispositivo e dois botões, 'Logar na aplicação' e 'Sair' respectivamente (como pode ser observado na Figura 24).



Figura 24 - Interface inicial da aplicação *mobile*

Ao tocar no botão 'Logar na aplicação' (Figura 22), a câmera do dispositivo será acionada para possibilitar a leitura do QR Code na página de login da interface *web* (ação demonstrada na Figura 23).

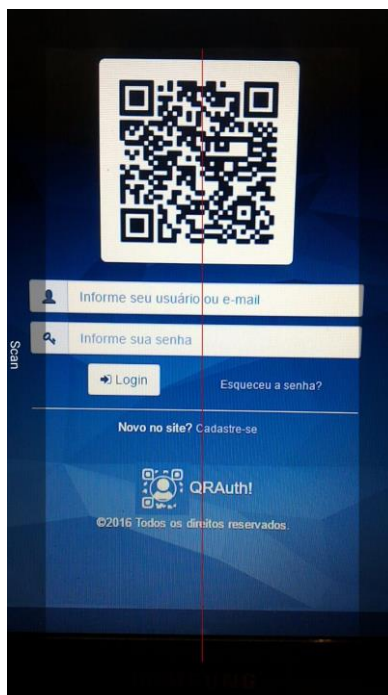


Figura 25 - Smartphone realizando a leitura do QR Code na interface de login

Assim que a leitura do *QR Code* é efetivada, o usuário automaticamente é redirecionado para uma das seguintes opções:

- Para a página inicial do sistema: Caso o usuário esteja apenas tentando se autenticar na interface web do QRAuth.
- Para a página de confirmação de vínculo com a *relying party*: Quando determinada aplicação de terceiro ainda não tenha a devida autorização do usuário para obter seus dados.
- Para o endereço designado na Url de *Callback* (demonstrado na Figura 21): Na situação em que uma *relying party*, já autorizada pelo usuário, necessita autenticá-lo.

Por questões de privacidade do usuário, antes de qualquer *relying party* ter acesso aos seus dados, é necessário conceder uma autorização (conforme pode ser visto na Figura 26)

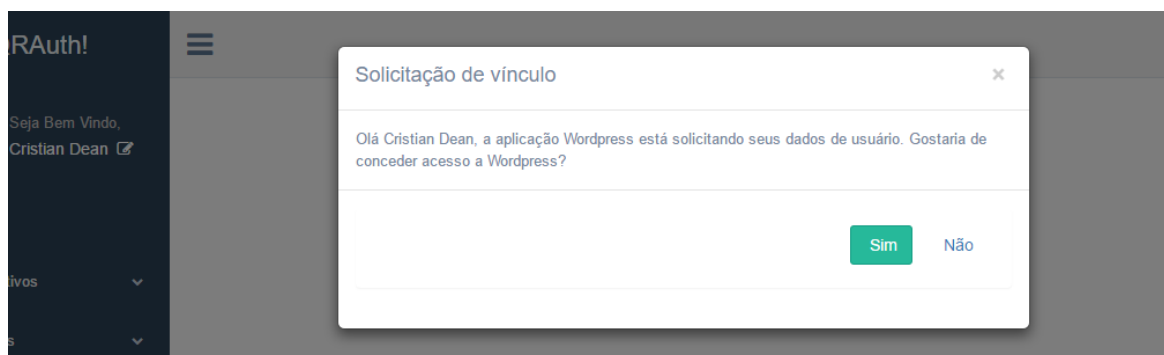


Figura 26 - Página de solicitação de vínculo do usuário

Na próxima seção serão apresentados testes que foram realizados na aplicação, assim como seu funcionamento em diversas situações.

### 4.3 TESTES

Com o intuito de demonstrar o funcionamento do sistema desenvolvido e seguir a metodologia proposta na Figura 14, esta seção apresentará os testes informais efetivados.

Para comprovar a viabilidade da solução proposta neste trabalho, apontando tanto para termos funcionais quanto para questões de integração com aplicações de terceiros, os testes foram realizados com o objetivo de simular uma situação real.

De acordo com o site W<sup>3</sup>TECHS (2016), mais de 25% de todos os *websites* disponíveis na internet utilizam como plataforma o sistema de gerenciamento de conteúdo WordPress. Além de sua popularidade a grande quantidade de plugins disponíveis no repositório oficial<sup>21</sup> da plataforma tornou o WordPress, na versão 4.6.1, uma cobaia ideal para servir de *Relying Party* nos testes realizados.

Tendo em vista que a aplicação desenvolvida utiliza-se do protocolo *OAuth* 2.0 para realizar a comunicação entre as partes envolvidas, foi realizada uma busca pelo termo “*OAuth*” no diretório de *plugins* do *WordPress* e centenas de resultados foram encontrados. Por possuir mais de 1.000 instalações ativas e ser compatível com 9 plataformas diferentes, entre elas: *Google*, *Facebook* e *LinkedIn*, o plugin WP-

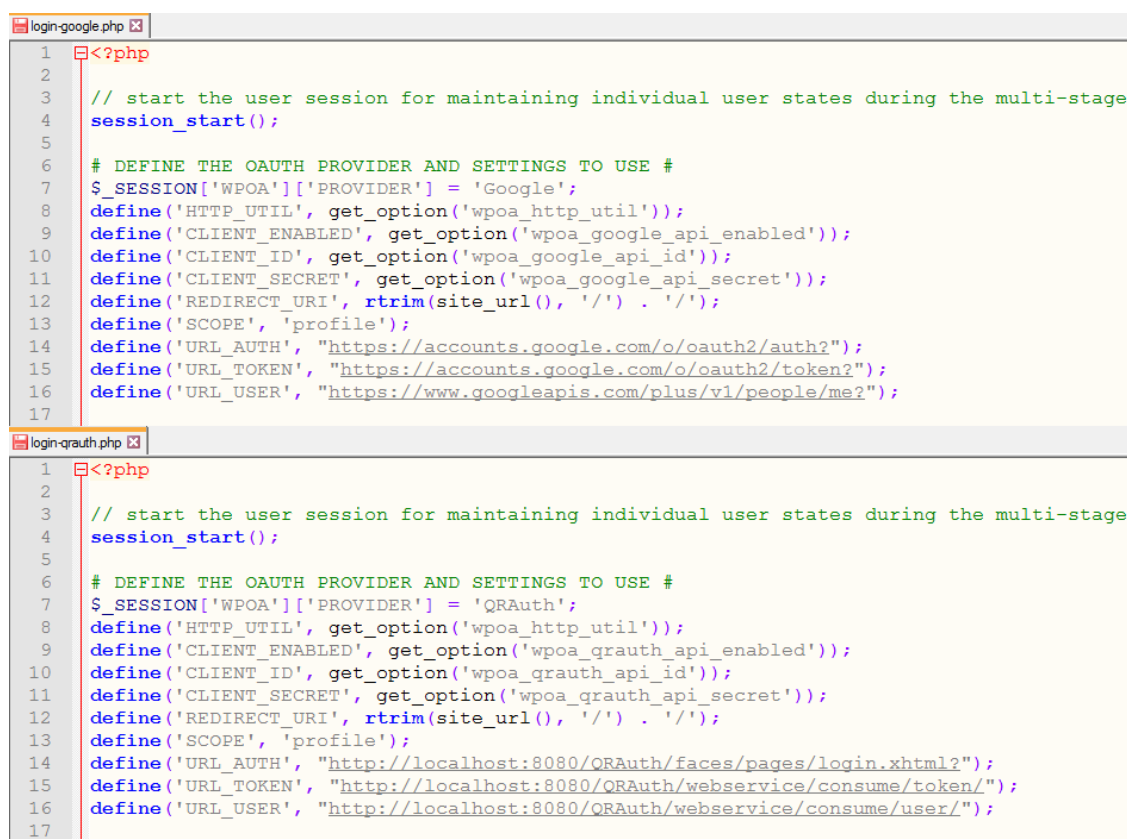
<sup>21</sup> Diretório de Plugins Wordpress. Disponível em: <<https://goo.gl/4mUIgY>>. Acesso em: 15 de Out. de 2016



OAuth<sup>22</sup> foi selecionado para fornecer a integração nos testes da aplicação desenvolvida neste trabalho.

Após uma análise nos códigos do *plugin* WP-Oauth, notou-se, primeiramente, o seguinte padrão no nome dos arquivos de configuração: **login-nomeDoldP.php**, onde **nomeDoldP** refere-se a uma identificação para o *Identity Provider*. Também percebeu-se que com poucas modificações (disponíveis na Figura 27) seria possível adaptá-lo para a realização dos testes.

Conforme pode ser conferido na Figura 27, os valores definidos para as constantes URL\_AUTH, URL\_TOKEN e URL\_USER foram devidamente alterados para as URLs responsáveis por estes serviços, disponíveis no *Web Service* criado. As outras constantes definidas ainda no escopo do código disponível na Figura 27 representam os campos do formulário apresentado na Figura 28.



```

login-google.php
1 <?php
2
3 // start the user session for maintaining individual user states during the multi-stage
4 session_start();
5
6 # DEFINE THE OAUTH PROVIDER AND SETTINGS TO USE #
7 $_SESSION['WPOA']['PROVIDER'] = 'Google';
8 define('HTTP_UTIL', get_option('wpoa_http_util'));
9 define('CLIENT_ENABLED', get_option('wpoa_google_api_enabled'));
10 define('CLIENT_ID', get_option('wpoa_google_api_id'));
11 define('CLIENT_SECRET', get_option('wpoa_google_api_secret'));
12 define('REDIRECT_URI', rtrim(site_url(), '/') . '/');
13 define('SCOPE', 'profile');
14 define('URL_AUTH', "https://accounts.google.com/o/oauth2/auth?");
15 define('URL_TOKEN', "https://accounts.google.com/o/oauth2/token?");
16 define('URL_USER', "https://www.googleapis.com/plus/v1/people/me?");
17

login-grauth.php
1 <?php
2
3 // start the user session for maintaining individual user states during the multi-stage
4 session_start();
5
6 # DEFINE THE OAUTH PROVIDER AND SETTINGS TO USE #
7 $_SESSION['WPOA']['PROVIDER'] = 'QRAuth';
8 define('HTTP_UTIL', get_option('wpoa_http_util'));
9 define('CLIENT_ENABLED', get_option('wpoa_grauth_api_enabled'));
10 define('CLIENT_ID', get_option('wpoa_grauth_api_id'));
11 define('CLIENT_SECRET', get_option('wpoa_grauth_api_secret'));
12 define('REDIRECT_URI', rtrim(site_url(), '/') . '/');
13 define('SCOPE', 'profile');
14 define('URL_AUTH', "http://localhost:8080/QRAuth/faces/pages/login.xhtml?");
15 define('URL_TOKEN', "http://localhost:8080/QRAuth/webservice/consume/token/");
16 define('URL_USER', "http://localhost:8080/QRAuth/webservice/consume/user/");
17
  
```

Figura 27 - Comparação entre arquivos de configuração do plugin WP-OAuth

<sup>22</sup> WP-Oauth - *WordPress plugins* – Disponível em: <<https://goo.gl/yQ5RYV>>. Acesso em: 15 de Out. de 2016

Assim que realizada a instalação do *plugin* através do gerenciador disponível no *WordPress*, verificou-se que o arquivo de configuração foi interpretado com sucesso até o momento, pois um bloco para a inserção das informações já estava disponível na página de configuração do *plugin*, como pode ser visto na Figura 28. Os *Token* da Aplicação e o *Secret Token* foram devidamente informados utilizando as informações da *Relying Party*, já demonstrada na Figura 21.

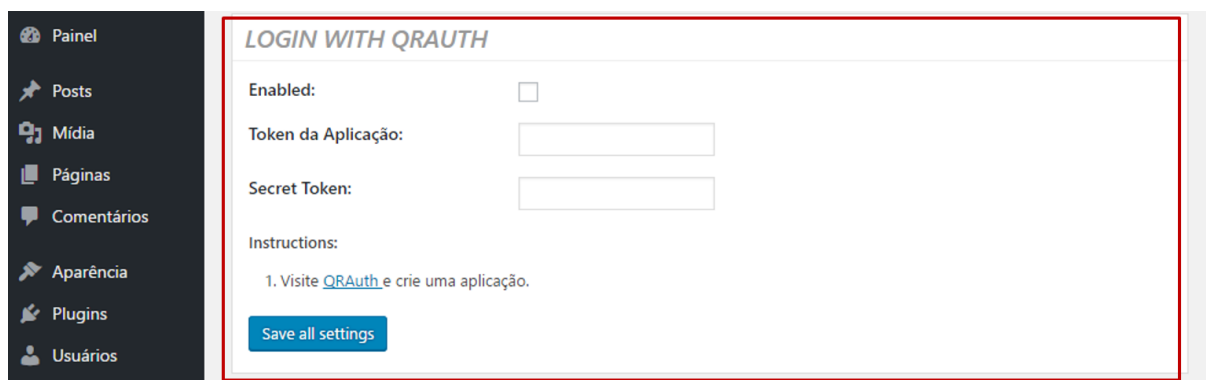


Figura 28 - Bloco referente às configurações do QRAuth no *WordPress*

O próximo passo foi habilitar a opção 'Qualquer pessoa pode se registrar' nas configurações gerais do *WordPress*, permitindo que os usuários possam ser cadastrados automaticamente através das informações providas do *Identity Provider*.

A partir do momento em que todas as configurações foram devidamente realizadas e o *plugin* habilitado, já é possível visualizar um botão descrito como 'Login with QRAuth' na página de login do *WordPress*, conforme pode ser visto na Figura 29. Ao clicar no botão o *plugin* redireciona o usuário para a seguinte URL: `http://endereco_do_idp/login.xhtml?response_type=code&client_id=[APP_TOKEN]&scope=profile&state=[ESTADO]`, o parâmetro **APP\_TOKEN** refere-se ao *token* da aplicação e o **ESTADO** é utilizado para proteger o usuário de ataques onde se utilizam de solicitações falsas entre as aplicações.

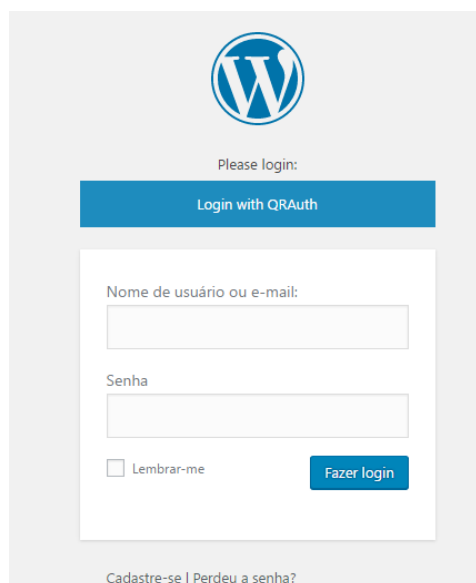


Figura 29 - Página de login do WordPress com o *plugin* WP-Oauth devidamente instalado

Ao ser redirecionado para a página de *Login* (já representada na Figura 17) foi necessário se identificar de alguma forma, ou utilizando a aplicação *mobile*, realizando a leitura do *QR Code* (conforme foi visto na Figura 25), ou através de credenciais de acesso (login/e-mail e senha).

Após se autenticar no *Identity Provider*, como ainda não há vínculo estabelecido entre o usuário e a *Relying Party* WordPress, a tela representada na Figura 26 solicita a permissão para que o *IdP* disponibilize os dados de identificação do usuário para a *RP* solicitante.

Ao clicar em Sim e fornecer o direito de obtenção dos dados por parte da aplicação requisitante, o usuário foi redirecionado para a seguinte URL: `http://endereco_do_rp/?state=5802a1bff19479.69391963&code=b0a13b8035043d58b280`, o parâmetro **state** tem como finalidade verificar se a requisição recebida foi realmente solicitada e o **code** é utilizado pela *relying party* para obter os dados do usuário autenticado naquela sessão em particular.

Para ter acesso aos dados do usuário, a *Relying Party* acessa a seguinte url: `http://endereco_do_idp:/webservice/consume/token/[APP_TOKEN]/[SECRET_TOKEN]/[TOKEN_DE_ACESSO]/` e obtém como resposta um texto no formato *JSON* como apresentado na Figura 30.

```

{
  "nome": "Cristian Dean",
  "token": "2894b1b5279ff67e70455996f778e459",
  "emailPrincipal": "web_cristian@hotmail.com",
  "emailSecundario": "23232",
  "telefonePrincipal": "(63) 33602 4056",
  "telefoneSecundario": "(32) 3",
  "logradouro": "Rua Duque de Caxias",
  "numero": 100,
  "complemento": "CasaÃ",
  "cidade": "Comendador Levy Gasparian",
  "estado": "Rio de Janeiro",
  "pais": "Brasil",
  "expire": 78507916
}

```

Figura 30 - Retorno JSON dos dados da sessão do usuário para a *Relying Party*

Como observado na Figura 30, além dos campos de informações do usuário e seu 'token' de identificação, há também a propriedade '*expire*' contendo o *timestamp* da data que a sessão irá expirar.

Outra forma de obter os dados do usuário sem que uma sessão esteja ativa, é efetuando uma requisição com a passagem de alguns parâmetros através da *url*: `http://endereco_do_idp:/webservice/consume/user/[APP_TOKEN]/[SECRET_TOKEN]/[USER_TOKEN]/`. O parâmetro **USER\_TOKEN** pode ser adquirido através do JSON que contém os dados da sessão do usuário (evidenciado na Figura 30) identificado como '*token*'.

Todos os testes informais propostos foram realizados com êxito no *software* desenvolvido. Além disso, obteve-se como parte dos resultados um *plugin* modificado pronto para se integrar a aplicação QRAuth utilizando o protocolo *OAuth*.

No próximo tópico serão expostas as conclusões levantadas a partir do desenvolvimento deste trabalho.

## 5 CONCLUSÃO

Este trabalho teve como propósito o desenvolvimento de uma ferramenta que permitisse o gerenciamento de identidades de forma centralizada. Por meio desta, foi possível perceber a importância de aplicações deste gênero para os usuários assíduos da internet, dispensando a criação e manutenção de diversas contas em sites diferentes.

Um diferencial do QRAuth em relação a outras aplicações existentes no mercado é a utilização do dispositivo móvel do usuário como um substituto para as senhas textuais, aumentando a segurança em computadores públicos e evitando problemas de esquecimento. Além disso, a interface gráfica do sistema, de forma bem intuitiva, garante que o usuário realize ações facilmente como alterar seus dados, adicionar dispositivos móveis e revogar aplicações, sem a necessidade de conhecimento avançado.

Como principal desvantagem analisada na produção deste trabalho, encontrou-se dificuldade no controle de certas questões, tais como a ação de realizar o encerramento da sessão, por parte do usuário, em aplicações de terceiros e consequentemente sua persistência na aplicação central (*Identity Provider*). Uma solução prática para este tipo de problema seria a especificação de um tempo limite relativamente baixo para o encerramento automático da sessão no provedor de identidade.

Em relação às tecnologias utilizadas, todas se adequaram perfeitamente em seus devidos papéis. A biblioteca de componentes *Primefaces* possibilitou, de forma fácil, gerar *QR Codes* e o *framework Jersey* permitiu a criação do *Web Service* sem muita burocracia.

Após um estudo acerca do protocolo *OAuth* (descrito na seção 2.6.1 FERRAMENTAS E TRABALHOS CORRELATOS) concluiu-se que sua aplicabilidade em sistemas de gerenciamento de identidade centralizados é bem apropriada. Embora o uso do protocolo na versão 2.0 traga uma camada a mais na arquitetura das aplicações, provocando um pequeno atraso, este trouxe também uma série de benefícios para este trabalho, os quais podem ser justificados por sua popularidade, flexibilidade, interoperabilidade e facilidade de troca de informações.

Todos os objetivos apresentados para a realização deste trabalho foram alcançados com êxito. Além disso, algumas propostas foram idealizadas para trabalhos futuros, estas serão apresentadas a seguir:

- Implementação de uma funcionalidade que permita a autenticação biométrica por meio de leitor de digitais em smartphones compatíveis, garantindo uma maior segurança para o usuário.
- Permissão a autenticação em aplicativos móveis presentes nos *smartphones* e *Tablets* que possuam o QRAuth instalado.
- Desenvolvimento de uma biblioteca em *Javascript* que permita ao desenvolvedor integrar sua aplicação ao QRAuth de forma transparente.

Diante do que foi apresentado e de todas as propostas elencadas para a produção deste trabalho, pode-se constatar sua viabilidade até mesmo para fins comerciais, tendo em vista seus benefícios já citados anteriormente além da facilidade de implantação.

## 6 REFERÊNCIAS

AMARAL, B. M.; MAESTRELLI, M. Segurança em Redes. **Centro Brasileiro de Pesquisas Físicas**, Rio de Janeiro, 2004. Disponível em:

<[http://cbpfindex.cbpf.br/publication\\_pdfs/nt00204.2006\\_01\\_30\\_22\\_51\\_07.pdf](http://cbpfindex.cbpf.br/publication_pdfs/nt00204.2006_01_30_22_51_07.pdf)>.

Acesso em: 18 Maio 2016.

ANDROID. Android Open Source Project, 2016. Disponível em:

<<http://source.android.com/>>. Acesso em: 09 Abril 2016.

ARAÚJO, G. Implementação de um Módulo de Autenticação e de Gestão de Perfis de Utilizador. **Faculdade de Engenharia da Universidade do Porto**, Porto, Março 2008. Dissertação (Mestrado em Computação Aplicada). Disponível em:

<[http://paginas.fe.up.pt/~ee07349/dissertacao/material/Guilherme\\_Araujo.pdf](http://paginas.fe.up.pt/~ee07349/dissertacao/material/Guilherme_Araujo.pdf)>.

Acesso em: 21 Abril 2016.

BARROS, A. J. P.; LEHFELD, N. A. D. S. **Fundamentos de metodologia**: um guia para a iniciação científica. 3. ed. São Paulo: Makron, 2000.

BHARGAV-SPANTZEL, A. et al. User Centricity: A Taxonomy and Open. **Journal of Computer Security - The Second ACM Workshop on Digital Identity Management**, Amsterdam, 5 Outubro 2007. 493-527.

BILBIE, A. A review into the uses of OAuth in higher. **Linkey**, Lincoln, Maio 2013.

BRAGA, A. M. Análise Comparativa e Proposta de Extensão à Arquitetura Criptográfica Java. **Biblioteca Digital da UNICAMP**, Campinas, 21 Setembro 1999. Disponível em:

<<http://www.bibliotecadigital.unicamp.br/document/?code=000306422>>. Acesso em: 07 Março 2016. Dissertação (Mestrado em Ciência da Computação).

BRASIL. **Pesquisa Brasileira de Mídia**. Brasília: Secom, 2015. Disponível em:

<<http://www.secom.gov.br/atuacao/pesquisa/lista-de-pesquisas-quantitativas-e-qualitativas-de-contratos-atuais/pesquisa-brasileira-de-midia-pbm-2015.pdf>>. Acesso em: 11 mar. 2016.

CAPURRO, R.; BIRGER, H. Perspectivas em Ciência da Informação, 2007. ISSN 1981-5344. Disponível em:

<[http://www.scielo.br/scielo.php?script=sci\\_arttext&pid=S1413-99362007000100012](http://www.scielo.br/scielo.php?script=sci_arttext&pid=S1413-99362007000100012)>. Acesso em: 19 mar. 2016.

CERT.BR. Cartilha de Segurança para Internet. **Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil**, 2012. Disponível em: <<http://cartilha.cert.br/senhas/>>. Acesso em: 19 Abril 2016.

CERT.BR. Estatísticas dos Incidentes Reportados ao CERT.br. **Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil**, 2015. Disponível em: <<http://www.cert.br/stats/incidentes/>>. Acesso em: 31 mar. 2016.

CERVO, A. L.; BERVIAN, P. A.; SILVA, R. D. **Metodologia Científica**. 6. ed. São Paulo: Pearson, 2006.

CHUANG, J.-C.; HU, Y.-C.; KO, H.-J. A Novel Secret Sharing Technique Using QR Code. **International Journal of Image Processing (IJIP)**, v. 4, n. 5, p. 468-475, Dezembro 2010. ISSN 1985-2304. Disponível em: <<http://www.cscjournals.org/library/manuscriptinfo.php?mc=IJIP-263>>. Acesso em: 10 Abril 2016.

COSTA, N. P. O.; FILHO, N. F. D. Análise e avaliação funcional de sistemas operacionais móveis: vantagens e desvantagens. **Revista de Sistemas e Computação - RSC**, Salvador, v. 3, p. 66-77, 2013. ISSN 2237-2903. Disponível em: <<http://www.revistas.unifacs.br/index.php/rsc/article/view/2581>>. Acesso em: 09 Abril 2016.

D. HARDT, E. The OAuth 2.0 Authorization Framework. **Internet Engineering Task Force (IETF)**, 2012. Disponível em: <<https://tools.ietf.org/html/rfc6749>>. Acesso em: 1 Outubro 2016.

DENSO WAVE INCORPORATED. QR Code, 2014. Disponível em: <<http://www.qrcode.com/en/about/>>. Acesso em: 10 Abril 2016.

DRETSKE, F.. **Knowledge and the flow of information**. Cambridge: MIT, 1981. ISBN 0-262-04063-8.



EMC. **O Universo Digital de Oportunidades: Rich Data e o valor crescente da Internet das Coisas**, 2014. Disponível em: <<https://www.emc.com/leadership/digital-universe/2014iview/executive-summary.htm>>. Acesso em: 13 mar. 2016.

FACEBOOK. Política de Dados, 2015. Disponível em: <<https://www.facebook.com/about/privacy>>. Acesso em: 20 Abril 2016.

FIELDING, R. T. Architectural Styles and the Design of Network-based Software Architectures. **Universidade da Califórnia (UCI)**, Irvine, 2000. Dissertação (Doutorado em Ciência da Informação e Computação). Disponível em: <[https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding\\_dissertation.pdf](https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf)>. Acesso em: 15 Abril 2016.

FLORENCIO, D.; HERLEY, C. A Large-Scale Study of Web Password Habits. **International World Wide Web Conference Committee**, Banff - Canadá, Maio 2007. Disponível em: <<http://research.microsoft.com/pubs/74164/www2007.pdf>>. Acesso em: 19 Abril 2016.

FOROUZAN, B. A. **Comunicação de Dados e Redes de Computadores**. 4. ed. [S.I.]: McGraw Hill, 2008.

GALVÃO, M. D. C. **Fundamentos em Segurança da Informação**. São Paulo: Pearson, 2015.

GARBE, G. **Trilhas em Segurança da Informação - Caminhos e ideias para a proteção de dados**. 1. ed. São Paulo - SP: Brasport, 2015. 70-97 p.

GARTNER. **Gartner Says Worldwide Information Security Spending Will Grow Almost 4.7 Percent to Reach \$75.4 Billion in 2015**, 2015. Disponível em: <<http://www.gartner.com/newsroom/id/3135617>>. Acesso em: 13 mar. 2016.

GIL, C. A. **Como Elaborar Projetos de Pesquisa**. 4. ed. São Paulo: Atlas S.A., 2002.

GOOGLE. Privacidade & Termos, 2016. Disponível em: <<https://www.google.com/intl/pt-BR/policies/privacy/>>. Acesso em: 20 Abril 2016.

IBGE - INSTITUTO BRASILEIRO DE GEOGRAFIA E ESTATÍSTICA. PNAD TIC: em 2014, pela primeira vez, celulares superaram microcomputadores no acesso

domiciliar à Internet, 2016. Disponível em:

<<http://saladeimprensa.ibge.gov.br/noticias.html?view=noticia&idnoticia=3133>>.

Acesso em: 09 abr. 2016.

IDC. Android and iOS Squeeze the Competition, Swelling to 96.3% of the Smartphone Operating System Market for Both 4Q14 and CY14, According to IDC , 2015. Disponível em: <<http://www.idc.com/getdoc.jsp?containerId=prUS25450615>>. Acesso em: 09 Março 2016.

IETF. A Transport Layer Security (TLS) protocolo. **RFC6101**, 2008. Disponível em: <<https://tools.ietf.org/html/rfc6101>>. Acesso em: 08 Abril 2016.

IETF. O Secure Sockets Layer (SSL) Protocol Version 3.0. **RFC5246**, 2011. Disponível em: <<https://tools.ietf.org/html/rfc5246>>. Acesso em: 08 Abril 2016.

JøSANG, A. et al. Trust Requirements in Identity Management. **Proceedings of the 2005 Australasian**, Darlinghurst, 2005. 99-108.

KAO, J. Developer's Guide to Building XML-based Web Services with the Java 2 Platform, Enterprise Edition (J. **TheServerSide**, 2001. Disponível em: <<http://www.theserverside.com/news/1365386/Developers-Guide-to-Building-XML-based-Web-Services-with-the-Java-2-Platform-Enterprise-Edition-J>>. Acesso em: 10 Abril 2016.

KUROSE, J. F.; ROSS, K. W. **Redes de Computadores e a Internet - Uma abordagem Top-Down**. Tradução de Opportunity Translations. 5. ed. São Paulo - SP: Pearson, 2009.

KUZNETSOVA, O. Grow Your App with Account Kit: Give People New Ways to Log In. **Facebook for developers**, 2016. Disponível em: <<https://developers.facebook.com/blog/post/2016/04/12/grow-your-app-with-account-kit/>>. Acesso em: 21 Abril 2016.

LAMPORT, L. Password authentication with insecure communication. **Magazine Communications of the ACM**, Nova Iorque - NY, v. 24, p. 770-772, nov. 1981. ISSN 0001-0782.

LE COADIC, Y.-F. **A ciência da informação**. Brasília: Briquet de Lemos, 1996.

LIAO, I.-E.; LEE, C.-C.; HWANG, M.-S. A password authentication scheme over insecure networks. **Journal of Computer and System Sciences**, Taichung - Taiwan, 1 out. 2005. 728-738. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0022000005001157>>. Acesso em: 27 Março 2016.

MARTINS, R. S. D. M. Composição dinâmica de Web Services. **Universidade do Vale do Rio dos Signos**, São Leopoldo, 2007. Dissertação (Mestrado em Computação Aplicada). Disponível em: <<http://www.professores.uff.br/screspo/rogerio.pdf>>. Acesso em: 14 Abril 2016.

McAFEE. Relatório do McAfee Labs sobre ameaças, 2015. Disponível em: <<http://www.mcafee.com/br/resources/reports/rp-quarterly-threats-aug-2015.pdf>>. Acesso em: 23 mar. 2016.

MENÉNDEZ, A. I. M. Uma ferramenta de apoio ao desenvolvimento de Web Services. **RIUFS - Repositório Institucional da Universidade Federal de Sergipe**, Aracajú, Agosto 2002. Dissertação (Mestrado em Informática). Disponível em: <<https://ri.ufs.br/handle/123456789/1089>>. Acesso em: 10 Abril 2016.

MICROSOFT DEVELOPER NETWORK. Diagramas de classe UML: referência, 2015. Disponível em: <<https://msdn.microsoft.com/pt-br/library/dd409437.aspx?f=255&MSPPErr=-2147217396>>. Acesso em: 11 Outubro 2016.

MORESI, E. A. D. Delineando o valor do sistema de informação de uma organização. **Ciência da Informação**, v. 29, 1 jun. 2000. ISSN 1518-8353. Disponível em: <<http://revista.ibict.br/index.php/ciinf/article/view/246>>. Acesso em: 19 mar. 2016.

MUMBAIKAR, S.; PADIYA, P. Web Services Based On SOAP and REST Principles. **International Journal of Scientific and Research Publication**, v. 3, p. 1-4, Maio 2013. ISSN 2250-3153. Disponível em: <<http://www.ijsrp.org/research-paper-0513/ijsrp-p17115.pdf>>. Acesso em: 14 Abril 2016.

NEWCOMER, E. **Understanding Web Services: XML, WSDL, SOAP and UDDI**. [S.l.]: Addison wesley, 2002.

O'GORMAN, L. Comparing Passwords, Tokens, and Biometrics for User Authentication. **Proceedings of the IEEE**, 08 Novembro 2004.

OAuth. OAuth 2.0, 2012. Disponível em: <<http://oauth.net/2/>>. Acesso em: 21 abr. 2016.

PICCOLO, S. Banco Tecnológico. **Revista Estilo BB**, 2016. Disponível em: <<http://revistaestilobb.com.br/banco-tecnologico/>>. Acesso em: 21 Abril 2016.

PUTASSO, C.; ZIMMERMANN, O.; LEYMANN, F. RESTful Web Services vs. “Big” Web Services. **International World Wide Web Conference Committee**, Beijing, Abril 2008. Disponível em: <<http://www2008.org/papers/pdf/p805-pautassoA.pdf>>. Acesso em: 22 Maio 2016.

ROESNER, F.; KOHNO, T.; WETHERALL, D. Detecting and Defending Against Third-Party Tracking on the Web. **9th USENIX Symposium on Networked Systems Design and Implementation**, Washington, 2012. Disponível em: <<http://www.franziroesner.com/pdf/webtracking-NSDI2012.pdf>>. Acesso em: 04 abr. 2016.

SÊMOLA, M. **Gestão da Segurança da Informação: Uma visão executiva**. 2. ed. [S.l.]: Elsevier, 2014.

SILVA, D. R. P. D. A memória humana no uso de senhas. **Pontifícia Universidade Católica do Sul**, Porto Alegre, Agosto 2007. Tese (Pós-Graduação em Psicologia).

SILVA, R. F. D.; GONÇALVES, P. R. Web Services – Uma Análise Comparativa. **Revista das Faculdades Integradas Claretianas**, São Paulo, n. 4, p. 7-19, Janeiro 2012. Disponível em: <<http://claretianorc.com.br/download?caminho=upload/cms/revista/sumarios/155.pdf&arquivo=sumario1.pdf>>. Acesso em: 15 Abril 2016.

SILVEIRA, A. D. S. A criptografia RSA e DES. **Revista F@pciência**, Apucarana, v. 5, p. 22-25, ago. 2009. ISSN 1984-2333. Disponível em: <[http://www.fap.com.br/fapciencia/005/edicao\\_2009/004.pdf](http://www.fap.com.br/fapciencia/005/edicao_2009/004.pdf)>. Acesso em: 2016 Abril 09.

SILVEIRA, L.; LIMA, W. Q. Um breve histórico conceitual da Automação Industrial e Redes para Automação. **UFRN-PPgEE**, Natal, Maio 2003. Disponível em: <[http://www.dca.ufrn.br/~affonso/FTP/DCA447/trabalho1/trabalho1\\_13.pdf](http://www.dca.ufrn.br/~affonso/FTP/DCA447/trabalho1/trabalho1_13.pdf)>. Acesso em: 10 Abril 2016.

SLAVIERO, M. TLS/SSL and .NET Framework 4.0. **Simple Talk**, 28 Setembro 2011. Disponível em: <<https://www.simple-talk.com/dotnet/.net-framework/tlsssl-and-.net-framework-4.0/>>. Acesso em: 08 Abril 2016.

SOARES, A.; VASCONCELLOS, H. Código de Barras: A presença visível da automação. **Administração de Empresas**, São Paulo, v. 31, p. 59-68, Janeiro 1991. ISSN 0034-7590. Disponível em: <[http://www.scielo.br/scielo.php?script=sci\\_arttext&pid=S0034-75901991000100009&lng=en&nrm=iso](http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0034-75901991000100009&lng=en&nrm=iso)>. Acesso em: 10 Abril 2016.

SOMMERVILLE, I. **Engenharia de Software**. 8. ed. São Paulo: Pearson, 2007.

SOUTO, E.; DOMENECH, M. C.; WANGHAM, M. S. Sistema de Gerenciamento de Identidades para a Rede Catarinense de Informações Municipais baseado no SAML. **XIV Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais**, Belo Horizonte, 2014. 424-433. Disponível em: <<http://www.lbd.dcc.ufmg.br/bdbcomp/servlet/Trabalho?id=22378>>. Acesso em: 27 Maio 2016.

STALLINGS, W. **Cryptography and Network Security**. 5. ed. [S.l.]: Pearson, 2010.

SUN, S.-T. et al. Investigating User's Perspective of Web Single Sign-On. **ACM Transactions on Internet Technolog**, Nova Iorque, 9 Janeiro 2012. Disponível em: <[http://ece.ubc.ca/~santsais/webssso\\_usability\\_journal.pdf](http://ece.ubc.ca/~santsais/webssso_usability_journal.pdf)>. Acesso em: 21 Abril 2016.

TASHI, I.; GHERNAOUTI-HÉLIE, S. Security metrics to improve information security management. **Proceedings of the 6th Annual Security Conference**, Las Vegas, NV, abr. 2007.

VIEIRA, G. Y. M. Projeto de um dispositivo de autenticação e assinatura. **Escola Politécnica da Universidade de São Paulo**, São Paulo, 2007. Dissertação (Mestrado em Engenharia de Computação). Disponível em:

<<http://www.teses.usp.br/teses/disponiveis/3/3141/tde-14012008-162619/pt-br.php>>.

Acesso em: 30 Abril 2016.

W<sup>3</sup>TECHS. World Wide Web Technology Surveys. **W<sup>3</sup>Techs**, 2016. Disponível em:

<<https://w3techs.com/>>. Acesso em: 15 Outubro 2016.

WEIZSÄCKER, C. F. V. **Aufbau der Physik [Foundation of physics]**. Munique:

[s.n.], 1985.

YABE, M. QR Code no Brasil: popularização do que já foi tendência. **adNews**, 2011.

Disponível em: <<http://www.adnews.com.br/artigos/qr-code-no-brasil-popularizacao-do-que-ja-foi-tendencia>>. Acesso em: 10 Abril 2016.