Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus "José Santilli Sobrinho"

GIOVANNI NOBUTI RODRIGUES USSUY

PLATAFORMA DE MICROSERVIÇOS PARA ESTRATÉGIA DE QR-CODE COLORIDO

Assis/SP
2018

Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus "José Santilli Sobrinho"

GIOVANNI NOBUTI RODRIGUES USSUY

PLATAFORMA DE MICROSERVIÇOS PARA ESTRATÉGIA DE QR-CODE COLORIDO

Projeto de pesquisa apresentado ao curso de Ciência da Computação do Instituto Municipal de Ensino Superior de Assis – IMESA e a Fundação Educacional do Município de Assis – FEMA, como requisito parcial à obtenção do Certificado de Conclusão.

Orientando(a): Giovanni Nobuti Rodrigues Ussuy
Orientador(a): Prof. Me. Guilherme de Cleva Farto

Assis/SP
2018

FICHA CATALOGRÁFICA

USSUY, Giovanni Nobuti Rodrigues.
Plataforma de microserviços para estratégia de QR-Code colorido / Giovanni Nobuti Rodrigues Ussuy. Fundação Educacional do Município de Assis –FEMA – Assis, 2018.
58p.

1. Layer Colored QR-Code. 2. Microserviços. 3. API. 4. QR-Code

CDD:
Biblioteca da FEMA

PLATAFORMA DE MICROSERVIÇOS PARA ESTRATÉGIA DE QR-CODE COLORIDO

GIOVANNI NOBUTI RODRIGUES USSUY

Trabalho de Conclusão de Curso apresentado ao Instituto Municipal de Ensino Superior de Assis, como requisito do Curso de Graduação, avaliado pela seguinte comissão examinadora:

Orientador: _____
Prof. Me. Guilherme de Cleva Farto

Examinador: _____
Prof. Dr. Luiz Ricardo Begosso

Assis/SP
2018

DEDICATÓRIA

À minha família, por sua capacidade de acreditar e investir em mim. Mãe, seu cuidado e dedicação foi que deram, em alguns momentos, a esperança para seguir em frente. Pai, sua presença significou segurança e certeza de que não estou sozinho nessa caminhada. E irmão que foi minha visão quando ficava difícil encontrar o caminho.

AGRADECIMENTO

Primeiramente agradeço aos meus pais, pelo amor, incentivo e apoio incondicional. A toda minha família, que nos momentos de minha ausência dedicados ao estudo superior, sempre fizeram entender que o futuro é feito a partir da constante dedicação no presente!

Aos professores da Fundação Educacional do Município de Assis (FEMA/IMESA) pelos conhecimentos transmitidos em sala de aula, em especial ao meu orientador, pelo emprenho dedicado à elaboração deste trabalho.

Por fim a todos que direta ou indiretamente fizeram parte da minha formação, o meu muito obrigado.

Nossas dúvidas são traidoras e nos fazem perder o que, com frequência, poderíamos ganhar, por simples medo de arriscar.

William Shakespeare

RESUMO

Os códigos de barras tornaram-se amplamente populares devido a sua velocidade, precisão e características funcionais de geração e leitura. Tornando os códigos de barras populares e universalmente reconhecidos, assim o mercado começou a exigir novos modelos de códigos capazes de armazenar maior volume de informações, mais tipos de caracteres e um espaço menor de impressão. Como resultado, vários esforços foram feitos para aumentar a quantidade de informações armazenadas. No entanto, tais melhorias têm alguns efeitos negativos, como áreas de código de barras ampliadas, operações de leitura complicadas e aumento dos custos de impressão.

Em virtude das vantagens do QR Code, a proposta deste trabalho é o de modelar e desenvolver uma arquitetura baseada em microserviços para a geração e a manipulação de QR Code colorido. Tal conjunto de serviços possibilitará maior simplicidade na adoção do conceito de Layer Colored QR Code, proposta de expansão da capacidade de armazenamento com multicamadas coloridas em QR Code.

Palavras-chave: QR Code, Microserviços, Layer Colored QR Code.

ABSTRACT

Bar codes have become widely popular because of their speed, accuracy, and functional generation and reading characteristics. Making barcodes popular and universally recognized, so the market began to require new code models capable of storing more information, more types of characters, and smaller print space. As a result, several efforts were made to increase the amount of information stored. However, such improvements have some negative effects, such as extended bar code areas, complicated reading operations, and increased printing costs.

Due to the advantages of QR Code, the purpose of this work is to model and develop a microservice-based architecture for the generation and manipulation of QR Code color. Such a set of services will make it easier to adopt the concept of Layer Colored QR Code, a proposal to expand the storage capacity with colored multi-layer QR code.

Keywords: QR Code, Microservice, Layer Colored QR Code**.**

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

De acordo com dados da IBM (2015), todos os dias são gerados 2,5 quintilhões de *bytes* de dados, sendo que 90% dos dados no mundo são obtidos por meio de sensores climáticos, redes sociais, fotos digitais, vídeos, registros de transações de compras, e outros foram gerados apenas nos últimos dois anos. Esta exorbitante quantidade de dados resulta na geração de novos grandes desafios no que se refere a maneira de manipulação, armazenamento e processamento de dados nas mais variadas áreas da computação (VIEIRA et al, 2012).

Os códigos de barras tornaram-se extremamente populares devido à suas características e vantagens de uso. Tornando os códigos de barras universalmente reconhecidos, assim o mercado começou a exigir novos modelos de códigos com novas propostas para armazenar mais informação, com mais tipos de caracteres e um menor espaço de impressão. Tal que, muitos esforços foram feitos para melhorar e ampliar as características de impressão. No entanto, tais melhorias tiveram alguns efeitos negativos, como áreas de código de barras ampliadas, operações de leitura complicadas e aumento dos custos de impressão (GRILLO et al, 2010).

Os códigos de barras tradicionais, como o EAN-13, são adotados para marcar produtos de varejo, normalmente contêm um número de série como chave para um banco de dados assim diz GS1. No entanto, segundo ONG et al (2010), os sistemas de códigos de barras mais novos têm se tornando bases de dados portáteis ao invés de apenas uma chave. Como resultado, é notável a evolução dos códigos de barras de um lado a outro bidimensional devido à necessidade de uma maior capacidade de dados.

O modelo de código QR Code, proposto em 1994 pela empresa japonesa Denso Wave Incorporated foi aprovado em 2000 como padrão ISO / IEC 18004, é uma estrutura bidimensional usada para transmitir informações por meio de um canal de varredura de impressão. *Quick Response* (QR) significa resposta rápida, uma vez que se destina a ser descodificado em alta velocidade. Conceitualmente, a ideia por trás dessa tecnologia não é muito diferente do código de barras linear, mas sua densidade de dados superior aliada à leitura de alta velocidade tornou muito popular ao longo dos últimos anos. Hoje em dia,

está sendo usado em propagandas, cartões de visita, cartazes de filmes, etiquetas de produtos, e outros (MELGAR et al, 2012).

A globalização da atividade econômica encontra um dos seus maiores exemplos de crescimento na indústria de software de gestão empresarial. Ao contrário do que se imagina, o mercado de ERP (Planejamento de Recurso Corporativo) no Brasil está longe da saturação. Porém uma nova realidade em termos de arquitetura de sistemas está sendo vislumbrada pelos grandes desenvolvedores de software nacional, a arquitetura orientada a serviços (FERNANDES; LIMA, 2008).

- A arquitetura orientada a serviços (SOA) é uma forma de elaborar, desenvolver, implementar e gerenciar sistemas, nos quais:
- Existe uma clara separação entre a interface do serviço e a implementação do serviço.
- Os consumidores de serviços são criados usando a funcionalidade dos serviços disponíveis.
- Uma infraestrutura SOA permite descoberta, composição e invocação de serviços.
- Os protocolos são predominantemente, mas não exclusivamente, trocas de documentos baseadas em mensagens.

Do ponto de vista mais técnico, o SOA é um estilo arquitetônico ou um paradigma de design. Não é uma arquitetura de sistema nem um sistema completo. Os sistemas que são construídos com base nas características SOA listadas acima são chamados de sistemas orientados a serviços (LEWIS, 2012).

Um Web Service é um sistema de software, identificado pelo meio de uma URI (Identificador Uniformes de Recursos), na qual interfaces públicas e contratos são definidos e descritos em XML. Estas definições podem ser descobertas por outros sistemas de software. Estes sistemas podem, então, interagir com o Web Service de uma maneira prescrita pela sua definição, usando mensagens baseadas em XML e transportadas por protocolos da Internet segundo CARTER, 2007 e LEWIS, 2012.

Para a arquitetura do projeto é proposto o Representational State Transfer (REST) introduzido e definido em 2000 por Roy Fielding em sua dissertação de doutorado. REST é um estilo arquitetônico para sistemas de design distribuído. Não é um padrão, mas um conjunto de restrições, como ser sem estado, ter uma relação cliente/servidor e uma interface uniforme (SPRING, 2018).

1.1 OBJETIVOS

O objetivo deste projeto de pesquisa é o de investigar e desenvolver uma plataforma que explore a utilização da API (Interface de Programação de Aplicativos) que fornece um conjunto de métodos e funções para Layer-Colored QR Code. Após uma revisão da literatura sobre as principais áreas desta pesquisa, será modelada e desenvolvida uma plataforma baseada em Microserviços para expor funcionalidades no contexto de QR Code Colorido.

Como resultados, esta pesquisa pretende contribuir com uma evolução na API de Layer-Colored QR, proposta em Ussuy e Farto (2016), bem como concretizar uma plataforma de consumo de WebServices baseados em microserviços.

1.2 JUSTIFICATIVAS

Os códigos de barras são populares e universalmente reconhecidos, assim o mercado começou a exigir novos modelos de códigos capazes de armazenar maior volume de informações, mais tipos de caracteres e um espaço menor de impressão (GRILLO et al., 2010). Mesmo com toda sua magnitude e utilização mundial a tecnologia do código de resposta rápida (*Quick Response* ou QR) tem capacidade de armazenamento limitada, o que pode inibir várias aplicações. Esta nova abordagem torna possível armazenar uma maior quantidade de dados em um QR Code a partir da sobreposição de vários QR Codes conforme apresentado na pesquisa de Ussuy e Farto (2016).

A arquitetura orientada a serviços possibilita a comunicação entre aplicações e plataformas distintas, desde que ambos estejam alinhados com o padrão SOA. Esta facilidade permite a combinação dos serviços providos por estas aplicações compondo novos serviços e funcionalidades.

1.3 MOTIVAÇÃO

O desenvolvimento deste projeto de pesquisa consiste no fato de que o QR Code é um tema ainda pouco explorado e pode contribuir com o futuro desta tecnologia.

Aplicar os conceitos estudados para o desenvolvimento do Microserviço agregará muito valor aos usuários e desenvolvedores, pois terão maior capacidade de armazenamento de dados em um único código QR e facilitara o acesso por meio dessa API para mais pessoas interessadas.

Outra motivação é a chance de, num futuro não muito distante, o mercado de trabalho necessitar de profissionais com conhecimento na linha do tema desta pesquisa, uma vez que a área de desenvolvimento de aplicações web cresce exponencialmente a cada dia.

1.4 PERSPECTIVAS DE CONTRIBUIÇÃO

Ao término desta pesquisa, pretende-se publicá-la na forma de artigos e divulga-la em instituições de ensino e/ou para pessoas com interesse nesta área, com o objetivo de promover e compartilhar os conhecimentos e resultados alcançados. O Microserviço possibilitará o acesso para outras plataformas necessitando apenas do uso da internet, com o objetivo de contribuir para os futuros softwares desenvolvidos.

1.5 METODOLOGIA DE PESQUISA

A proposta e objetivos deste trabalho acadêmico serão alcançados por meio de pesquisas teóricas, de forma a adquirir os conhecimentos necessários por meio da leitura de artigos científicos, livros, monografias, dissertações, teses, guias práticos e técnicos, livros e fontes digitais confiáveis, tornando possível a elaboração e implementação de um serviço web junto de um aplicativo para comprovar a aplicabilidade do modelo proposto.

Primeiramente, serão realizados estudos da arquitetura REST e Microserviços. Em seguida, serão realizados estudos de frameworks utilizadas para o desenvolvimento de aplicativos para Eclipse IDE (Plugin Development Environment). Por fim, será realizada a implementação do aplicativo, testes e validação do mesmo.

1.6 ESTRUTURA DO TRABALHO

A estrutura deste trabalho será composta das seguintes partes:

- **Capítulo 1 – Introdução:** Neste capítulo é contextualizada a área de estudo e apresentará os objetivos, justificativas, motivação, perspectivas de contribuição e metodologia de pesquisa para o desenvolvimento deste trabalho.
- **Capítulo 2 – Representação de dados com *QR Code Colorido*:** Neste capítulo, será apresentado a estrutura formada em volta da tecnologia do código QR colorido.
- **Capítulo 3 – Arquitetura Orientada a Serviços:** Neste capítulo, será discutida a proposta da arquitetura de sistemas orientada a serviços.
- **Capítulo 4 – Proposta de Trabalho**: Neste capítulo, será definida utilizações para o QR code multicolorido.
- **Capítulo 5 – Desenvolvimento do Trabalho:** Neste capítulo, serão apresentados detalhes sobre o funcionamento da ferramenta e seu desenvolvimento.
- **Capítulo 6 – Conclusão:** Neste capítulo, serão revisitados e discutidos as vantagens e desvantagens encontradas durante o desenvolvimento do trabalho.
- **Referências**

2. REPRESENTAÇÃO DE DADOS COM QR CODE COLORIDO

Esta seção tem como objetivo, apresentar a representação de dados em Layer-Colored QR Code, bem como, suas vantagens e benefícios, arquitetura e tecnologias.

2.1 LAYER-COLORED QR-CODE

QR Code (Código de Resposta Rápida do inglês *Quick Response Code*), é uma matriz de símbolos, que consiste na representação gráfica dos dados, por meio de um seguimento de módulos, apresentados no formato de quadrados pretos e brancos, designadamente distribuídos em um quadrado maior, conforme pode ser observado na Figura 1 (ISO, 2015).

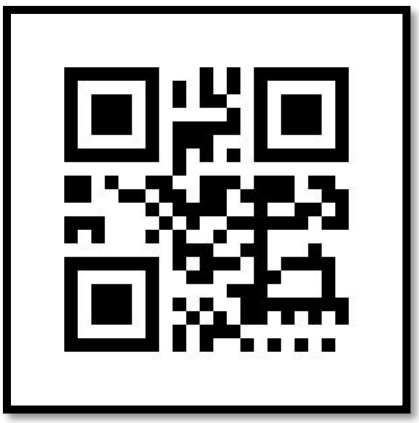

Figura 1: QR Code

Em um QR Code padrão, cada módulo constitui um único bit, seguindo uma regra simples: 1 para quadrados pretos e 0 para quadrados brancos. Com a finalidade de se obter uma maior capacidade para o armazenamento de informações, introduz se cores na representação do QR Code. A maior parte das abordagens com QR Code coloridos utilizam apenas o preto, o branco e mais duas cores, armazenando assim, 4 possíveis informações diferentes: [00, 01, 10, 11] (Ussuy e Farto, 2016).

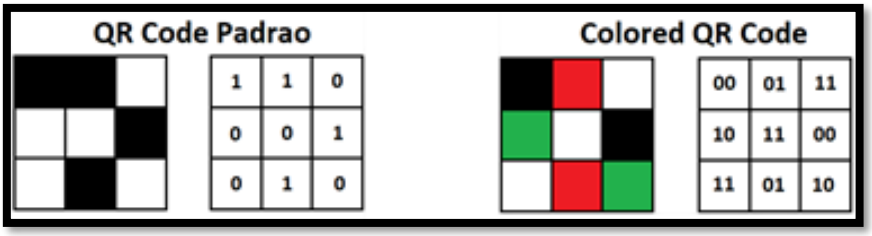

Figura 2: Módulos em Colored QR Code
Fonte: USSUY; FARTO, 2016.

Em Layer Colored QR Code, várias cores podem ser utilizadas, proporcionando uma maior capacidade para armazenamento de dados. Quanto maior a quantidade de cores utilizadas para a geração do Layer Colored QR Code maior será a capacidade de armazenamento de dados (Ussuy e Farto, 2016).

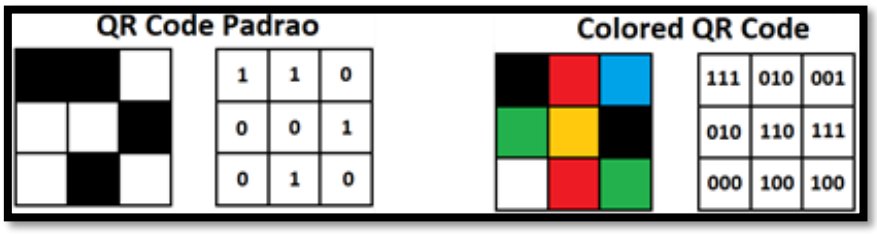

Figura 3: Módulos em Layer Colored QR Code
Fonte: USSUY; FARTO, 2016.

A geração do QR Code multicolorido em Layer Colored QR Code ocorre por meio da sobreposição de QR Codes padrões, aumentando a capacidade de armazenamento a cada sobreposição realizada. A cada sobreposição realizada, aumenta-se o número de cores utilizadas, juntamente com a capacidade de armazenamento. Para a identificação da quantidade de cores e sobreposições realizadas durante o processo de leitura e decodificação, é acrescentado um mapa de cores ao lado da imagem gerada (Ussuy e Farto, 2016).

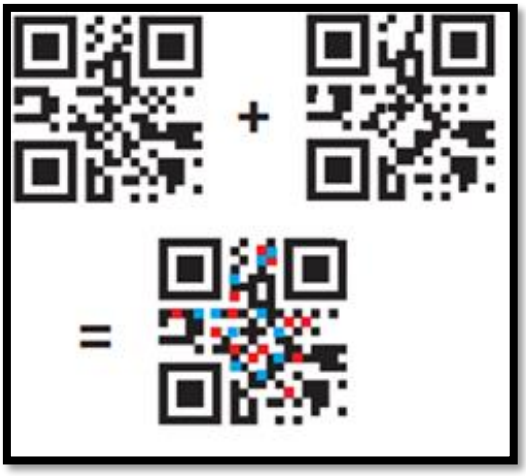

Figura 4: QR Code gerado em Layer Colored QR Code
Fonte: USSUY; FARTO, 2016.

2.2 VANTAGENS E BENEFÍCIOS

Layer Colored QR Code é uma API, ou seja, disponibiliza para outras aplicações um conjunto de funcionalidades com o pretexto de realizar serviços, se tornando uma ferramenta invisível aos clientes finais, de forma geral é composta por uma série de funções acessíveis somente por programação. Esta API foi desenvolvida a partir do modelo de QR Code desenvolvido pela Denso Wave Incorporated, possuindo assim os benefícios já existentes na ferramenta, algum desses benefícios como resistência a sujeita e danos, correção de erros e legibilidade.

Contendo um código para correção de erros, mesmo que um símbolo esteja parcialmente sujo ou danificado, o QR Code pode restaurar os módulos de dados. Com a capacidade de restaurar até 30% dos dados, dependendo do tamanho e modelo do QR Code (GRILLO et al, 2010).

Figura 5: Resistência a sujeira e danos
Fonte: GRILLO et. Al

Existem quatro níveis diferentes para correção de erros. A função de correção de erros é implementada por meio do código Reed-Solomon, que é altamente resistente a rupturas e erros. A partir desta funcionalidade de correção de erros, os códigos podem ser lidos corretamente, mesmo quando eles são manchados ou danificados. O nível de correção de erros pode ser configurado pelo usuário. Aumentar o nível de correção de erros melhora a capacidade de correção, contudo, a quantidade de dados contidos no QR Code também aumenta (SOON, 2008).

Capacidade de correção de erros	
Level L	Aprox. 7%
Level M	Aprox. 15%
Level Q	Aprox. 25%
Level H	Aprox. 30%

Tabela 1: Capacidade de Correção de erros de acordo com o nível
Fonte: QRCODE, 2015

A função de correção de erros é capaz de corrigir dois tipos de erros, módulos que não podem ser escaneados (apagados ou rasurados) e módulos que são convertidos errados. Como o QR Code é uma matriz simbológica, um defeito na conversão de um módulo preto para branco, ou o inverso, afetará na decodificação dos caracteres, apresentando um resultado válido, porém diferente. Devido ao QR ser uma matriz simbológica, são necessários dois módulos corretos para cada módulo convertido erroneamente (ISO, 2000).

Figura 6: QR Code distorcido
Fonte: GRILLO et. Al

O QR Code apresenta uma leitura omnidirecional e de alta velocidade. Mesmo que uma imagem esteja distorcida ou em uma superfície curva, por meio dos padrões de detecção de posição nos três cantos do símbolo, o código pode ser lido a partir de qualquer ângulo dentro de 360º, sem a necessidade de alinhamento entre o scanner e os símbolos do código. Estes padrões de detecção garantem uma leitura rápida e estável, eliminando quaisquer interferências de fundo (DENSO ADC, 2011).

2.3 ARQUITETURA

Com o objetivo de aumentar ainda mais a capacidade de armazenamento de dados, Layer Colored QR Code possibilita a utilização de N cores. A cada sobreposição realizada aumentam-se o número de cores utilizadas e a capacidade de armazenamento de dados

Possibilitando a identificação da versão do Layer Colored QR Code utilizada, juntamente com as cores utilizadas, um mapa de cores será definido para a leitura do QR Code. Na geração do QR Code, uma região específica será reservada para o mapa de cores no canto superior direito, ao lado do QR Code, possibilitando que N cores sejam interpretadas (USSUY; FARTO, 2016).

Figura 7: Exemplo de Layer Colored QR Code
Fonte: USSUY; FARTO, 2016.

O mapa de cores, imagem adicional ao QR Code colorido, representa apenas as cores extras utilizadas (não exibe as cores: branco, preto, vermelho, verde e azul). O mapa é exibido em forma de matriz com 3 cores por linha, com um tamanho de bloco configurável.

Supondo a criação de um Layer Colored QR Code, com 3 *Layers*, apresentando a seguinte sintaxe [*Layer*][*Layer*]*Layer*]. Seguindo a fórmula 2 ^ [Quantidade de *Layers*], 8 cores poderão ser utilizadas: branco (000), preto (111), vermelho (100), verde (010), azul (001), amarelo (110), laranja (101) e rosa (011) (USSUY; FARTO, 2016).

Figura 8: QR Code 3 Layers
Fonte: USSUY; FARTO, 2016.

2.4 TECNOLOGIAS

ZXing ou melhor “Zebra Crossing”, é uma biblioteca de processamento de imagem de código de barras 1D/2D *Open-Source* implementada em Java, com suporte a outras linguagens como: C++, Objective C, JRuby, entre outras (OWEN, 2016).

QRGen é uma API de geração de QRCode para Java criada uma camada acima da ZXing, ela é totalmente dependente da biblioteca ZXing, sendo assim, para a geração de QR Codes em Java são necessários os arquivos da ZXing (GULLAKSEN, 2018).

O QRGen consiste em três módulos: Core, Javase e Android. Ao desenvolver um aplicativo Java, você precisa adicionar o módulo Javase à sua lista de dependências. O módulo principal necessário será adicionado automaticamente pelo seu sistema de compilação (GULLAKSEN, 2018).

3. ARQUITETURA ORIENTADA A SERVIÇO

Segundo Accenture, 2012, SOA é uma arquitetura que define como funções de negócios distintas, implementadas por sistemas autônomos, devem operar conjuntamente para executar um processo de negócio.

Assim, SOA é um modelo de desenvolvimento de software que visa possibilitar que os componentes de um processo de negócio sejam integrados mais facilmente, reutilizando suas funções em diferentes aplicações.

3.1 INTRODUÇÃO

A medida que o tempo passa, nos deparamos com uma existência cada vez maior de sistemas de computação no nosso cotidiano. No entanto, independente da complexidade do sistema, o ponto chave atualmente é a interação entre sistemas. Ao invés de desenvolvermos grandes aplicações como um único bloco, podemos encapsular algumas funções importante e disponibiliza-las na forma de serviços em uma rede (SILVA; KON., 2006; SINGH; HUHNS, 2005).

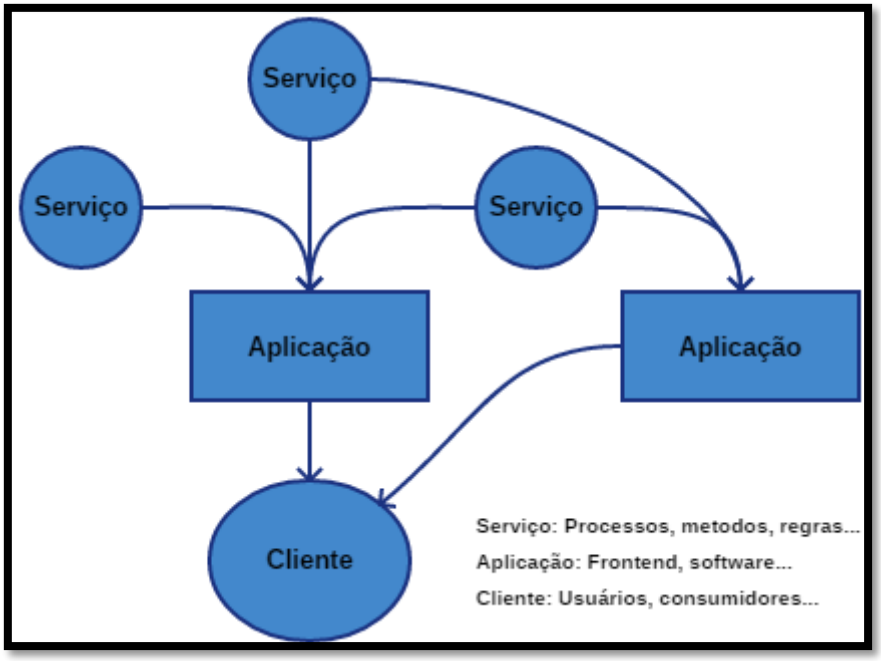

Figura 9: Serviços

3.2 CONCEITOS DE MICROSERVIÇOS

O estilo arquitetural de *Microservices* consiste no desenvolvimento de uma aplicação como um conjunto de pequenos serviços, cada um executando em seu próprio processo. Os serviços podem ser construídos de forma modular, com base na capacidade de negócio da organização em questão.

Pelo fato dos serviços serem construídos de forma independente, a implantação e a escalabilidade podem ser tratadas de acordo com a demanda. Essa independência também permite que os serviços sejam escritos em diferentes linguagens de programação e diferentes tecnologias, visando atender a necessidades específicas da melhor maneira (DEVMEDIA, 2018).

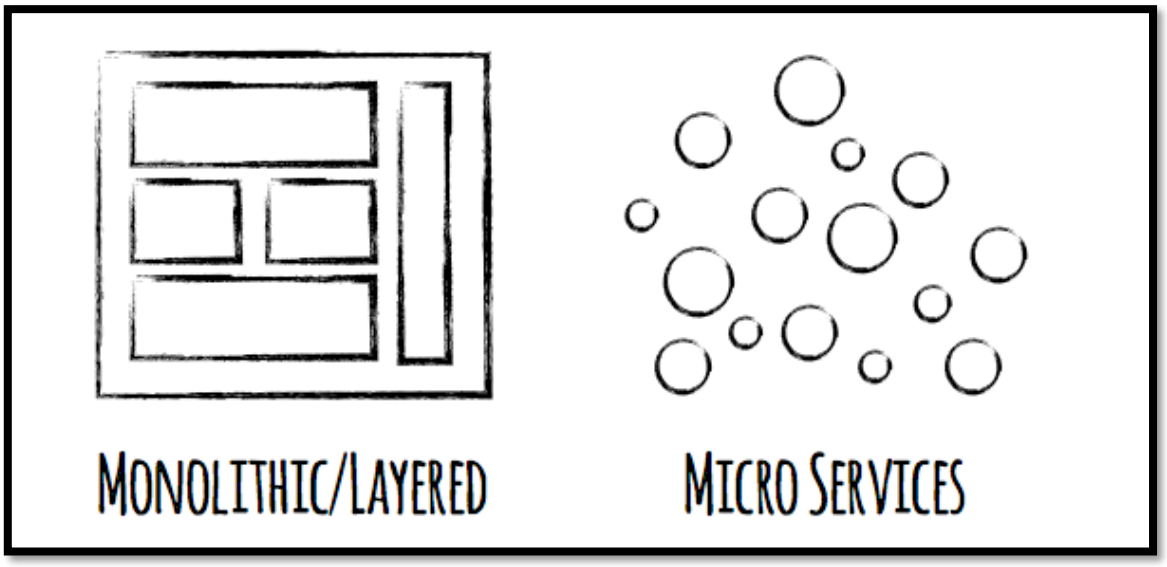

Figura 10: Monolithic vs Microservice
Fonte: DURÂES, 2015

Diferente de uma aplicação monolítica que é basicamente composta por módulos ou funcionalidades agrupadas que, juntas, compõem o software. Os Microserviços são funções separas que compõem um ou mais softwares, a maior vantagem, é a reutilização e a autonomia das regras de negócios, tornando o desenvolvimento livre e independente.

3.3 BOAS PRÁTICAS

A usabilidade de um software é fundamental para seu sucesso. Como programadores precisamos estar atentos a práticas que podem melhorar tanto a experiência do usuário quanto o funcionamento do sistema. É um fator de extrema importância em qualquer sistema, pois diz respeito a como o usuário desempenha suas atividades e como o software reage a isso (DEVMEDIA, 2018).

3.3.1 BAIXO ACOPLAMENTO

Acoplamento significa o quanto uma classe depende da outra para funcionar. E quanto maior for esta dependência entre ambas, dizemos que estas classes estão fortemente acopladas, o forte acoplamento, torna muito custoso a sua manutenção e o seu gerenciamento, pois qualquer mudança pode afetar toda a cadeia de classes (DEVMEDIA, 2018. NEVES, 2017).

De acordo com Silva e Kon, 2006. Cada atividade (função de negócio) é implementada como um serviço, ou seja, um componente independente que poderá ser utilizado quantas vezes for necessário em partes diversas do sistema. Assim, vemos que SOA é, essencialmente, uma arquitetura baseada em componentes, onde cada componente preserva a sua independência e interage, apenas, por meio de interfaces bem definidas. Este é o conceito conhecido como baixo acoplamento e que está presente na orientação a serviços.

3.3.2 INTEROPERABILIDADE

Segundo o Governo Eletrônico, 2017, interoperabilidade pode ser entendida como uma característica que se refere à capacidade de diversos sistemas e organizações trabalharem em conjunto de modo a garantir que pessoas, organizações e sistemas computacionais interajam para trocar informações de maneira eficaz e eficiente.

Um webservice é um componente, ou unidade lógica de aplicação, acessível pelo meio de protocolos padrões de Internet. Como componentes esses serviços possuem uma

funcionalidade que pode ser reutilizada sem a preocupação de como é implementada. Web Services combinam os melhores aspectos do desenvolvimento baseado em componentes e a Web (IWEB, 2003. THOMAZ; CORREIA, 2010).

3.3.3 REST

REST é um dos modelos de arquitetura que foi descrito por Roy Fielding, um dos principais criadores do protocolo HTTP, em sua tese de doutorado e que foi adotado como o modelo a ser utilizado na evolução da arquitetura do protocolo HTTP (FISCHER, 2013).
Muitos desenvolvedores perceberam que também poderiam utilizar o modelo REST para a implementação de Web Services, com o objetivo de se integrar aplicações pela Web, e passaram a utilizá-lo como uma alternativa ao SOAP.
REST na verdade pode ser considerado como um conjunto de princípios, que quando aplicados de maneira correta em uma aplicação, a beneficia com a arquitetura e padrões da própria Web (FERREIRA, 2017).

3.4 EXEMPLOS E BENEFICIOS DE MICROSERVIÇOS

Na Figura 11, temos um exemplo do modo como o fluxo de um software realiza a validação de suas regras implementado utilizando os conceitos relacionados a orientação a serviços. Onde as “Etapas” são providas pelo software que faz a requisição para o serviço do provedor e o mesmo retorna uma resposta.

No nível inferior estão todas dependências que o cliente atualmente utiliza, mantendo-as como serviços, cria a capacidade de mudar apenas as partes necessárias durante a evolução do projeto, possibilitando e facilitando negócios extremamente dinâmicos, alojados em pontos fixos.

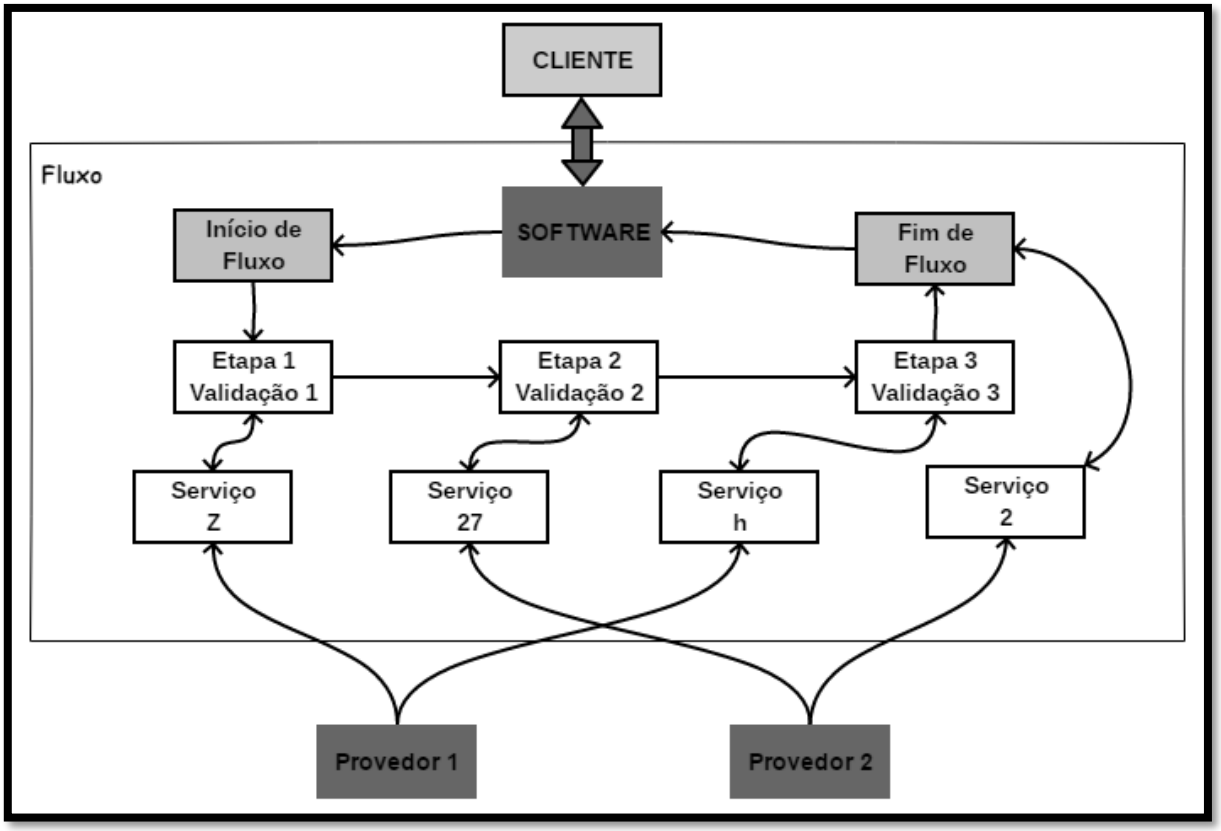

Figura 11: Fluxo com utilização de serviços

Ao criarem a Editora Casa do Código, decidiram seguir outro caminho para o e-commerce. Lá não poderia correr o risco de ficar com a loja fora do ar caso alguma funcionalidade periférica falhe, então quebraram a arquitetura em serviços menores. Dessa forma, há um sistema principal, que é a loja e está hospedada no Shopify, e vários outros sistemas que gravitam em torno dela. Temos uma aplicação que faz a liberação dos e-books para os clientes, outro que contabiliza os royalties para autores, outro para cuidar da logística de envio dos livros impressos para o cliente, um painel de visualização dos livros comprados, um para fazer a liberação de vale presentes e outro para promoções (ALMEIDA, 2015).

Dessa forma, quando um evento acontece na loja online, os diferentes sistemas precisam ser notificados. Para essas notificações usaram requisições HTTP, assim, quando uma compra é confirmada na loja online, todos os sistemas recebem em um *Endpoint* essa requisição HTTP, contendo um JSON com todos os dados da compra. Cada sistema decide o que fazer com as informações recebidas, de acordo com a necessidade (ALMEIDA, 2015).

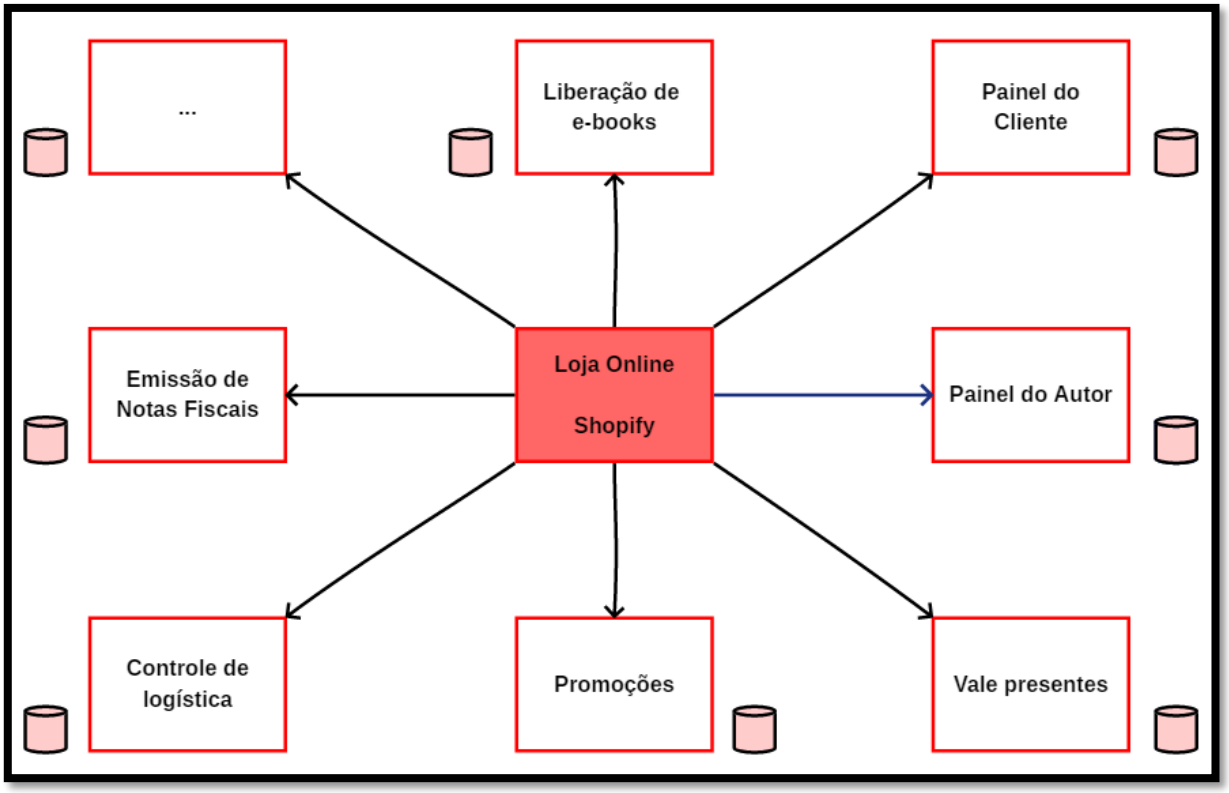

Figura 12: Arquitetura Editora Casa do Codigo
Fonte: ALMEIDA, 2015

3.4.1 REUSABILIDADE

O reuso de recursos está associado à necessidade de adaptação do serviço a diferentes tipos de requisições e ambientes, dando corpo ao conceito de composição de serviços. Utilização de padrões de projeto para aumentar as chances de utilizar determinada funcionalidade em situações distintas, muitas vezes não consideradas no momento da criação.

Principal objetivo é reduzir esforço no desenvolvimento, reduzindo o custo e aumentando a produtividade. Isto se dá pelo fato da construção ser baseada num momento específico do negócio e raramente esta abordagem consegue acompanhar a evolução estratégica desejada (MACHADO, 2004. SILVA; KON, 2006. COSTA; NETO, 2007. THOMAZ; CORREIA, 2010. NEVES, 2017).

3.4.2 AUTONOMIA

A lógica governada por um serviço é circundada por limites explícitos. O serviço possui controle dentro destes limites e não depende de outros serviços para executar sua governança. Este princípio é fortemente influenciado pelo princípio de baixo acoplamento do serviço, pois quanto mais recursos compartilhados o serviço utilizar, menor será sua autonomia para o negócio (SILVA; KON, 2006. THOMAZ; CORREIA, 2010. NEVES, 2017).

4. PROPOSTA DO TRABALHO

Este trabalho tem por objetivo utilizar a API desenvolvida de Layer Colored QR Code em Ussuy e Farto (2016) e modelar uma interface que disponibiliza todas funcionalidades da mesma, o serviço será desenvolvido para, principalmente, facilitar o uso da biblioteca para diferentes recursos e propostas.

O microserviço desenvolvido em Java será utilizado para, principalmente, facilitar a utilização da API de Layer Colored QR Code.

4.1 MICROSERVIÇO

O microserviço será uma página web, que não apresenta interface gráfica. Por meio de uma abordagem de REST e JSON, desta forma REST será utilizado como estilo arquitetural que realiza a comunicação com outras aplicações através do formato JSON, com o objetivo de expandir a capacidade de gerar e ler Layer Colored QR Code, permitindo que quaisquer aplicações que consumam os microserviços realizem a geração e a manipulação de QR Codes coloridos.

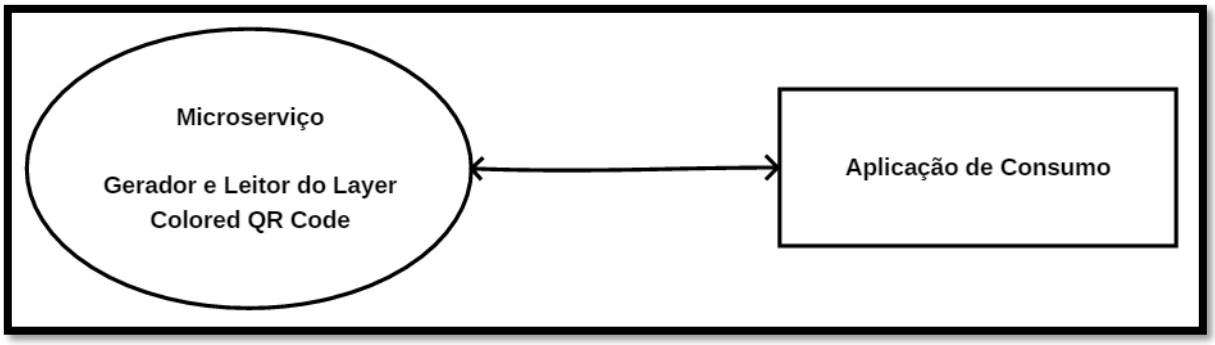

Figura 13: Microserviço

4.1.1 RESTFUL

As condições de uso das aplicações, levando em consideração as situações apresentadas, requerem a adoção de soluções tecnológicas que permitam aos desenvolvedores entregar

os serviços e APIs de forma rápida, confiável e com baixo custo. Uma das tecnologias que permite isso são os Web Services que podem ser definidos como aplicações cliente/servidor utilizando-se da comunicação através da internet, por meio do protocolo HTTP (*HyperText Transfer Protocol*), para prover serviços entre softwares que estejam executando em diferentes plataformas.

Os *Web Services* RESTful (*Representational State Transfer*) são mais adequados para a utilização em cenários mais básicos, e também são melhor adaptados ao uso do protocolo HTTP do que os serviços SOAP. Os serviços RESTful são mais leves, o que significa que podem ser desenvolvidos com menor esforço, tornando-os mais fáceis de serem adotados como parte da implementação de um sistema (DEVMEDIA, 2018).

Desta forma, o modelo arquitetônico a ser utilizado será RESTful, pois o mesmo mantém uma interface limpa e de fácil acesso para futuros consumidores.

4.2 DESCRIÇÃO DA PROPOSTA

O modelo de utilização será uma proposta de API de Layer Colored QR Code, onde será possível preencher cada camada do QR Code a ser gerado. Esta abordagem servirá para facilitar a distribuição e a utilização da API desenvolvida em Ussuy e Farto (2016). Proporcionando a capacidade de estruturar o QR Code multicolorido e molda-lo da melhor forma sob a vontade do utilizador.

A página a ser desenvolvida irá consumir o serviço criado e irá facilitar o uso pelo cliente, tornando a utilização da API como um produto, a Figura 14 exemplifica o modelo de criação de um QR Code.

Ao término do preenchimento das camadas, realizando a requisição, será retornado a imagem em formato Base64, para que possa ser transmitida a imagem por meio de requisições Web, o cliente deve decodificar os *bytes* e transformar o arquivo em uma imagem para visualização.

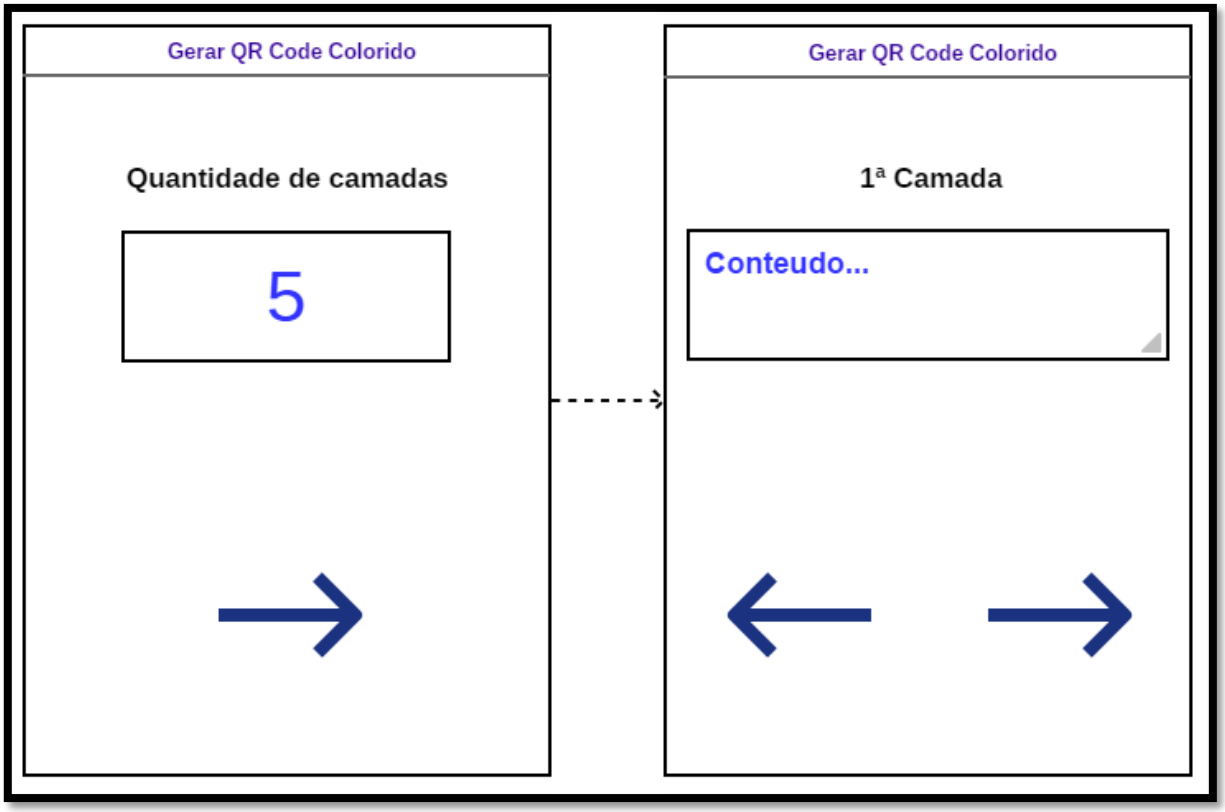

Figura 14: Gerar QR Code por camadas

O *front-end* será uma interface simples e intuitiva para o uso da ferramenta de Layer Colored QR Code. De proposta inicial também será possível colocar todo conteúdo em um arquivo, o conteúdo dimensional, ao realizar o upload do arquivo a gerar, a ferramenta irá utilizar a quantidade de camadas necessárias para compactar o arquivo em um único QR Code colorido.

4.3 CODIFICAÇÃO BASE64

Segundo Garbin (2017), Base64 é um método para codificação de dados e transferência de dados principalmente utilizado na Internet. É constituído por 64 caracteres, sendo eles [A-Z, a-z, 0-9, +, /] que deram origem ao seu nome.

De acordo com o CCM (2017) o princípio da codificação Base64 consiste em utilizar caracteres imprimíveis para codificar qualquer tipo de dado em 8 bits. A codificação Base64 utiliza um alfabeto de 64 caracteres (US-ASCII) imprimíveis clássicos para representar um dado de 6 bits. Os 64 símbolos deste alfabeto são escolhidos para serem universalmente legíveis e para não ter significado nos principais protocolos de serviço de mensagens.

Ao percorrer os dados binários da esquerda para a direita, grupos de 24 bits são criados concatenando blocos de 3 dados de 8 bits. Em seguida, cada grupo de 24 bits é dividido em 4 grupos de 6 bits, correspondente a 4 caracteres do alfabeto Base64. A codificação de dados em Base64 resulta em um espaço de aproximadamente 33% a mais que os dados originais.

Desta forma a codificação Base64 será utilizada no trabalho para realizar a transferências de arquivos entre requisições, assim como na leitura, onde é necessário enviar a imagem para o Microserviço poder decodificar e processar a imagem do QR Code colorido.

4.4 SPRING FRAMEWORK

Spring é um framework de código aberto (*open source*), criado por Rod Johnson, em meados de 2002. Foi criado com o intuito simplificar a programação em Java, possibilitando construir aplicações que antes só era possível utilizando EJB's.

O Spring atualmente possui diversos módulos como Spring Data (trata da persistência), Spring Security (trata da segurança da aplicação) entre outros módulos. Mas o principal (*core*) pode ser utilizado em qualquer aplicação Java, as principais funcionalidades são a injeção de dependência (CDI) e a programação orientada a aspectos (AOP), cabe ao desenvolvedor dizer ao Spring o que quer usar. O que faz dele uma poderosa ferramenta, pois não existe a necessidade de se arrastar todas as ferramentas do framework para criar uma aplicação simples (DEVMEDIA, 2012).

4.5 ANGULAR

O Angular é uma plataforma que facilita a criação de aplicativos com a *Web*. Angular combina modelos declarativos, injeção de dependência, ferramentas de ponta a ponta e práticas recomendadas integradas para resolver desafios de desenvolvimento. Angular permite que os desenvolvedores criem aplicativos que vivem na *Web*, em dispositivos móveis ou na área de trabalho (ANGULAR, 2018).

Segundo Guedes (2017), Angular é um framework para desenvolvimento *front-end* com HTML e TypeScript que, no final, é compilado para JavaScript. Com ele, é possível construir aplicações juntando trechos de código HTML, que são chamados de *templates.* Ao contrário do que algumas pessoas acham, o Angular não é continuação do AngularJS. Angular é um framework totalmente novo e remodelado codificado do zero, com lições apreendidas do AngularJS.

Aplicativos angulares são modulares e o Angular possui seu próprio sistema de modularidade chamado NgModules. Esses NgModules são contêineres para um bloco de

código coeso dedicado a um domínio de aplicativo, um fluxo de trabalho ou um conjunto estreitamente relacionado de recursos. Eles podem conter componentes, provedores de serviços e outros arquivos de código cujo escopo é definido pelo NgModule (Google, 2018).

De acordo com Guedes (2017), o Angular, aproveita o máximo da nova web (com todo o poder do HTML5), usando conceito de *Single Page Application* e sendo base para aplicações *mobiles*. Com ele, podemos fazer desde simples páginas *Web* até grandes sistemas e aplicações, aplicativos *mobile* com Ionic2 - que tem como base de desenvolvimento o Angular.

4.6 BIBLIOTECA LAYER COLORED QR CODE

Será utilizada a biblioteca de Layer Colored QR Code de Ussuy e Farto (2016), para gerar e ler QR Codes coloridos, o serviço será desenvolvido com a framework Spring, utilizando a linguagem de Java para poder utilizar a biblioteca ‘.jar’ de Layer Colored QR Code, a interface do usuário será desenvolvida em Angular pois teve um grande aumento de utilizadores e a mesma facilita o processo de desenvolvimento.

5. DESENVOLVIMENTO DO TRABALHO

Com o objetivo de utilizar e facilitar o uso da API desenvolvida de Layer Colored QR Code em Ussuy e Farto (2016), esta pesquisa apresentou uma maneira simples e universal de manipular e utilizar a geração e leitura de QR Codes coloridos.

Para o desenvolvimento dos Microserviços expostos pela API proposta, foi utilizado o framework Spring Boot que é um projeto dentro do ecossistema Spring, o mesmo fornece dependências “iniciadoras” para simplificar a configuração de compilação, ele consegue isso favorecendo a convenção sobre a configuração, sendo necessário informar ao Spring Boot quais módulos deseja utilizar (Web, *Template*, Persistência, Segurança, etc.) que ele vai reconhecer e configurar.

5.1.1 REQUISIÇÃO PARA GERAÇÃO DE QR CODE MULTICOLORIDO

Para a requisição do serviço de geração do QR Code colorido, por meio do Microserviço implementado com base na abordagem proposta, é necessário enviar uma requisição do método POST para o *Endpoint* “[localhost:8080]/qrcode/create”. Ao efetuar essa solicitação é necessário enviar no corpo da requisição o objeto no formato JSON contendo os textos que serão utilizados para a geração das camadas e demais informações para a geração do QR Code colorido.

```
export class LcqcGerarService {

  lcqcUrl = 'http://localhost:8080/qrcode/create';

  ConfLCQC = [];

  constructor(private http: Http) { }

  create(confLCQC: ConfLCQC): Promise<ConfLCQC> {
    const headers = new Headers();
    headers.append('Content-Type', 'application/json');

    return this.http.post(this.lcqcUrl, JSON.stringify(confLCQC), { headers })
      .toPromise().then(response => response.json());
  }
```

O método “create” nesse trecho de código desenvolvido com plataforma VS Code e a tecnologia Angular, cria um cabeçalho de requisição, informando que o tipo de envio é um JSON e que será requisitado uma chamada com o método POST por meio do protocolo HTTP no *Endpoint* (lcqcUrl = ‘http://localhost:8080/qrcode’) referente a requisição de geração do QR Code com múltiplas camadas.

Os atributos do objeto do JSON devem ter o mesmo nome que o imposto na aplicação para que, ao receber por parâmetro, possa receber corretamente.

```
public class ConfLCQC implements Serializable {

	private static final long serialVersionUID = 1L;

	private List<String> layers;

	private String characterSet;

	private String errorCorrection;

	private Integer margin;

	private Integer colorsPerLine;

	private Integer colorLenght;

	private String colorMap;

	private Integer size;

	private String image;
```

Esses atributos são:

- Layers: Uma lista de “String” que contém as camadas (vários QR Codes) que serão agrupados para gerar um único QR Code.
- CharacterSet: conjunto de caracteres para geração do QR Code.
- ErrorCorrection: nível de correção de erro nos QR Codes informados na Tabela 1.
- Margin: Margem existente entre o QR Code e o mapa de cores.
- ColorPerLine: Cores por linha no mapa de cores.
- ColorLenght: tamanho das cores do mapa de cores.

- ColorMap: mapa de cores para não gerar pelo sistema, deve ser enviado como uma “String” base64 contendo o arquivo “.json” do mapa de cores.
- Size: Tamanho da imagem do Layer Colored QR Code.
- Image: Utilizado para retornar como resposta a imagem criada após o processo de geração, o retorno é realizado em base64 com uma imagem “.png”.

A classe para geração do QR Code multicolorido recebe como parâmetro o corpo da requisição, que deve ser um objeto em JSON contendo os mesmos atributos que as configurações (ConfLCQC) do QR Code. A anotação “@RequestBody” do Spring Framework tem a capacidade de pegar os atributos do corpo da requisição, comparar, criar e preencher um objeto contendo os valores iguais da classe de ConfLCQC, para facilitar a manipulação.

```
	@ApiOperation(value = "Retorna imagem contendo o LCQC")
	@PostMapping("/create")
	public String create(@RequestBody ConfLCQC confLCQC) throws IOException,
	IllegalArgumentException, IllegalAccessException {
		String retorno;
		try {
			retorno = qrCodeService.create(confLCQC);
		} catch (Exception e) {
			retorno = "Ocorreu um erro ao realizar a criacao do qr code";
		}
		return retorno;
	}
```

A classe de controle com o *Endpoint* “/create” é responsável pela geração de QR Codes coloridos, a mesma necessita obrigatoriamente apenas dos “layers” de camadas que serão aglomerados para geração do QR Code colorido, sendo que, se os outros atributos da geração estiverem vazios a classe irá usar um padrão. Ao receber uma requisição o processo de geração faz uma solicitação para o método de geração implementado.

```
// Pasta temporária
		String tempPath = "C:\\Users\\giova\\Documents\\Faculdade\\TCC\\colored-
qrcode\\temp";

		// Limpeza da pasta temporária que armazena os arquivos do processo
```

```
            funcs.clearDirectory(new File(tempPath));

            List<String> inputList = new ArrayList<>();
            List<String> listPath = new ArrayList<>();
            String response = "";


            countLayers = 0;
```

Ao iniciar o processo de processamento dos dados, é gerado uma variável que contém o caminho para armazenamento dos arquivos do processo, a pasta temporária. Recebe o caminho inteiro referente a pasta “/temp” localizada dentro da pasta do projeto. Outras variáveis também são inicializadas que são fundamentais para a criação do Layer Colored QR Code.

Dando continuidade, é listada todas as camadas que foram preenchidas no objeto de configurações do LCQC (confLCQC).

```
// Listagem das camadas
            confLCQC.getLayers().forEach(l -> {
                  if (l != null && !"".equals(l)) {
                        inputList.add(l);
                        listPath.add(tempPath + "\\QRCode" + countLayers + ".png");
                        countLayers++;
                  }

            });
```

Nesta etapa serão listadas apenas as informações que estiverem preenchidas dentro do *array,* ignorando partes vazias ou nulas. Na geração do LCQC apenas as camadas necessitam estarem preenchidas, pois durante o processo são recuperados padrões afim de evitar erros.

```
      LayerColoredWriter lcw = new LayerColoredWriter();

            // Métodos de codificação
            if (!"".equals(confLCQC.getCharacterSet().trim()))
                  lcw.setCharacterSet(confLCQC.getCharacterSet());
            else
                  lcw.setCharacterSet("UTF-8");

            if (confLCQC.getMargin() != null && confLCQC.getMargin() > 0)
                  lcw.setMargin(confLCQC.getMargin());
            else
```

```
                lcw.setMargin(2);

            if ("L".equals(confLCQC.getErrorCorrection()))
                lcw.setErrorCorrection(ErrorCorrectionLevel.L);
            else if ("M".equals(confLCQC.getErrorCorrection()))
                lcw.setErrorCorrection(ErrorCorrectionLevel.M);
            else if ("Q".equals(confLCQC.getErrorCorrection()))
                lcw.setErrorCorrection(ErrorCorrectionLevel.Q);
            else
                lcw.setErrorCorrection(ErrorCorrectionLevel.H);
```

Nesse trecho de código ainda no método “*create”* é instanciado um objeto do tipo “*LayerColoredWriter()”* referente a API de Layer Colored QR Code para geração do QR Code colorido, onde são informados os métodos para a codificação das camadas. Caso as informações recebidas não estejam preenchidas, é recuperado um padrão para não ocorrer erros durante o processo. Sendo “UTF-8” padrão para o “CharacterSet” com medida de margem igual a dois e nível de correção “H”.

Após a configuração dos métodos de codificação é preenchido as definições do mapa de cores.

```
// Definições do mapa de cores
            ColorMapSettings colorMapSettings = new ColorMapSettings();

            if (confLCQC.getColorsPerLine() != null && confLCQC.getColorsPerLine() >0)
                colorMapSettings.setColorsPerLine(confLCQC.getColorsPerLine());
            else if (countLayers > 16)
                colorMapSettings.setColorsPerLine(countLayers * countLayers * 2);
            else
                colorMapSettings.setColorsPerLine(countLayers * countLayers);

            if (confLCQC.getColorLenght() != null && confLCQC.getColorLenght() > 0)
                colorMapSettings.setColorLenght(confLCQC.getColorLenght());
            else
                colorMapSettings.setColorLenght(2);

            if (confLCQC.getColorMap() != null && !confLCQC.getColorMap().isEmpty()) {
                String colorMapPath =
funcs.GenerateColorMapJson(confLCQC.getColorMap(), countLayers);

                if (colorMapPath == "ERRONADECODIFICACAO") {
                    lcw.generatePallete(countLayers);
                } else {
                    lcw.generatePalleteByJson(colorMapPath);
                }
            } else {
                lcw.generatePallete(countLayers);
```

O valor do mapa de cores deve ser enviado em formato base64 contendo um arquivo “.json” com as informações das combinações de cores e a quantidade exata em relação a quantidade de camadas a serem geras. Se o mapa enviado for invalido para a geração, será criado um de acordo com a criação do LCQC. Neste trecho também é definido o tamanho das cores do mapa, junto da quantidade por linhas.

```
try {

                    // Gera QRCodes zxhing
                    lcw.generateQRCode(inputList);

                    // Pasta para gerar o QRCode colorido
                    lcw.setColoredOutputPath(tempPath + "\\ColoredQRCode.png");

                    // Gerar QRCode colorido
                    if (confLCQC.getSize() < 100)
                          lcw.generateColoredQRCode(listPath, 400);
                    else
                          lcw.generateColoredQRCode(listPath, confLCQC.getSize());

                    // Transforma a imagem gerada em png para base64
                    File file = new File(tempPath + "\\ColoredQRCode.png");
                    FileInputStream fis = new FileInputStream(file);
                    byte[] data = new byte[fis.available()];
                    fis.read(data);
                    fis.close();
                    file.delete();

                    confLCQC.setImage(Base64.getEncoder().encodeToString(data));

                    Gson gson = new Gson();

                    response = gson.toJson(confLCQC);

              } catch (Exception e) {
                    response = e.getLocalizedMessage();
              }

              // Retorna a imagem em base64 ou erro
              return response;
```

Após as configurações do Layer Colored QR Code estarem definidas, é realizado a geração dos QR Codes junto da criação do QR Code multicolorido, e então a imagem gerada é codificada para base64 em uma variável, essa variável do tipo “String” é transformada em formato JSON e enviada como resposta.

Para utilização do Web Service desenvolvido foi implementado uma página *Web* SOFEA que explana as utilizações do Layer Colored QR Code. Na Figura 15 é apresentado o *front-end* desenvolvido em Angular, estando separado do *back-end* é conectado em uma outra porta de aplicação.

A página de geração contém um formulário para preenchimento com as configurações e as camadas do Layer Colored QR Code. Ao completar o preenchimento e clicar no botão para gerar (flutuando ao lado da lixeira vermelha) é enviado a requisição para o método “*create*” em “LcqcGerarService” contendo as informações do formulário que ao término apresenta o QR Code colorido acima das configurações. Conforme representado na Figura 15, onde é possível visualizar todas informações para geração do QR Code multicolorido.

Figura 15: Interface de usuário para geração de LCQC

5.1.2 SWAGGER-UI

De acordo com Swagger (2018), a Swagger UI é um projeto de código aberto para renderizar visualmente a documentação de uma API, permite que qualquer pessoa esteja em sintonia com o desenvolvimento, facilitando a visualização e interação com os recursos da API sem ter nenhuma das lógicas de implementação conhecida. É gerado automaticamente a partir de sua especificação OpenAPI, com a documentação visual facilita a implementação para teste de *back-end* e consumo do lado do cliente

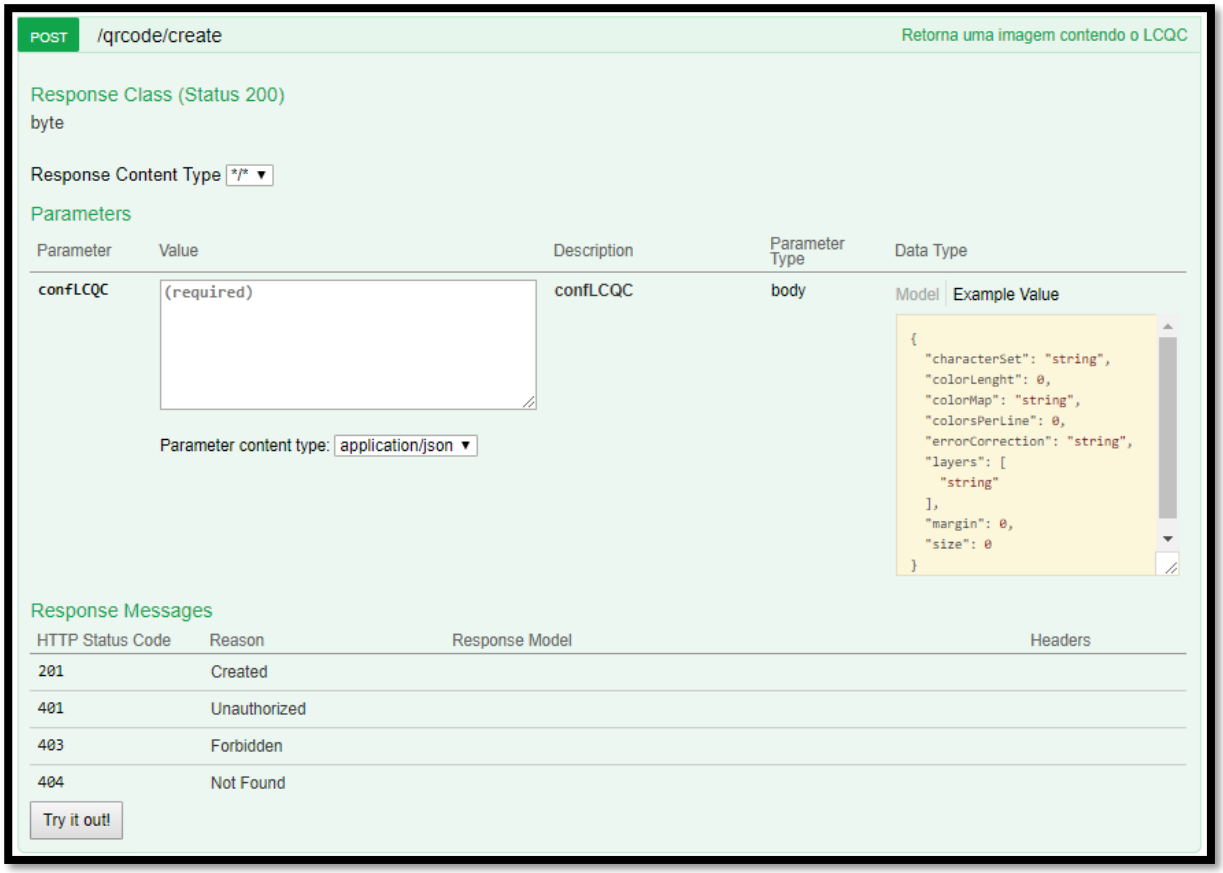

Figura 16: Swagger geração

A visualização do Swagger UI permite um rápido entendimento sobre a API e facilita o uso e teste da mesma. Na Figura 16 é possível notar que existe um campo amarelo contendo o modelo do objeto e outras informações necessárias para realizar a requisição de geração do Layer Colored QR Code, desta forma facilita o uso do Serviço por parte externa dos usuários.

5.2.1 REQUISIÇÃO PARA LEITURA DE UM QR CODE MULTICOLORIDO

Da mesma forma que a requisição para geração do QR Code multicolorido, por meio do Microserviço implementado, é necessário enviar uma requisição do método POST para o *Endpoint* “[localhost:808]/qrcode/read”, o motivo de ter dois metodos POST vem da necessidade de enviar arquivos, onde o create envia o objeto para gerar e a leitura envia o arquivo imagem para ler. Ao realizar essa requisição é necessário enviar no corpo dela em formato JSON o arquivo imagem para a leitura do mesmo, nesse momento é necessário converter a imagem para um arquivo “criptografado” pois não existe maneira de enviar a imagem inteira e também seria muito custoso, desta forma é utilizado o formato “FormData” que fornece uma maneira fácil de construir um conjunto de pares chave/valor representando campos de um elemento, para ser possível envia-lo como um parâmetro da requisição.

```
export class LcqcLeitorService {

  lcqcUrl = 'http://localhost:8080/qrcode/read';

  constructor(private http: HttpClient) { }

  read(file: File): Observable<HttpEvent<{}>> {
    const headers = new HttpHeaders();
    headers.append('Content-Type', 'application/json');
    const formdata: FormData = new FormData();
    formdata.append('file', file);

    const req = new HttpRequest('POST', this.lcqcUrl, formdata, {
      reportProgress: true,
      responseType: 'json'
    });
    return this.http.request(req);
  }
```

Outra semelhança com a geração é a necessidade do header na requisição para informar que o formato de mensagem é JSON.

```
@ApiOperation(value = "Retorna lista contendo a leitura do LCQC")
```

```
	@PostMapping("/read")
	public String read(@RequestParam("file") MultipartFile file) throws IOException {
		String retorno;
		try {
			retorno = qrCodeService.read(file);
		} catch (Exception e) {
			retorno = "Ocorreu um erro ao realizar a leitura do qrCode";
		}
		return retorno;
	}
```

A classe para a leitura dos códigos de resposta rápida colorido, recebe o arquivo imagem e retorna à tradução do mesmo.

```
public String read(MultipartFile file) throws IllegalStateException, IOException {
		// Pasta temporária
		String imagePath = "C:\\Users\\giova\\Documents\\Faculdade\\TCC\\colored-
qrcode\\temp";

		// Limpa a pasta temporaria
		funcs.clearDirectory(new File(imagePath));

		// Converte e grava o arquivo
		File pasteFile = new File(imagePath + "\\ColoredQRCode.png");
		file.transferTo(pasteFile);

		// Caminho para leitura da imagem recebida
		LayerColoredReader lcr = new LayerColoredReader();
		lcr.readPallete(imagePath + "\\ColoredQRCode.png");
		lcr.setOutputPath(imagePath);

		// Controi um JSON contendo uma lista da leitura do QR Code colorido
		Gson gson = new Gson();
		String responseList = gson.toJson(lcr.readLayerColored());

		return responseList;
	}
```

Ao receber a imagem, é convertida para formato “.png” e guardada na pasta temporária, utilizando a imagem que está na pasta temporária, é realizado o processo de tradução, que após o término, retorna uma lista contendo as camadas que estavam acopladas no Layer Colored QR Code.

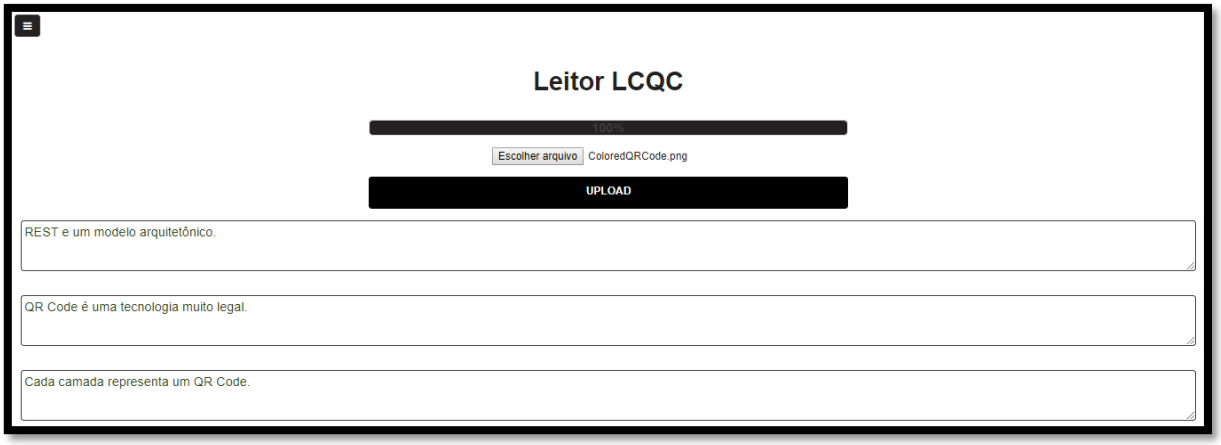

Figura 17: Interface de usuário para leitura de LCQC

A interface para leitura do QR Code colorido recebe o arquivo, aceitando apenas imagem, ao realizar o processo de leitura apresenta áreas de texto contendo as camadas de QR Codes. O exemplo da Figura 17 apresenta a leitura de um Layer Colored QR Code de 3 camadas.

5.2.2 SWAGGER-UI DE LEITURA

A interface para leitura do QR Code colorido fica facilmente visível por meio da ferramenta Swagger UI onde apresenta os elementos para realizar a requisição.

Figura 18: Swagger leitura

Nos parâmetros da requisição é necessário o arquivo imagem do QR Code colorido para possível decodificação do mesmo, o arquivo é obrigatório. Outros valores são apresentados como mensagens de resposta para possíveis erros no momento de execução da leitura.

5.3 MAPA DE CORES COMO CHAVE DE SEGURANÇA

Ao realizar uma requisição para geração do LCQC, se o arquivo “ColorMap” for enviado, a plataforma interrompe a geração do mapa de cores pela API, retornando a imagem do QR Code sem o mapa de cores, tornando necessário sempre que for realizar a leitura desse QR Code inserir o arquivo com o mapa que contém as combinações de cores.

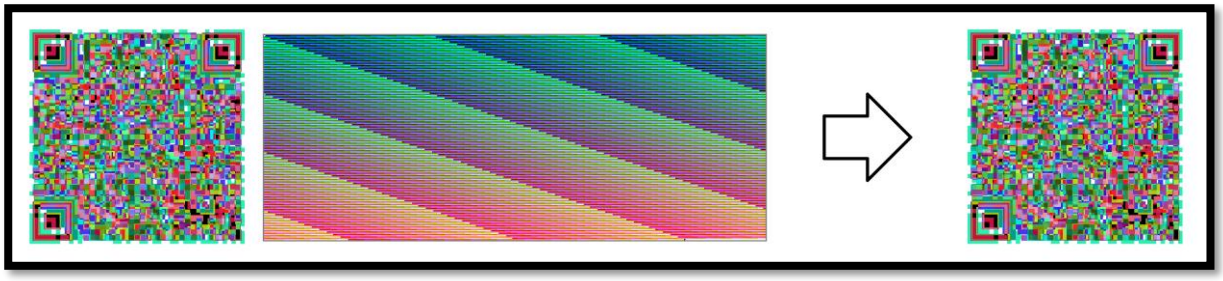

Figura 19: Layer Colored QR Code sem mapa de cores

No LCQC para cada sobreposição de QR Code, são necessárias 2 ^ [*Layers*] de cores para sua escrita, uma para cada combinação de bit onde um LCQC com 3 *layers*, utilizara 8 cores para sua escrita, uma para cada combinação de bit na matriz do QR Code. Ao esconder o mapa de cores, é possível elevar o nível de segurança das informações contidas. Utilizando o mapa de cores como uma chave para a criptografia das informações foi possível aumentar o nível de segurança das informações contidas no QR Code. Tornando o LCQC seguro contra leitores indesejados.

Realizando uma equação matemática para identificar a quantidade de combinações possíveis de cores ao tentar interpretar o mapa de cores do LCQC, excogitando um cenário onde a ferramenta que estiver realizando a tarefa para quebrar a criptografia, por exemplo, de um QR Code colorido com 4 *layers*, tiver a capacidade de analisar todas as cores

utilizadas na imagem, sabendo que o bit menos significativo sempre será branco e que todas as cores possíveis estão visíveis sobre os bits mais significativos do QR Code colorido, é realizado a operação de arranjo simples da quantidade de cores menos uma que é a branca, chegando a um total com mais de 1 trilhão (1.307.674.368.000) de combinações.

$$\mathrm{An, p} = \frac{\mathrm{n!}}{(\mathrm{n} - \mathrm{p})!}$$

- N = Quantidade total de elementos no conjunto.
- P = Quantidade de elementos por arranjo.

Mas se for reavaliado o fato que ocorre constantemente onde nem todas as cores estão visíveis ao topo do LCQC, temos um drástico aumento nesses números. O padrão RGB que é utilizado para gerar os QR Codes multicoloridos contém mais de 16 milhões (16.777.216 ou 256^3) de cores diferentes, desta forma a quantidade de combinações deve levar em consideração todas possibilidades possíveis sempre sobre 23 camadas. A quantidade de combinações torna-se absurda, assim temos aproximadamente a equação de arranjo simples da quantidade de cores menos uma que é a branca, vezes a quantidade de cores possíveis, realizado em sequência para cada quantidade de camadas estipulada:

$$\left(\frac{7!}{(7-7)!}\ x\ 256^3\right) + \left(\frac{15!}{(15-15)!}\ x\ 256^3\right) + \cdots + \left(\frac{8.388.607!}{(8.388.607 - 8.388.607)!}\ x\ 256^3\right)$$

6. CONCLUSÃO

6.1 CONSIDERAÇÕES FINAIS

Microserviços facilitam muito a vida de seus utilizadores, não só para o criador, como também para os consumidores, aliado a boas práticas é capaz de criar ferramentas com estratégias de reutilização para diversos modos de uso, aprimorando e ampliando o uso de diversas fontes.

REST na verdade pode ser considerado como um conjunto de princípios, que quando aplicados de maneira correta em uma aplicação, a beneficia com a arquitetura e padrões da própria Web (FISCHER., 2013, FERREIRA., 2017).

Desta forma REST pode ser entendida como o padrão do protocolo HTTP levando em consideração que esse modelo de arquitetura foi descrito por Roy Fielding, um dos principais criadores do protocolo HTTP.

Com o grande aumento dos dados no contexto de *Big Data*, a internet pode ser considerada a melhor solução para esse fim, dando importância ao processamento, essa necessidade pode ser resolvida com grandes servidores de aplicação.

A estratégia de QR Codes tem diversas utilidades e seu crescimento global é imenso e a cada dia, é possível notar que o uso desta adoção vem aumentando. A estratégia de Layer Colored QR Codes é rica de valores e abrange ainda mais a capacidade de armazenamento.

De acordo com Ussuy e Farto (2016) em Layer Colored QR Code, várias cores podem ser utilizadas, proporcionando uma maior capacidade para armazenamento de dados. Quanto maior a quantidade de cores utilizadas para a geração do Layer Colored QR Code maior será a capacidade de armazenamento de dados.

O fato de movimentar a API de Layer Colored QR Code para Web traz também uma grande melhora na velocidade de leitura e geração, pois se tratando de estar em uma rede, o servidor que estiver mantendo a API para acesso de todos, deve conter uma ou várias máquinas que estarão preparadas para múltiplas requisições e com grande velocidade de processamento.

6.2 TRABALHOS FUTUROS

Existem fatores ainda a serem melhorados em relação a API de Layer Colored QR Code e o *Web Service*, melhorando a performance do serviço e adicionando novas funcionalidades a estratégia de QR Code multicoloridos ou melhorando a segurança da API RESTful.

Em trabalhos futuros para uma melhor utilização dos recursos de requisição Web, a aplicação de *Web Services* pode ser adaptada para utilizar as funcionalidades da linguagem GraphQL ou RestQL como estratégia de Microserviços.

Além disso, trabalhos futuros podem experimentar a plataforma desenvolvida em contextos como internet das coisas, para identificação de objetos, tornando possível e mais viável o desenvolvimento de APIs mobile, para identificação de QR Codes multicoloridos.

REFERÊNCIAS

ACCENTURE, **Accenture Service Oriented Architecture.** Dublin, Irlanda. 2012.

ALMEIDA, Adriano; **Arquitetura de microserviços ou monolítica?** 2015.Disponível em: <http://blog.caelum.com.br/arquitetura-de-microservicos-ou-monolitica/> Acesso em: 15 de mar. 2018.

ANGULAR. **What is Angular?.** 2018. Disponível em: < https://angular.io/docs> Acesso em: 06 de ago. 2018.

CARTER, S. **The new language of business: SOA & Web 2.0.** EUA, IBM Press, ISBN 0-13-195654-X, 2007.

CCM. **Codificação Base64.** 2017. Disponível em: <https://br.ccm.net/contents/55-codificacao-base64> Acesso em: 01 de ago. 2018.

COSTA, Ivanir; NETO, Antonio Rodrigues Carvalho. **Tendências sobre a arquitetura orientada a serviços – SOA.** Anais do XXVII Encontro Nacional de Engenharia de Produção – ENEGEP, Foz do Iguaçu. 2007. 9p

DENSO ADC. **QR Code Essentials.** 2011. 8p. Disponível em: <http://denso-adc.com/download>. Acesso em: 21 de out. 2017.

DEVMEDIA. **DevCast: Boas práticas de usabilidade**. Disponível em: <https://www.devmedia.com.br/boas-praticas-de-usabilidade/38763>. Acesso em: 05 de mar. 2018.

DEVMEDIA. **Entendendo Coesão e Acoplamento.** Disponível em: < https://www.devmedia.com.br/entendendo-coesao-e-acoplamento/18538>. Acesso em: 26 de jan. 2018.

DEVMEDIA. **Introdução à arquitetura de microservices com Spring Boot.** Disponível em: <https://www.devmedia.com.br/introducao-a-arquitetura-de-microservices-com-spring-boot/33703#> Acesso em: 10 de mar. 2018.

DEVMEDIA, **Introdução a web services RESTful,** Disponível em: <https://www.devmedia.com.br/introducao-a-web-services-restful/37387>. Acesso em: 17 de mar. 2018.

DEVMEDIA, **Introdução ao Spring Framework,** Disponível em: < https://www.devmedia.com.br/introducao-ao-spring-framework/26212>. Acesso em: 05 de ago. 2018.

DURÃES, Ramon. **Introdução ao conceito de Microservices**. 2015. Diponível em: <http://www.ramonduraes.net/2015/05/10/introduo-ao-conceito-de-microservices/> Acesso em: 14 de mar. 2018

FERNANDES, Marcelo Eloy; LIMA, Carlos Roberto Camello. **Um estudo sobre aplicação da arquitetura orientada a serviços - SOA em uma software house.** Engepep. 10p. 2008.

FERREIRA, Rodrigo. **REST: Princípios e boas práticas.** Caelum. 2017

FISCHER, Ludovico. **Guia de introdução aos conceitos HTTP e REST.** 2013. Disponível em: <https://code.tutsplus.com/pt/tutorials/a-beginners-guide-to-http-and-rest--net-16340> Acesso em: 14 de mar. 2018.

GARBIN, Wolmir Cezer. **O que é Base 64 e para que serve?.** 2017. Disponível em: < https://receitasdecodigo.com.br/web/o-que-e-base-64-e-para-que-serve> Acesso em: 01 de ago. 2018.

GOVERNO ELETRÔNICO; **Padrões de Interoperabilidade.** São Paulo. 2017

GRILLO, Antonio; LENTINI, Alessandro; QUERINI, Marco; ITALIANO, Giuseppe F. **High Capacity Colored Two Dimensional Codes.** Proceedings of the International Multiconference on Computer Science and Information Technology. 2010. 709p.

GS1. **EAN-13 and EAN-8.** Disponível em: < http://www.gs1.se/en/our-standards/Capture/ean-13-and-ean-8/> Acesso em: 04 de ago. 2018.

GUEDES, Thiago. **Crie aplicações com Angular: O novo framework do Google.** Casa do Código. 5 de maio de 2017.

GULLAKSEN, Ken. **QRGen: a simple QRCode generation api for java built on top ZXING**. 2018. Disponível em: <https://github.com/kenglxn/QRGen>. Acesso em: 19 de mar. 2017.

IWEB. **Web Services.** 2003. 3p. Disponível em: <http://www.iweb.com.br/iweb/home.php>. Acesso em: 03 de mar. 2018.

LEWIS, Grace. **Getting Started with Service Oriented Architecture (SOA) Terminology**. 2012. 8p. Disponível em: <https://www.sei.cmu.edu/library/assets/whitepapers/SOA_Terminology.pdf> . Acesso em: 13 de nov. 2017.

MACHADO, João Coutinho. **Um estudo sobre o desenvolvimento orientado a serviços.** 89p. Dissertação de Mestrado. PUC-Rio. 2004

MELGAR, Vizcarra; ZAGHETTO, Alexandre; MACCHIAVELLO, Bruno; NASCIMENTO Anderson C. A. **CQR CODES: COLORED QUICK-RESPONSE CODES.** Universidade de Brasília. 2012. 5p.

ONG, Siong Khai; CHAI, Douglas; RASSAU, Alexander. **The Use of Alignment Cells in MMCC Barcode.** Edith Cowan University, Austrália. 2010. 5p.

NEVES, Raphael Oliveira. **SOA: princípios de projetos orientados a serviço.** 2017. Disponível em: <https://www.profissionaisti.com.br/2017/05/soa-principios-de-projetos-orientados-a-servico/>. Acesso em: 09 de mar. 2018.

ONO, Satoshi; NAKAYAMA, Shigeru**. A System for Decorating QR Code with Facial Image Based on Interactive Evolutionary Computation and Case-Based Reasoning.** Department of Biomedical Engineering and Information Science, Graduate School of Science and Engineering, Kagoshima University. 2010. 6p.

OWEN, Sean. **Barcode Contents.** 2016. Disponível em: <https://github.com/zxing/zxing/wiki/Barcode-Contents>. Acesso em: 20 de fev. 2018.

OWEN, Sean. **Official ZXing ("Zebra Crossing").** 2017. Disponível em: <https://github.com/zxing/zxing> . Acesso em: 20 de fev. 2018.

P.S. André; R.A.S. Ferreira. **Colour multiplexing of quick-response (QR) codes**. ELECTRONICS LETTERS, EUA, 20 nov. 2014, Vol. 50 No. 24 pp. 1828-1830.

SILVA, Filipe Madeira; KON, Fábio. **SOA – Arquitetura Orientada a Serviços.** Trabalho de Conclusão de Curso. Departamento de Ciência da Computação do Instituto de Matemática e Estatística da Universidade de São Paulo. 2006. 39p.

SINGH, Munindar P.; HUHNS, Michael N. **Service-Oriented Computing.** John Wiley & Sons. 2005. 589p.

SPRING. **REST.** 2018. Disponível em: <https://spring.io/understanding/REST>. Acesso em: 28 de out. 2017.

SWAGGER. **Interface do Swagger.** 2018. Disponível em: <https://swagger.io/tools/swagger-ui/>. Acesso em 29 de junho de 2018.

THOMAZ, Lucas Roberto; CORREIA, Ronaldo Celso Messias; **Arquitetura Orientada a Serviços baseada na Tecnologia Jini: um caso de teste em Computação Ubíqua.** Universidade Estadual Paulista – Faculdade de Ciências e Tecnologia. 2010. 66p.

USSUY, Gabriel Yoshiharu Rodrigues; FARTO, Guilherme de Cleva. **Adoção de QR Code como proposta para representação visual e interação em grandes volumes de dados**. Trabalho de Conclusão de Curso – IMESA – Fundação Educacional do Município de Assis, SP, Assis, 2016. 58p.

VIEIRA, Marcos Rodrigues; FIGUEIREDO Josiel Maimone; LIBERATTI, Gustavo; VIEBRANTZ, Alvaro Fellipe Mendes. **Bancos de Dados NoSQL: Conceitos, Ferramentas, Linguagens e Estudos de Casos no Contexto de Big Data.** Minicurso apresentando no Simpósio de Bancos de Dados 2012. 2p.