UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
COORDENAÇÃO DO CURSO DE CIÊNCIA DA COMPUTAÇÃO
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

Márcio José Araújo

# QRmesse
# Automatização de eventos comunitários

TRABALHO DE CONCLUSÃO DE CURSO

SANTA HELENA

2018

Márcio José Araújo

# QRmesse

## Automatização de eventos comunitários

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Ciência da Computação da Universidade Tecnológica Federal do Paraná, Câmpus Santa Helena, como requisito parcial à obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof. Me. Giuvane Conti

Santa Helena, Paraná

**2018**

Ministério da Educação
**Universidade Tecnológica Federal do Paraná**
Câmpus Santa Helena
Coordenação do Curso de Ciência da Computação - COCIC

# TERMO DE APROVAÇÃO

## “QRmesse: Automatização de Eventos Comunitários”

por

## “Márcio José Araújo”

Este Trabalho de Conclusão de Curso foi apresentado às 15:50 do dia 26 de junho de 2018 na sala L8 como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação na Universidade Tecnológica Federal do Paraná - UTFPR - Câmpus Santa Helena. O aluno foi arguido pela Banca de Avaliação abaixo assinados. Após deliberação, a Banca de Avaliação considerou o trabalho aprovado.

Prof. Me. Giuvane Conti
(Presidente - UTFPR/Santa Helena)

Prof. Dra. Arlete Teresinha Beuren
(Avaliador 1 - UTFPR/Santa Helena)

Prof. Dr. Thiago França Naves
(Avaliador 2 - UTFPR/Santa Helena)

Prof. Dr. Franck Carlos Vélez Benito
(Professor Responsável pelo TCC - UTFPR/Santa Helena)

Prof. Dra. Arlete Teresinha Beuren
(Coordenador do curso de Bacharelado em Ciência da Computação – UTFPR/Santa Helena)

A folha de aprovação assinada encontra-se na Coordenação do Curso

*Dedico este trabalho a minha família,*
*minha amada esposa Vera,*
*maior incentivadora nesse percurso*
*e a meus queridos filhos Estela e Maurício José,*
*que sustentam minha motivação.*

# RESUMO

Este trabalho apresenta o desenvolvimento do sistema QRmesse, que substitui as planilhas usadas para a compra de produtos em eventos comunitários, como festivais de igrejas e festas escolares. A implementação do sistema exigiu um estudo, relatado aqui, sobre o processo de automação de venda e entrega de produtos, auxiliado por dispositivos móveis Android, como leitores de QR Code. O aplicativo projetado para dispositivos móveis se integra a um sistema Java EE via Web Service. Além disso, o QRmesse usa frameworks e bibliotecas que facilitam a construção dos aplicativos Java EE, Web Services e Android - este trabalho descreve sua implementação. O sistema foi testado em um evento, mostrando desempenho e usabilidade satisfatórios. A partir desse teste, surgiram problemas, que foram abordados e resolvidos. Conclui que a automação de eventos comunitários proporciona uma melhor experiência aos seus participantes e, portanto, contribui para o sucesso de tais eventos.

**Palavras-chave**: Java EE; Aplicativo Android; QR Code; Web Service.

# ABSTRACT

This work presents the development of the QRmesse system, which replaces the sheets used for purchasing products in community events, such as church festivals and school parties. The system implementation required a study, reported here, on the automating process of selling and delivering products, aided by Android mobile devices, as QR Code readers. The application designed for mobile devices integrates to a Java EE system via Web Service. In addition, the QRmesse uses frameworks and libraries that make easier to build the applications Java EE, Web Services, and Android - this work describes their deployment. The system was tested in one event, showing satisfactory performance and usability. From this test, issues came, that were approached and solved. It concludes that the automating of community events provides a better experience to their participants, and so contributes to the success of such events.



**Keywords**: Java EE; Android Application; QR Code; Web Service.

# LISTA DE FIGURAS

# LISTA DE QUADROS

# LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| 1D | Unidimensional |
| 2D | Bidimensional |
| 3D | Tridimensional |
| AIDC | Automatic Identification Data Capture |
| API | Application Programming Interface |
| B2B | Business to Business |
| B2C | Business to Commerce |
| CEP | Código de Endereçamento Postal |
| DER | Diagrama Entidade Relacionamento |
| HTTP | Hypertext Transfer Protocol |
| JAXB | Java Architecture for XML Binding |
| JAX-RS | Java API for RESTful Web Services |
| JAX-WS | Java API for XML Web Services |
| JPA | Java Persistence API |
| JSF | Java Server Faces |
| JSON | JavaScript Object Notation |
| NoSQL | Not Only SQL |
| QR | Quick Response |
| REST | REpresentational State Transfer |
| RPC | Remote Procedure Call |
| SGBD | Sistema Gerenciador de Banco de Dados |
| SOA | Service-Oriented Architecture |
| SOAP | Simple Object Access Protocol |
| SQL | Structured Query Language |
| UDP | User Datagram Protocol |
| WSDL | Web Services Description Language |
| XML | eXtensible Markup Language |

**SUMÁRIO**

# 1 INTRODUÇÃO

Eventos comunitários são definidos como festas organizadas por grupos pertencentes a qualquer instituição filantrópica, promovidas com a finalidade de angariar fundos e que demandam esforço coletivo e prazer em um mesmo acontecimento (OLIVEIRA, 2007).

Mesmo geralmente não sendo organizados nem executados por profissionais, os eventos comunitários objetivam lucro. Os ganhos financeiros têm como finalidade angariar fundos para determinado fim. Assim, o destino dos recursos arrecadados pode ser desde a construção ou reforma de uma edificação até o fortalecimento financeiro da instituição promotora.

A organização desses eventos, quando não é terceirizada, geralmente depende da participação de voluntários, portanto, essa estrutura não profissional geralmente implica simplicidade na logística aplicada ao evento. Assim, é comum que, em eventos de pequeno e médio porte, a venda de produtos que serão consumidos pelos participantes da festa seja realizada por meio de fichas de plástico ou papel, disponíveis em pontos de venda localizados no ambiente do evento. A entrega dos produtos está condicionada à entrega da ficha correspondente ao produto desejado.

## 1.1 PROBLEMA

O sistema de fichas pode ocasionar filas, tanto nos pontos de venda quanto nos pontos de entrega dos produtos. Nos pontos de venda, uma das dificuldades advém do cálculo para o troco, quando necessário, já que as fichas de produtos geralmente possuem preços variados. Assim, no momento da compra, o consumidor informa ao atendente quais e quantos produtos deseja adquirir. Em seguida, o atendente multiplica a quantidade de fichas por seu respectivo valor em moeda corrente, soma os valores e informa o resultado ao cliente que, enfim, entrega a quantidade de notas ou moedas, correspondente ao valor das fichas de produtos. Dessa forma, este trabalho aborda a venda de produtos em eventos comunitários sob a perspectiva comercial, ou seja, trata analogamente um evento comunitário como se fosse um conglomerado comercial temporário, em que o pagamento dos produtos deve sempre preceder a entrega.

Nos pontos de entrega o tempo de espera depende da disponibilidade do produto. Alguns produtos – como refrigerantes, doces e bolos – estão prontos para a entrega. Assim, o cliente entrega a ficha e recebe o produto. Nesse caso, a ordem de recebimento depende da ordem de

chegada do cliente ao ponto de entrega. Todavia, existem produtos, geralmente comestíveis, que são preparados durante o evento, como pastéis, pinhão e porções. Portanto, a entrega desse tipo de produto pode demorar alguns ou vários minutos, a depender da demanda, o que pode ocasionar filas, às vezes tumultuosas, pois em alguns momentos a demanda supera muito a oferta. Ocasionalmente, a fila desaparece e o que ocorre é uma aglomeração de pessoas em volta da barraca fornecedora do produto. Essa situação pode causar estresse nos participantes do evento: ajudantes-voluntários e clientes.

## 1.2 SOLUÇÃO PROPOSTA

A fim de atenuar essa situação, este trabalho apresenta o sistema computacional QRmesse, que auxilia na gestão de festas e eventos comunitários. Fazem parte do sistema uma aplicação Java Web com WebService RESTful e uma aplicação Android. O QRmesse organiza os processos de venda e entrega de produtos em eventos comunitários, bem como gera base de informações que podem ser utilizadas para otimizar eventos futuros da mesma organização. A principal particularidade do sistema está na substituição das fichas, que são trocadas por produtos, por um cartão dotado de QR Code, lido por *smartphones* Android, conectados a uma aplicação Java *web*. Ademais, o desenvolvimento da aplicação leva em conta restrições orçamentárias e o espírito voluntário que há entre os organizadores desses eventos.

## 1.3 JUSTIFICATIVA

A partir da evolução e do aumento da oferta de dispositivos eletrônicos no mercado, sistemas computacionais puderam ser desenvolvidos, a fim de satisfazer as exigências de controle administrativo e oportunizar melhorias no atendimento ao cliente (SENHORAS, 2003). Segundo Maximiano (2006), eficiência é a melhor relação obtida entre custos e benefícios. Considera-se que a eficiência na execução de um serviço se relacione de forma proporcionalmente inversa ao tempo de espera, pois, conforme Hui e Tse (2001), quanto mais longa uma pessoa acredita ter sido a espera, mais negativa será sua avaliação do serviço. Além disso, a expectativa de esperar acarreta custos econômicos e psicológicos (OSUNA, 1985). Ainda, Hui e Tse (2001) mostram que é mais fácil o consumidor aceitar emocionalmente o tempo de espera quando informado sobre qual é sua posição na fila ou quanto tempo aproximadamente precisa esperar para receber um produto.

## 1.4 OBJETIVO GERAL

Assim, o objetivo deste trabalho é desenvolver um sistema computacional que facilite o controle da venda e entrega de produtos em eventos comunitários utilizando dispositivos móveis como leitores de QR Code.

### 1.4.1 Objetivos específicos

- Desenvolver um banco de dados PostgreSql utilizando funções pl/pgsql;
- Desenvolver uma aplicação Java Web JSF;
- Implementar a camada de segurança utilizando o *framework* Spring Security;
- Desenvolver uma estrutura Web Service RESTful;
- Desenvolver uma aplicação cliente Android;
- Implementar a leitura de QR Code na aplicação Android.

A fim de organizar as informações, o trabalho está dividido em capítulos que pretendem facilitar a compreensão. O capítulo 2 apresenta as principais ferramentas utilizadas para a produção do sistema QRmesse. No capítulo 3, está a descrição da maneira como ocorreu a implementação dessas ferramentas no sistema. O capítulo 4 apresenta os resultados, por meio de imagens da interface do usuário, o relato de um evento de validação em que o QRmesse foi utilizado e algumas limitações do sistema identificadas. Finalmente, o capítulo 5 contém as considerações finais acerca do trabalho realizado e sugestões para trabalhos futuros.

# 2 REFERENCIAL TEÓRICO

Este capítulo apresenta literaturas que sustentam a base teórica do problema apresentado na Introdução e as técnicas e tecnologias utilizadas para o desenvolvimento do sistema QRmesse.

## 2.1 FESTAS E EVENTOS COMUNITÁRIOS

As festas fortalecem os laços entre o homem e o meio, as celebrações sempre refletiram o modo de pensar dos grupos sociais e como eles percebem e concebem seu ambiente, além de demonstrar o quão valorizam, mais ou menos, certos lugares (BEZERRA, 2012). Nessa mesma linha, Fernandez (2001, p. 7) diz que: “As festas são fenômenos primordiais e indissociáveis da civilização, porque nelas os homens sempre alcançam os mais altos níveis de sociabilidade”. Contudo, pode-se dizer que atualmente há, além dos valores e das tradições populares, interesses econômicos e comerciais na organização de festas comunitárias, como defende Bezerra (2012).

## 2.2 AUTOMAÇÃO PARA OTIMIZAR RESULTADOS

### 2.2.1 Identificação e Captura Automática de Dados

Segundo Romano (2011), uma das necessidades de atividades do varejo é a utilização de ferramentas com o fim de agilizar a captura de dados para evidenciar benefícios de custo, produtividade, qualidade, flexibilidade e inovação. Diversas tecnologias podem ser empregadas em dispositivos para identificação e captura automática de dados (AIDC: Automatic Identification Data Capture), e Romano (2011) resume as tecnologias AIDC em 6 grupos, presentes na Figura 1.

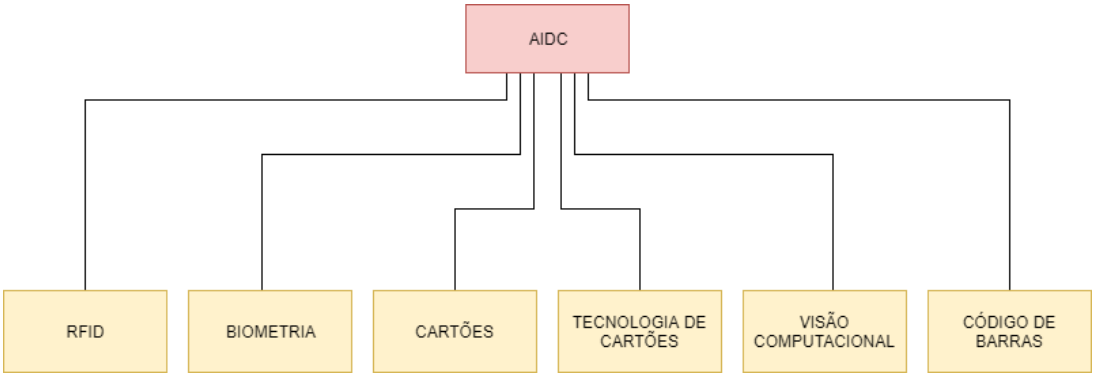


Figura 1: Grupos de tecnologias AIDC.

**Fonte**: Adaptado de Romano (2011).

O grupo Código de Barras possui dois tipos: o unidimensional e o bidimensional, apresentados na Figura 2. O padrão unidimensional é comumente encontrado em quase todos os produtos comercializados. No ano de 2002, já havia mais de um milhão de produtos com códigos de barras no Brasil (GS1 BRASIL, 2017).

Conforme Gao, Prakash e Jagatesan (2007), o padrão bidimensional foi inventado no final dos anos 80 para atender às necessidades de codificar dados alfanuméricos, incluindo letras, números e marcas de pontuação. No mesmo trabalho, os autores destacam que a utilização dos códigos de barra bidimensionais se popularizou em diferentes áreas e que, em geral, há dois tipos de códigos de barras 2D, os empilhados e de matriz. No grupo dos empilhados estão presentes padrões como o PDF417 e o Code 49, já no grupo de códigos em formato de matriz estão o QR Code e o Data Matrix. A Tabela 1 mostra as diferenças entre alguns tipos de códigos de barra 2D.

Tabela 1: Capacidades, funcionalidades e padrões para alguns tipos de códigos de barra bidimensionais.

| | **QR Code** | **PDF417** | **DataMatrix** | **Maxi Code** |
|---|---|---|---|---|
| Desenvolvedor (país) | Denso (Japão) | Symbol Technologies (EUA) | RVSI Acuity CiMatrix (EUA) | UPS (EUA) |
| Numérico | 7.089 | 2,710 | 3,116 | 138 |
| Alfanumérico | 4.296 | 1.850 | 2.355 | 778 |
| Binário | 2.953 | 1.018 | 1.556 | 93 |
| Kanji | 1.817 | 554 | 778 | |
| Principais funcionalidades | Ampla capacidade de armazenamento; Tamanho de impressão pequeno; Alta velocidade de leitura. | Ampla capacidade de armazenamento | Tamanho de impressão pequeno | Alta velocidade de leitura |
| Padrões | AIM Internacional<br>JIS<br>ISO | AIM Internacional<br>ISO | AIM Internacional<br>ISO | AIM Internacional<br>ISO |

**Fonte**: Traduzido de Gao, Prakash e Jagatesan (2007).

Atualmente, o padrão mais popular de códigos de barras 2D é o QR Code (Quick Response Code – código de resposta rápida). Esse padrão foi anunciado pela Denso Wave em 1994 (QR CODE.COM, 2017), com a finalidade de ampliar a capacidade de informações armazenadas por códigos de barras unidimensionais. Ele pode armazenar até 354 vezes mais informações e ser lido até 10 vezes mais rápido do que seu antecessor (PARRA, 2015), o código de barras unidimensional.

Ambos os padrões de códigos de barras podem ser facilmente impressos por qualquer impressora comum, pois a imagem gerada por eles é comumente preta e branca. Todavia, existe a necessidade de leitura, e, nesse aspecto, pode-se considerar a versão 2D mais eficiente, pois ela é mais segura e mais rápida. Chuang, Hu e Ko (2010) afirmam que, do ponto de vista de segurança, códigos de barra 1D podem ser facilmente reconhecidos observando as linhas verticais e as lacunas entre as barras. Ainda, apontam a necessidade de alinhamento do ângulo de leitura como outro fator negativo. O padrão 2D, mais especificamente o QR Code, possui amplo intervalo de ângulos para leitura, devido à marcação de 3 cantos, destacados na Figura 2b, que determina qual é a posição dos dados na matriz. Seu padrão simbólico não é fácil de ser interpretado por olhos humanos. Já dispositivos móveis dotados de câmera podem fazer a leitura de forma eficiente, bastando apenas a instalação de aplicativos.

a: Unidimensional 1D b: Bidimensional 2D

Figura 2: Código de barras

Hoje, com o avanço do processamento de imagens e das capacidades multimídia dos dispositivos móveis, eles podem ser utilizados como codificadores e decodificadores de códigos de barra (GAO, PRAKASH e JAGATESAN, 2007).

## 2.3 APLICAÇÕES ANDROID

Conforme Rubin (2007), o Android é a primeira plataforma verdadeiramente aberta e abrangente para dispositivos móveis. Ele inclui um sistema operacional, interface de usuário e aplicativos, enfim, com ele todo o *software* pode ser executado em um telefone celular, mas sem os obstáculos proprietários que impediam a inovação móvel. Deitel (2016, p.4) afirma que a franqueza – ou grau de abertura – é uma vantagem de desenvolver aplicativos Android, pois é um sistema operacional aberto e gratuito e por isso diversos aplicativos Android, também de código-fonte aberto e gratuito, produzidos pelo Google e outros desenvolvedores, estão disponíveis na Internet. Ainda na mesma obra, Deitel afirma que:

> O grau de abertura da plataforma estimula a rápida inovação. Ao contrário do iOS patenteado da Apple, que só existe em dispositivos Apple, o Android está disponível em aparelhos de dezenas de fabricantes de equipamento original (OEMs) e em numerosas operadoras de telecomunicações em todo o mundo. A intensa concorrência entre os OEMs e as operadoras beneficia os consumidores.

### 2.3.1 Desenvolvimento de Aplicações Android

O desenvolvimento de aplicações para o sistema operacional Android pode ser realizado em diversas linguagens de programação, dentre elas estão Java, Kotlin, C#, C++ e Python. Todavia, como descrito em Android Developers (2018a), a ferramenta oficial para desenvolvimento de aplicativos Android é o IDE Android Studio, no qual Java é a linguagem

padrão. A linguagem Kotlin, desenvolvida pela JetBrains (mesma empresa responsável pelo IDE IntelliJ Idea, em que o próprio Android Studio é baseado), cujo projeto teve início em 2011 (KRILL, 2011), também é uma linguagem oficial para desenvolvimento Android.

O Gradle é o sistema de compilação de base no Android Studio. Outros recursos específicos do Android podem ser acessados pelo Android Plugin para Gradle. O Gradle pode ser executado como uma ferramenta integrada no menu do Android Studio ou via linha de comando. Além disso, ele realiza o gerenciamento de dependências (ANDROID DEVELOPERS, 2018a).

### 2.3.2 Aplicações Android para Controle de Pedidos

Atualmente, o acesso a *smartphones* está na taxa de um aparelho por habitante no Brasil (MEIRELLES, 2017). Trabalhos como os de Fernandes, Marcilio e Marques (2015), Monarim (2013), Barros, Lemos e Amaral (2015) e Almeida Holanda e Sena do Nascimento (2012) mostram que dispositivos móveis podem ser utilizados como uma ferramenta na automação de processos.

Os primeiros dispositivos móveis utilizados como ferramentas para a realização de pedidos em restaurantes foram os Personal Digital Assistants (PDAs), também conhecidos como Palmtops. Os trabalhos de Patel, Patel e Obersnel (2007) e Cheong, Chiew e Yap (2010) apresentam utilizações de PDAs como ferramentas para a solicitação de pedidos. Contudo, o surgimento dos *smartphones* suscitou a evolução de aplicações que utilizam dispositivos móveis para esse fim, conforme o trabalho de Tanpure, Shidankar e Joshi (2013). Estudos como os de Dhore et al. (2014) e Sarkar et al. (2014) mostram que aplicações Android, com a finalidade de gerenciar pedidos, aumentam a velocidade de atendimento, a qualidade do serviço e a popularidade de estabelecimentos, além de reduzir erros humanos.

### 2.3.3 A API ZXing

Uma alternativa para o desenvolvimento de aplicações Android que implementam a leitura de códigos de barra é a API ZXing (*zebra crossing*), uma biblioteca de processamento de imagem de código de barras de 1D e 2D, de código aberto, implementada em Java, com abertura para outras linguagens (ZXING PROJECT, 2017). O Quadro 1 apresenta os modelos

suportados por essa biblioteca. Trabalhos como os de Félix (2016), Vanz (2012) e Fernandes, Marcilio e Marques (2015) utilizam e descrevem, inclusive com códigos-fonte, como funciona a integração da API ZXing com aplicações Java e Android.

Quadro 1: Formatos de códigos de barra suportados pela API ZXing.

| **Produto 1D** | **Industrial 1D** | **2D** |
|---|---|---|
| UPC-A | Code 39 | QR Code |
| UPC-E | Code 93 | Data Matrix |
| EAN-8 | Code 128 | Aztec (beta) |
| EAN-13 | Codabar | PDF 417 (beta) |
| | ITF | MaxiCode |
| | RSS-14 | |
| | RSS-Expanded | |

**Fonte**: Adaptado de ZXing Project (2017).

## 2.4 BANCOS DE DADOS POSTGRESQL

Existem diversos sistemas gerenciadores de bancos de dados (SGBDs). Dentre os mais utilizados, conforme o *ranking* DB-Engines presente na Figura 3, está o PostgreSQL. Ele pode ser acessado a partir das principais linguagens de programação, incluindo C, C++, Perl, Python, Java, Tcl e PHP (STONES e MATTHEW, 2006). Atualmente, segue o padrão da indústria para linguagens e consulta o SQL:2003. Por ser uma aplicação de código-fonte aberto, uma equipe de desenvolvedores da Internet mantém o PostgreSQL. Assim, os usuários têm acesso ao código-fonte e contribuem com correções, aprimoramentos e sugestões para novos recursos. Os lançamentos oficiais do PostgreSQL são feitos via <http://www.postgresql.org> (STONES e MATTHEW, 2006).

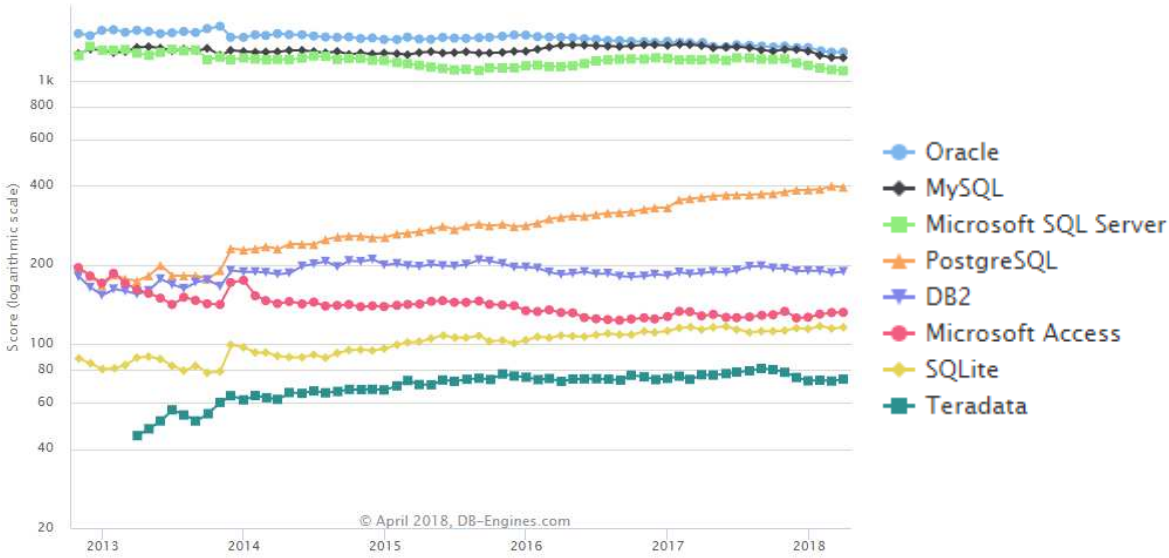


Figura 3: Ranking *top* 10 da popularidade mundial de SGBDs.
**Fonte**: Extraído de DB-Engines (2018).

### 2.4.1 Linguagens Procedurais no PostgreSQL

O PostgreSQL permite que o desenvolvedor estenda a funcionalidade do servidor de banco de dados, por meio da criação de funcionalidades, usando a linguagem de programação C, e conecte-as ao servidor no momento da inicialização do banco de dados. Uma extensão pode ser tão simples quanto uma única função extra ou tão complexa quanto uma linguagem de programação completa. Várias dessas extensões, conhecidas como linguagens procedurais carregáveis, são incluídas na distribuição padrão do PostgreSQL. Essas linguagens permitem a criação de funções personalizadas, conhecidas como *stored procedures*, de forma rápida e mais fácil do que escrever em C (MATTHEW e STONES, 2005). Atualmente, na versão 10 do PostgreSQL, existem quatro linguagens procedurais disponíveis na distribuição padrão: PL/pgSQL, PL/Tcl, PL/Perl e PL/Python (POSTGRESQL, 2018).

### 2.4.2 PL/pgSQL

Por ser SQL uma linguagem de consulta declarativa, ela permite apenas a substituição de argumentos. Já a PL/pgSQL ou PLpgSQL inclui recursos como declaração e atribuição de variáveis, estruturas de seleção, estruturas de condição e *loopings* (MOMJIAN, 2001). Os objetivos de desenvolvimento da PL/pgSQL ou PLpgSQL eram justamente criar uma

linguagem procedural carregável, que pudesse ser usada para criar funções e desencadear procedimentos, adicionar estruturas de controle à linguagem SQL, executar cálculos complexos, herdar todos os tipos, funções e operadores definidos pelo usuário. SQL é a linguagem que o PostgreSQL e vários outros bancos de dados relacionais usam como linguagem de consulta. É portátil e fácil de aprender, mas toda instrução SQL deve ser executada individualmente pelo servidor de banco de dados (POSTGRESQL, 2018).

As funções PLpgSQL podem ser declaradas para aceitar argumentos passados por parâmetros. Também podem ser declaradas para aceitar e retornar os tipos polimórficos, conjuntos ou tabelas de qualquer tipo de dado que possa ser retornado como uma única instância. Além disso, uma função PL pgSQL pode ser declarada para retornar *void* se não tiver valor de retorno, ou seja, se apenas executar alguma ação na base de dados. Algumas das vantagens do uso de PLpgSQL em um banco de dados são: Eliminação de chamadas extras entre cliente e servidor; Resultados intermediários desenecessários à aplicação cliente não precisam ser delegados ou transferidos entre o cliente e servidor; Iterações de análise ou consulta podem ser evitadas.

## 2.5 PLATAFORMA JAVA EE

De acordo com Layka (2014), uma aplicação *web* é um conjunto de páginas da *web* capaz de gerar conteúdo dinâmico em resposta a solicitações, e parte da popularidade da linguagem Java pode ser atribuída a seu uso na criação aplicações *web*. O Java 2 Platform Enterprise Edition (J2EE) apareceu no final dos anos 90 e trouxe para a linguagem Java uma robusta plataforma de *software* para desenvolvimento corporativo. O J2EE, predecessor do Java Platform Enterprise Edition (Java EE), foi desafiado a cada nova versão, mal-entendido ou mal utilizado, era rebuscado e competia com *frameworks* de código aberto. Era visto como uma tecnologia muito pesada. Então, o Java EE se beneficiou dessas críticas para melhorar, e hoje se concentra na simplicidade (GONÇALVES, 2013). O objetivo da plataforma Java EE é fornecer aos desenvolvedores um conjunto eficiente de APIs que podem reduzir o tempo de desenvolvimento, reduzir a complexidade da aplicação e melhorar o desempenho das aplicações (ORACLE, 2017). Ao contrário de páginas da *web* estáticas, uma aplicação *web* permite a execução de atividades e o armazenamento das informações resultantes.

Uma especificação Java desenvolvida para facilitar o desenvolvimento de aplicações *web* é a Java Server Faces (JSF). Concebida para facilitar a criação de interfaces gráficas e inspirada no modelo de componente Swing e em outras estruturas GUI (interface gráfica de usuário), a JSF permite que os desenvolvedores pensem em termos de componentes, eventos, *beans* auxiliares e suas interações, em vez de requisições, respostas e linguagem de marcação. O objetivo é tornar o desenvolvimento da Web mais rápido e fácil, oferecendo suporte a componentes da interface do usuário (como caixas de texto, caixas de listagem, painéis com guias e grades de dados) em uma abordagem de desenvolvimento rápido de aplicativos (RAD) (GONÇALVES, 2013).

### 2.5.1 EclipseLink

Como documentado em The Eclipse Foundation (2014), o EclipseLink é uma estrutura de persistência Java eficiente e flexível para armazenar objetos Java em um banco de dados, relacional ou NoSQL, e para converter objetos Java em documentos XML ou JSON. O EclipseLink fornece APIs e ambiente em tempo de execução para implementar a camada de persistência da Plataforma Java. Além disso, implementa Java Persistence API (JPA) arquitetura Java para XML Binding (JAXB) e outras tecnologias de persistência baseadas em padrões. O documento também lista algumas vantagens quando um projeto utiliza mapeamento objeto-relacional por meio do EclipseLink:

a) EclipseLink é o provedor de persistência padrão para domínios do WebLogic Server, com suporte para JPA 2.1.
b) A implementação do EclipseLink Java para XML Binding (JAXB) é a implementação padrão do JAXB no WebLogic Server. O EclipseLink implementa completamente o JAXB e também inclui outros recursos avançados. Por padrão, pode ser aproveitado o EclipseLink JAXB na API Web Services XML (JAX-WS) e API Java para aplicativos RESTful Web Services (JAX-RS).
c) A integração do registro do EclipseLink no WebLogic Server fornece infraestrutura de *log* abrangente e integrada.

## 2.6 WEB SERVICES

Web Services são tecnologias para a construção de aplicativos distribuídos. Esses serviços, que podem ser disponibilizados via Internet, usam um sistema de mensagens HTML, XML, somente texto, PDF, JPEG, JSON, dentre outros não vinculados a sistemas operacionais específicos ou linguagens de programação (ORACLE, 2017). Assim, se a disponibilização de algum serviço pode ser automatizada, ou seja, pode responder a requisições de forma padronizada – como a localização de um endereço a partir de um CEP –, existe a possibilidade de implantação de um Web Service.

De acordo com Oracle (2010), a arquitetura Web Services pode ser visualizada em termos de papéis e de pilhas de protocolos, conforme os Quadros 2 e 3.

Quadro 2: Papéis Web Service.

| **Papel** | **Descrição** |
|---|---|
| *Service provider* | Fornece o serviço implementando-o e tornando-o disponível na Internet. |
| *Service requester* | Este é o usuário do serviço que o acessa abrindo uma conexão de rede e envia uma solicitação XML. |
| *Service registry* | Este é um diretório centralizado de serviços em que os desenvolvedores podem publicar novos serviços ou encontrar os existentes. |

**Fonte:** Traduzido de Oracle (2010).

Quadro 3: Pilhas de protocolos Web Services.

| **Pilha** | **Descrição** |
|---|---|
| *Service transport layer* | Camada de transporte de serviço, usa o protocolo HTTP para transportar mensagens entre aplicações. |
| XML *messaging layer* | Camada que codifica mensagens em formato XML, usando SOAP para trocar informações entre computadores. Define a especificação de envelope para dados encapsulados que são transferidos, as regras de codificação de dados e chamada de procedimento remoto (RPC). |
| *Service description layer* | Camada que descreve a interface pública para um Web Service específico, usando o protocolo WSDL (Web Service Description Language). |
| *Service discovery layer* | Essa camada centraliza serviços em um registro comum, utiliza o Protocolo de Descrição Universal (UDP). |

**Fonte:** Traduzido de Oracle (2010).

### 2.6.1 SOA – Arquitetura Orientada a Serviços

SOA é uma arquitetura orientada a serviços. Hurwitz et al. (2007) definem SOA como uma arquitetura orientada a serviços para a construção de aplicações como um conjunto de componentes caixa-preta, livremente acoplados e orquestrados para oferecer um nível de serviço bem definido, ligando os processos de negócios. Gartner (2017) diz que SOA é um paradigma de *design* e disciplina, que ajuda a TI a atender às demandas do negócio, com benefícios como tempo de mercado mais rápido, custos mais baixos, melhor consistência da aplicação e agilidade. Isso produz sistemas modulares interoperáveis, que são mais fáceis de utilizar e manter. A SOA cria sistemas mais simples e rápidos, que aumentam a agilidade e reduzem o custo total de propriedade. Kumar, Narayan e Ng (2009), de forma sucinta, definem SOA como uma maneira de arquitetar o aplicativo corporativo como um conjunto de serviços que todos os usuários da empresa desejam.

A chegada dos serviços *web* adicionou um marco significativo à história da TI. A Internet e a World Wide Web ajudaram na promoção da categoria B2C (Business-to-Consumer) das organizações. O desenvolvimento de novos padrões XML e de vocabulários avançados ajudou a promover a categoria B2B das organizações. As transações na *web* tornaram-se norma, e tanto B2C quanto B2B (Business-to-Business) ganharam significante impulso desde a chegada do HTTP. A SOA parece ser o novo mantra para as empresas enfrentarem os desafios comerciais iminentes e sobreviverem a eles, portanto, implementar SOA usando serviços da *web* parece ser o caminho mais adequado (KUMAR, 2010).

Erl (2005) aponta que, quando é construída uma solução de automação por meio de serviços, cada serviço pode encapsular uma tarefa realizada por um passo individual ou um subprocesso composto por um conjunto de etapas e, se necessário, um serviço pode encapsular toda a lógica do processo (Figura 4).

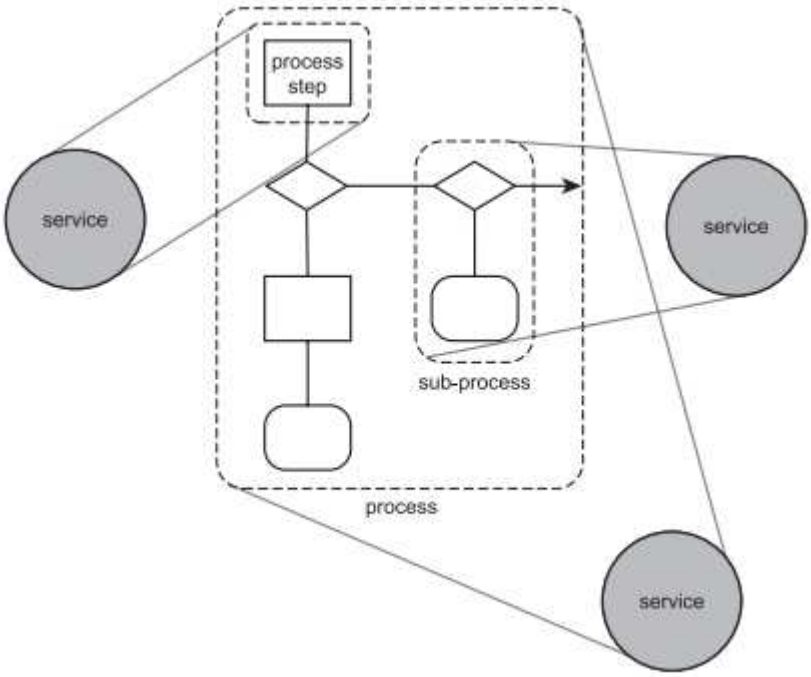


Figura 4: Serviços encapsulando quantidades variáveis de processos lógicos.

**Fonte**: Erl (2005).

### 2.6.2 O Padrão Arquitetural REST

Conforme Conti (2015), o padrão arquitetural REST (REpresentational State Transfer) cria um sistema hipermídia distribuído, utilizando elementos da Web. O termo REST foi criado por Fielding (2000), em sua tese de doutorado, que também propõe que esse padrão seja norteado por seis princípios básicos, resumidos no

Quadro 4.

Quadro 4: Restrições da arquitetura REST.

| Princípio | Descrição |
|---|---|
| Cliente-servidor | O cliente mantém o estado da sessão e cada requisição deve conter apenas informações necessárias para atender à solicitação. |
| Sem estado | A comunicação entre cliente e servidor não deve ter estado. O servidor não precisa guardar o estado do cliente. Em vez disso, os clientes devem incluir todas as informações necessárias na solicitação, para que o servidor possa atender a elas. |
| Sistema em camadas | Várias camadas hierárquicas como *gateways*, *firewalls* e *proxies* podem existir entre o cliente e o servidor. As camadas podem ser adicionadas, modificadas, reordenadas ou removidas de forma transparente, para melhorar a escalabilidade. |
| *Cache* | Os recursos podem ser armazenados em *cache* sempre que possível. O cabeçalho da resposta indica se os dados são armazenados em *cache* ou não, cabeçalhos de cache são suportados para melhorar o desempenho. |
| Interface uniforme | Todas as interações entre clientes, servidores e componentes intermediários são baseadas na uniformidade de suas interfaces. A restrição de interface uniforme é dividida em quatro sub-restrições: identificação de recursos, representações de recursos, mensagens autodescritivas e hipermídia como motor do estado de aplicação. |
| Código sob demanda | Essa é uma restrição opcional em que os clientes podem ampliar suas funcionalidades, baixando e executando o código sob demanda. Os exemplos incluem *scripts* de JavaScript, *applets* de Java, Silverlight e assim por diante. |

As aplicações que aderem a essas restrições são consideradas RESTful, logo, REST é um padrão arquitetural, e as aplicações que utilizam tal padrão recebem o nome RESTful Web Service. Essas restrições não ditarão a tecnologia real a ser usada para o desenvolvimento de aplicativos. Em vez disso, a adesão a essas diretrizes e práticas recomendadas torna um aplicativo escalável, visível, portátil, confiável e capaz de funcionar melhor. Em teoria, é possível que uma aplicação RESTful seja construída usando qualquer infraestrutura de rede ou protocolo de transporte. Na prática, os aplicativos RESTful utilizam recursos da Web e usam HTTP como protocolo de transporte (VARANASI e BELIDA, 2015).

# 3 METODOLOGIA

Este capítulo apresenta inicialmente a arquitetura do sistema, como as aplicações pertencentes a ele se interligam. Em seguida, descreve como se deu a implantação e implementação das ferramentas, técnicas e tecnologias descritas no capítulo anterior, bem como as justificativas de algumas escolhas.

## 3.1 ARQUITETURA DO SISTEMA

Para a implantação do sistema foram construídas duas aplicações, uma aplicação *web* com Web Service e um aplicativo Android, contudo, o primeiro passo a ser executado foi a modelagem e construção da base de dados PostgreSQL. Assim, a aplicação *web* Java EE foi desenvolvida para permitir o armazenamento, na base de dados, das informações necessárias à realização de um evento comunitário gerenciado pelo QRmesse e o acesso a elas. Essa aplicação também realiza o monitoramento do consumo de produtos durante o evento. Em seguida, foi desenvolvido o código, também na linguagem de programação Java, para a implantação do Web Service, e assim possibilitar o acesso às informações cadastradas e, quando necessário, adicionar informações por meio dos dispositivos móveis. Finalmente, iniciou-se o desenvolvimento da aplicação Android que permite a leitura do QR Code dos cartões e acessa o Web Service para solicitar ou adicionar informações ao sistema. A Figura 5 apresenta a arquitetura do sistema QRmesse, que é composto pela interligação dessas aplicações.

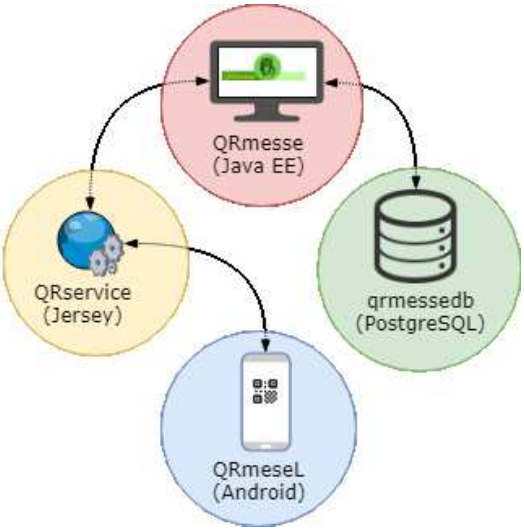


Figura 5: Interligação entre as aplicações do sistema QRmesse.

## 3.2 BASE DE DADOS

A base de dados foi construída no ambiente do *software* para gerenciamento de banco de dados pgAdmin III, para bases PostgreSQL. A utilização do PostgreSQL justifica-se pela possibilidade de inclusão da linguagem procedural carregável PL/pgSQL para o PostgreSQL. Esse recurso evita excessivas requisições cliente-servidor, elimina a necessidade de empacotamento ou transferência de resultados intermediários e evita múltiplas rodadas de consulta pelo servidor *web* (POSTGRESQL, 2017). Dessa forma, o retorno de operações que requerem a busca em mais de uma tabela e, além disso, necessitam percorrer registros para buscar informações será realizado diretamente no SGBD. Além das 16 tabelas, presentes no Diagrama Entidade Relacionamento (DER) do QRmesse (Apêndice I), a base de dados contém 3 *triggers* e 4 funções, presentes no Quadro 5.

O controle do saldo do cartão é realizado por meio de duas *triggers*. Por exemplo, quando for solicitado o saldo do cartão, deverão ser acessados todos os registros das tabelas tb_consumo e tb_carga referentes ao cartão desejado, em seguida, devem-se percorrer esses registros, adicionando os valores de carga e subtraindo os valores de consumo, para finalmente chegar ao valor do saldo. Para realizar esse processo utilizando apenas recursos da aplicação Java, seriam necessárias duas requisições mais o cálculo do saldo (Figura 6).

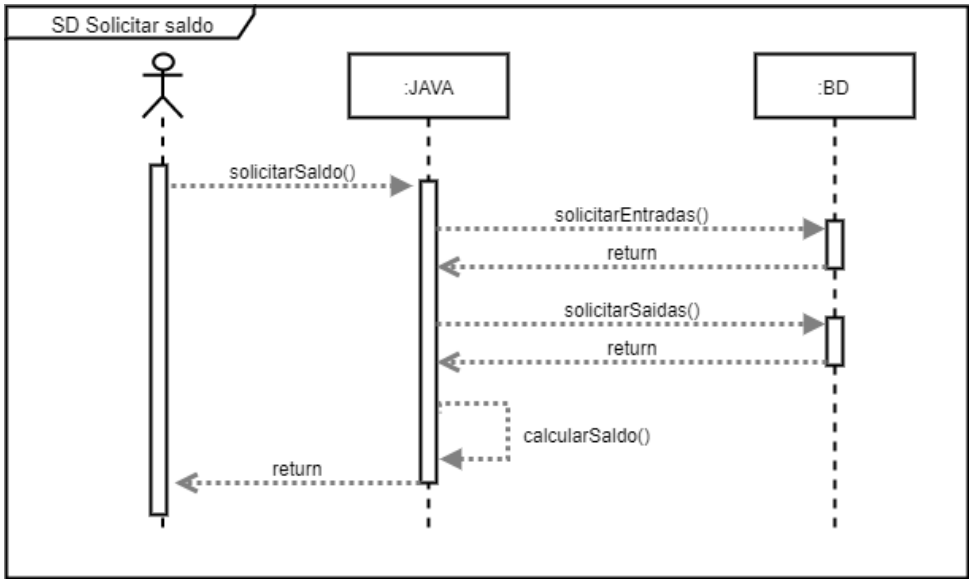


Figura 6: Diagrama de sequência para a solicitação de saldo sem PL/pgSQL.

Para simplificar o processo de cálculo do saldo do cartão foram desenvolvidas duas funções PLpgSQL. A *trigger* t_atualiza_saldo_cartao_consumo é disparada toda vez que um registro é adicionado na tabela tb_consumo, para que o valor do pedido seja subtraído do saldo do cartão. Pode-se notar, na Figura 9, que a tb_consumo não está diretamente relacionada com a tb_cartao, pois entre elas está a tb_pedido. Todavia, o preço do produto consumido, cujo valor será descontado do saldo do cartão, está presente na tb_consumo. Já a *trigger* t_atualiza_saldo_cartao_carga adiciona ao saldo atual do cartão o valor da carga e é disparada após a inserção de cada registro na tabela tb_carga. A Figura 7 contém o código de criação da *trigger* que dispara a função PLpgSQL (Figura 8) após o consumo de produtos.

```
5   CREATE TRIGGER t_atualiza_saldo_cartao_consumo
6       AFTER INSERT
7       ON public.tb_consumo
8       FOR EACH ROW
9       EXECUTE PROCEDURE public.func_atualiza_saldo_pos_consumo_prod();
```

Figura 7: *Trigger* disparada após o registro de consumo.

```
1   CREATE FUNCTION public.func_atualiza_saldo_pos_consumo_prod()
2       RETURNS trigger
3       LANGUAGE 'plpgsql'
4   AS $$
5   declare
6       valDesc float;
7       idCartao int;
8   begin
9       IF(UPPER(TG_OP) = 'INSERT') THEN
10          select preco into valDesc from tb_produto_valor where id = NEW.preco;
11          select cartao into idCartao from tb_pedido where id = New.pedido;
12          UPDATE tb_cartao set SALDO = (SALDO - (valDesc * NEW.quantidade))
13              WHERE ID = idCartao;
14          return new;
15          END IF;
16      return new;
17  end;
18  $$;
```

Figura 8: Função que atualiza o saldo do cartão após o consumo.

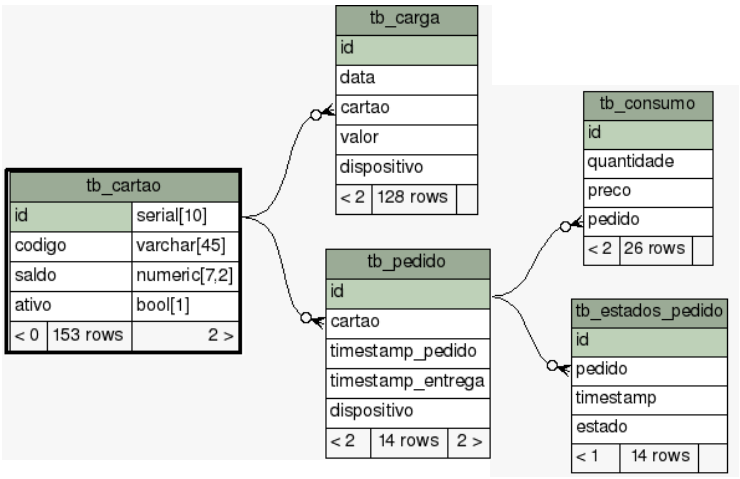


Figura 9: Relacionamentos da tabela tb_cartao com dois graus de separação.

A fim de simplificar o código da aplicação Java EE, foram desenvolvidas 4 funções PL/pgSQL. A utilização dessas funções também reduz a necessidade de requisições para o servidor do banco de dados, pois a partir de apenas uma chamada de função PLpgSQL são retornadas informações de várias tabelas. O Quadro 5 contém a descrição das funções implantadas no banco de dados.

Quadro 5: Funções PL/pgSQL desenvolvidas no banco de dados.

| Função | Descrição |
|---|---|
| func_pedidos_cartao | Retorna um registro com informações sobre o pedido do cartão, conforme id do cartão passado por parâmetro. |
| func_produtos_pedido | Retorna um registro com informações sobre os produtos do pedido, conforme id do pedido passado por parâmetro. |
| func_vendas_produtos_por_evento | Retorna uma lista de registros, cada um com informações sobre cada produto consumido no evento, conforme id do evento passado por parâmetro. |
| func_receita_evento | Retorna o valor arrecadado do evento cujo id foi passado por parâmetro. |

Outra funcionalidade implantada no sistema que aproveita os recursos do SGBD atende a eventuais funcionalidades do sistema que requisitem o valor total arrecadado com a venda de produtos durante o evento. Para tanto, basta realizar a chamada da função func_receita_evento, passando o número do id do evento como parâmetro. Pode-se notar no código-fonte, presente

na Figura 10, que a obtenção desse valor requer o acesso a informações de 5 tabelas. Dessa forma, qualquer aplicação que tenha acesso à base de dados poderá obter esse valor por meio de uma única chamada de função. A Figura 11 mostra como foi implantada essa chamada na aplicação Java Web do QRmesse.

```
1  CREATE OR REPLACE FUNCTION public.func_receita_evento(idevento integer)
2      RETURNS numeric
3      LANGUAGE 'plpgsql'
4      VOLATILE
5  AS $$
6  Declare
7      receita numeric;
8  BEGIN
9          select sum(pv.preco * c.quantidade) into receita from tb_consumo c
10         join tb_produto_valor pv on pv.id = c.preco
11         join tb_pedido pd on pd.id = c.pedido
12         join tb_produto_barraca pb on pb.produto = pv.produto
13         join tb_barraca b on b.id = pb.barraca
14         where b.evento = idevento;
15     return receita;
16 END;
17 $$;
```

Figura 10: Código da função PLpgSQL func_receita_evento.

```
41     public BigDecimal getReceita(Integer idEvento) {
42         StoredProcedureQuery query = ConexaoHibernate
43                 .getConnection()
44                 .createStoredProcedureQuery("func_receita_evento");
45         query.registerStoredProcedureParameter(1, Integer.class, ParameterMode.IN)
46                 .setParameter(1, idEvento);
47         query.execute();
48         return (BigDecimal) query.getSingleResult();
49     }
```

Figura 11: Chamada de função PLpgSQL na aplicação Java *web*.

### 3.2.1 Eventos

A tabela tb_evento é responsável por armazenar as informações sobre o evento. Além de identificar o evento pelo seu nome, os campos dt_inicio e dt_fim são obrigatórios para que o sistema responda apenas às solicitações realizadas durante esse período. Essa tabela está diretamente relacionada com as tabelas tb_barraca e tb_caixa, que armazenam informações sobre os pontos de venda de produtos e carga de cartões, respectivamente. Além disso, há também um relacionamento direto com a tabela instituição, que armazena informações sobre a instituição promotora do evento, conforme a Figura 12.

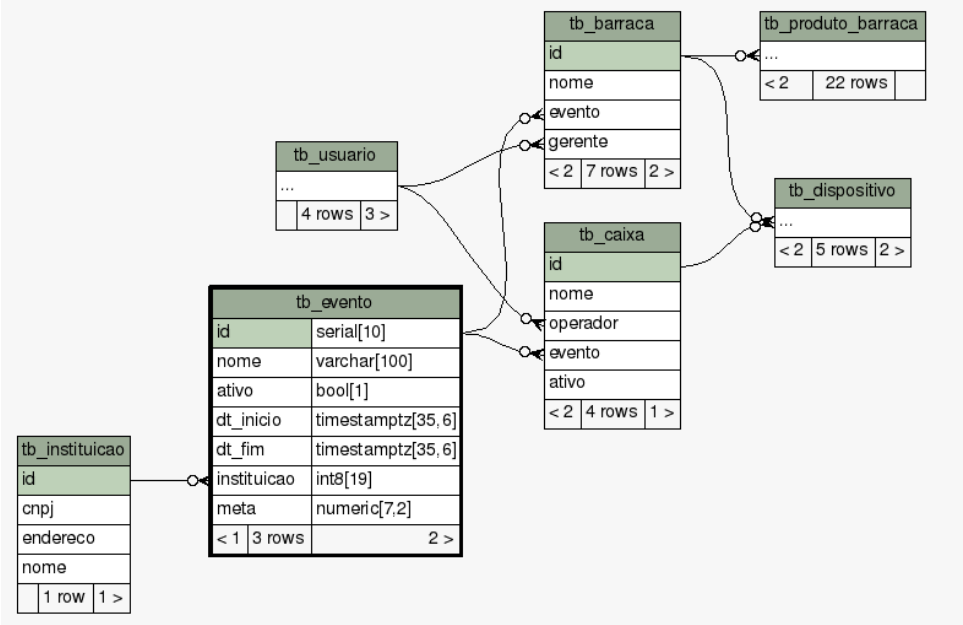


Figura 12: Relacionamentos da tabela tb_evento.

### 3.2.2 Pedidos

O registro dos pedidos é realizado na tabela tb_pedidos. A Figura 13 mostra todas as tabelas diretamente relacionadas a ela.

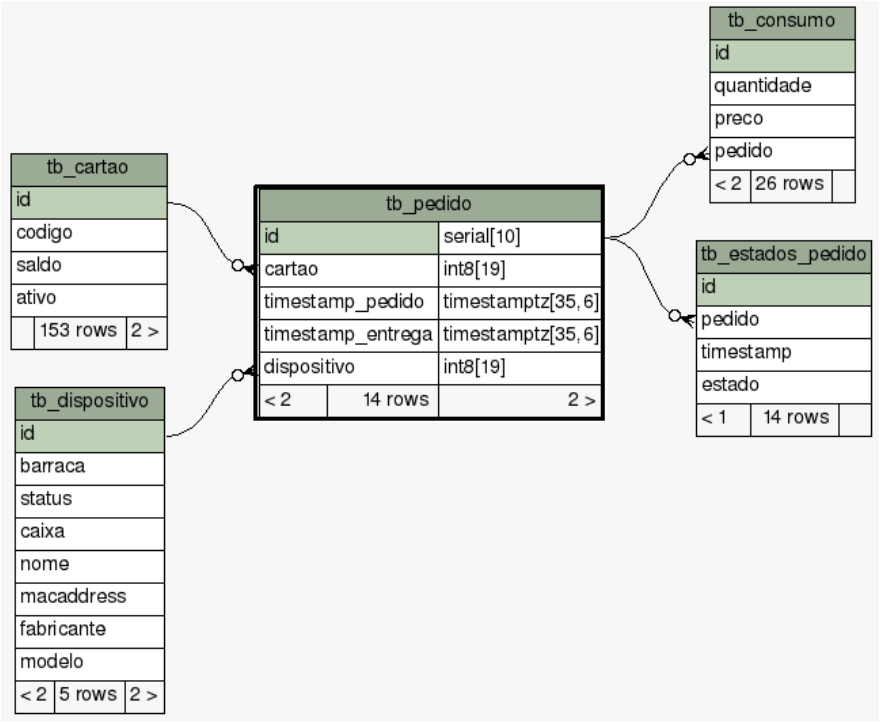


Figura 13: Relacionamentos diretos da tabela tb_pedido.

Portanto, pedidos são efetuados para um cartão, a partir de um dispositivo. Todo pedido gera ao menos um consumo e cada mudança de estado do pedido deve ser registrada. Somente dispositivos em que a barraca caixa não está nula conseguem realizar pedidos, outrossim, o *macadress* do dispositivo deve ser cadastrado (Figura 14), para que somente dispositivos cadastrados possam realizar pedidos.

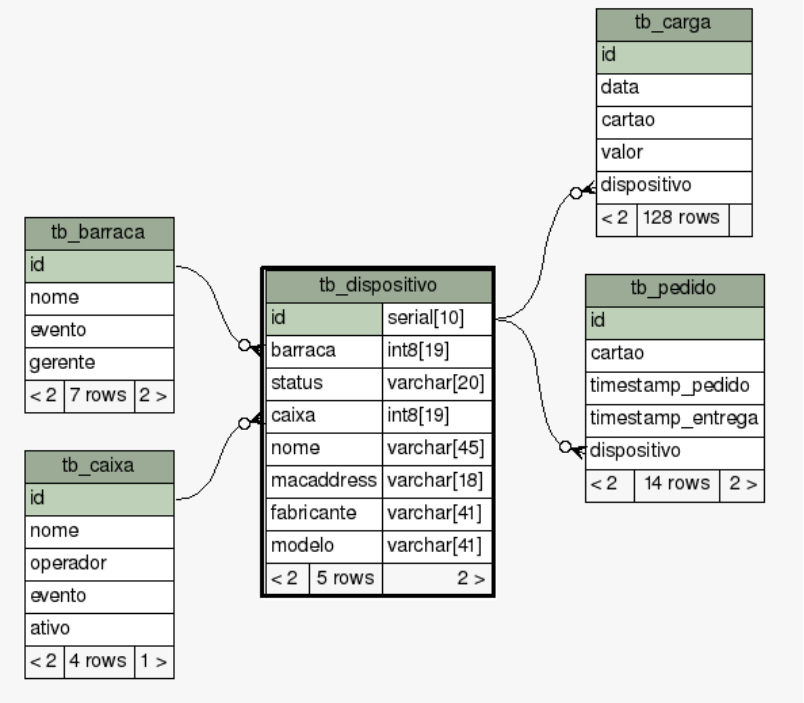


Figura 14: Relacionamentos diretos da tabela tb_dispositivo.

## 3.3 APLICAÇÃO JAVA WEB

A aplicação Java Web QRmesse tem como objetivo principal permitir o cadastro das instâncias das entidades necessárias à realização do evento. Dentre essas entidades estão: a instituição, os usuários, os cartões, os dispositivos, os produtos e os pontos de venda e de entrega. Além disso, o evento pode ser monitorado pela aplicação, ou seja, podem-se visualizar em tempo real a quantidade de cartões ativos, a quantidade de produtos consumidos e as filas de pedidos em preparo.

### 3.3.1 Geração do QR Code

A geração da imagem do QR Code deu-se pela utilização do *framework* PrimeFaces, presente na aplicação Java EE. A imagem do QR Code é lida pelos dispositivos móveis e o código correspondente é enviado ao Web Service por meio da conexão *wireless*. Esse código permite que o sistema encontre, na base de dados, o cartão, a fim de realizar (re)carga ou consumir saldo. O código é gerado a partir de uma função de *hash* criptográfico, dessa forma, pretende-se prevenir o sistema contra fraudes, pois, se, por exemplo, o código fosse apenas uma sequência numérica, bastaria a qualquer participante do evento, cujo QR Code do cartão gerasse um valor númerico N, imprimir um QR Code N+1 e tentar utilizá-lo no evento. A Figura 15 traz o trecho de código que mostra um código QR Code na tela que exibe os cartões.

```
28                  <p:column>
29                      <pe:qrCode id="qrCodeElem"
30                                 renderMethod="img"
31                                 renderMode="0"
32                                 text="#{cartaoBean.selecionado.codigo}"
33                                 label="QRmesse"
34                                 size="111"
35                                 fillColor="000000"
36                                 fontName="Ubuntu"
37                                 fontColor="#01A9DB"
38                                 ecLevel="H"
39                                 radius="0.5"/>
40                  </p:column>
```

Figura 15: Trecho de código que gera QR Code usando PrimeFaces.

## 3.4 IMPLANTAÇÃO DO WEB SERVICE

A necessidade de interligar a aplicação Java Web e a aplicação Android motivou a utilização de um *framework* Web Service. Para tanto, essa arquitetura (Figura 16) foi implantada sob o padrão RESTFul, em meio ao protocolo HTTP. Dessa forma, implementou-se a estrutura cliente-servidor, em que a aplicação cliente está nos dispositivos móveis presentes nas barracas, nos pontos de venda, pontos de consulta ou eventualmente em computador operado por usuários mantenedores do sistema. Já a aplicação Web Service está presente na máquina servidora do sistema. Dessa forma, disponibilizam-se os serviços da aplicação para qualquer outra aplicação que também possua acesso HTTP.

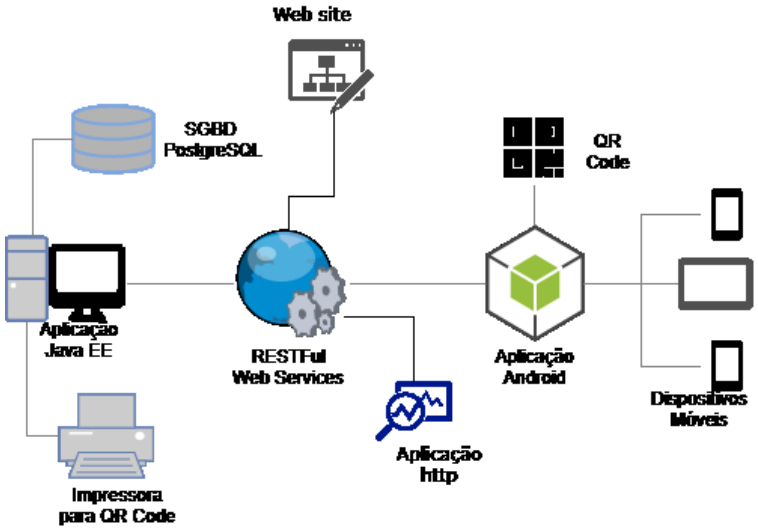


Figura 16: Arquitetura do projeto utilizando a implantação da arquitetura RESTFul.

Presente na mesma aplicação Java Web, o WebService QRservice disponibiliza informações sobre consumo de produtos e o saldo de cartões, permite a carga e o consumo de produtos, sempre após a leitura do QR Code pelos dispositivos móveis. Portanto, é por meio dela que o acesso dos dispositivos ao banco de dados torna-se possível.

O Web Service QRservice, desenvolvido sob o *framework* Jersey, realiza a interface entre os dispositivos móveis que contêm a aplicação QRmesseL e o banco de dados qrmessedb. Todavia, para que seja concedido acesso às funcionalidades disponibilizadas pelo QRservice, é necessário que o dispositivo tenha sido cadastrado na aplicação *web* e que o usuário esteja autenticado na aplicação. O sistema de *login*, que realiza a autenticação, foi implantado por meio do *framework* Spring Security. O Quadro 6 apresenta algumas das principais funcionalidades disponibilizadas pelo QRservice.

Quadro 6: Principais funcionalidades disponibilizadas pelo QRservice.

| **Função** | **Descrição** |
|---|---|
| getBarracaByMacDispositivo | Retorna a barraca pelo endereço MAC do dispositivo |
| postCarga | Realiza a carga de saldo no cartão |
| postConsumo | Realiza o pedido e o consumo de saldo do valor total do pedido |
| getProdutoByBarraca | Retorna a lista de produtos pelo id da barraca |

A Figura 17 contém o diagrama de sequência da carga de cartão. Já na Figura 18 o mesmo tipo de diagrama representa o consumo.

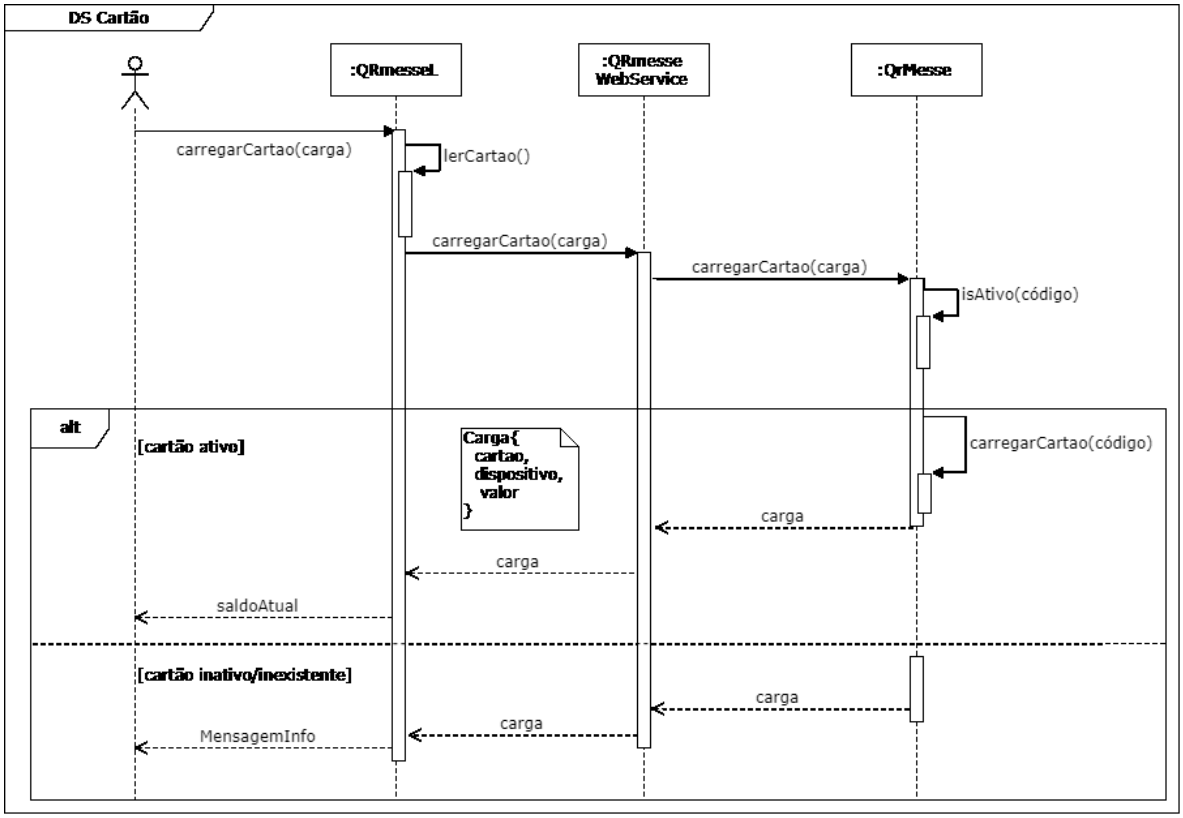


Figura 17: Diagrama de sequência da carga de cartão.

Portanto, ao receber uma solicitação de carga da aplicação Android, o sistema Web Service encaminha a solicitação ao sistema QRmesse, que primeiramente verifica se o cartão está ativo, caso esteja, a carga será efetuada, se não, a recarga não é efetuada. Finalmente, uma mensagem informando o novo saldo do cartão ou que a recarga não foi efetuada será retornada ao Web Service, e este retornará a mensagem à aplicação Android.

Um pouco mais complexo do que o processo de (re)carga é o processo de pedido, pois primeiramente precisam ser adicionados os produtos, e em seguida o cartão. Imediatamente após essa leitura, o pedido é enviado para o Web Service, validado e, se for o caso, o saldo será consumido, caso contrário, o sistema retornará o motivo da falha ao tentar realizar o consumo.

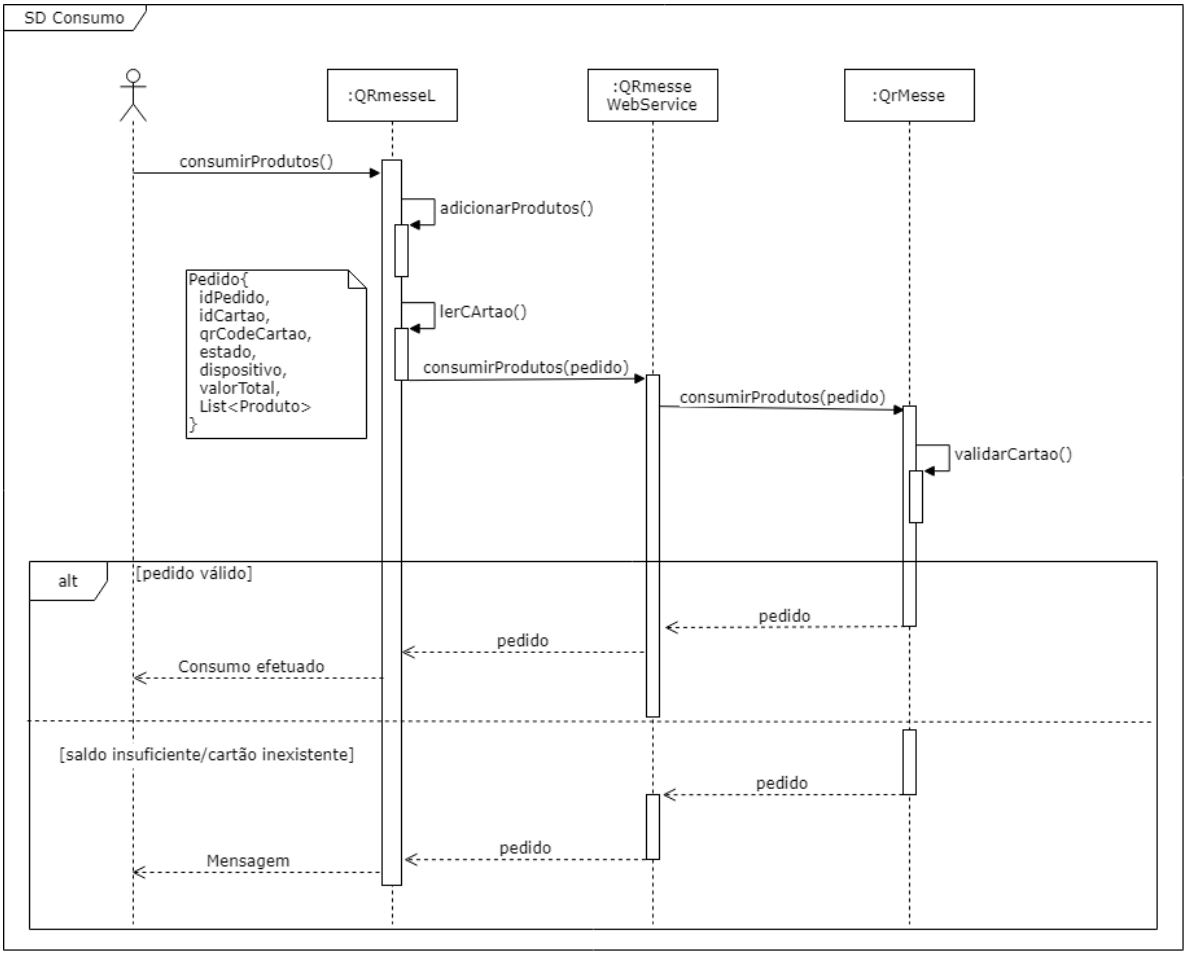


Figura 18: Diagrama de sequência do consumo de produtos.

## 3.5 APLICAÇÃO ANDROID

O aplicativo QRmesseL, desenvolvido no IDE Android Studio, permite a realização de consultas como saldo de cartão e de pedidos na fila para entrega. Além disso, a aplicação realiza a carga/recarga de saldo nos cartões e os pedidos de produtos que não são de pronta entrega. Para tanto, o dispositivo desempenha o papel de leitor de código QR Code. Essa leitura foi implantada com a utilização da biblioteca ZXing. As implantações de código referentes a acessos das funcionalidades do Web Service foram realizadas a partir da biblioteca HTTP Volley para Android, desenvolvida pela Google para simplificar a implantação de requisições de rede em aplicativos Android. Ademais, em alguns casos, o conteúdo das mensagens envolvendo o Web Service refere-se a objetos Java, então, a biblioteca Gson também está presente no aplicativo QRmesseL.

Como mencionado no referencial teórico, uma das vantagens de desenvolver aplicações para Android é a variedade de bibliotecas de código aberto disponíveis. A Figura 19 destaca as bibliotecas importadas, tecnicamente ditas dependências binárias remotas (ANDROID DEVELOPERS, 2018b), por meio de dependências adicionadas ao código do Gradle.

```
34  dependencies {
35      implementation fileTree(dir: 'libs', include: ['*.jar'])
36      implementation 'com.android.support:appcompat-v7:26.1.0'
37      implementation 'com.android.support.constraint:constraint-layout:1.1.1'
38      implementation 'com.android.support:design:26.1.0'
39      testImplementation 'junit:junit:4.12'
40      androidTestImplementation 'com.android.support.test:runner:1.0.2'
41      androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.2'
42
43      compile 'com.google.zxing:core:3.2.1'
44      compile 'com.journeyapps:zxing-android-embedded:3.2.0@aar'
45      compile 'com.android.volley:volley:1.0.0'
46      compile 'com.google.code.gson:gson:2.8.0'
47  }
```

Figura 19: Dependências binárias remotas adicionadas à aplicação QRmesseL

### 3.5.1 Leitura do QR Code no QRmesseL

A leitura do QR Code, na aplicação Android, utiliza a API ZXing (*zebra crossing*). A API ZXing é compatível tanto com aplicações Java quanto com aplicações Android. A utilização dos recursos presentes nessa biblioteca tem como objetivo realizar a leitura de QR Code utilizando a câmera de um dispositivo Android. Portanto, para utilizá-la, a aplicação solicitará ao usuário permissão de acesso à câmera. O código do método lerCartao, que inicializa o processo de leitura do QR Code está presente na Figura 20.

```
 99      private void lerCartao() {
100
101          IntentIntegrator integrator = new IntentIntegrator(this);
102          integrator.setDesiredBarcodeFormats(IntentIntegrator.QR_CODE_TYPES);
103          integrator.setPrompt("Camera Scan");
104          integrator.setCameraId(0);
105          integrator.setOrientationLocked(true);
106          integrator.initiateScan();
107
108      }
```

Figura 20: Trecho de código que ativa a leitura de QR Code no dispositivo móvel.

O resultado da leitura é retornado por meio de um objeto IntentIntegrator, na sobreposição do método onActivityResult (Figura 21). Quando for reconhecido um código QR Code pela câmera do dispositivo, esse código retornará como um objeto do tipo String (linha 117). Caso o operador do dispositivo móvel pressione o botão voltar, ou seja, cancele a leitura do cartão, aparecerá para ele a mensagem “Leitura cancelada” (linha 115).

```
110     @Override
111     protected void onActivityResult(int requestCode, int resultCode, Intent data) {
112         IntentResult result = IntentIntegrator.parseActivityResult(requestCode, resultCode, data);
113         if (result != null) {
114             if (result.getContents() == null) {
115                 Toast.makeText(this, "Leitura cancelada", Toast.LENGTH_LONG).show();
116             } else {
117                 String qrLido = result.getContents();
118                 solicitarSaldoCartao(qrLido);
119             }
120         } else {
121             super.onActivityResult(requestCode, resultCode, data);
122         }
123     }
```

Figura 21: Método que recebe o retorno do resultado da leitura do QR Code.

### 3.5.2 Requisições para o Web Service

A biblioteca HTTP Volley realiza a interface de comunicação com o Web Service. As requisições para o Web Service são realizadas em 9 pontos diferentes da aplicação por meio dessa biblioteca. A grosso modo, para utilizar o Volley, deve-se criar um RequestQueue e passar para ele os objetos Request. O RequestQueue gerencia *thread*s de trabalho para executar as operações de rede, como ler e gravar no *cache*, além de analisar as respostas. As requisições fazem a análise de respostas brutas e o Volley se encarrega de despachar a resposta analisada de volta para o *thread* principal, que então entrega o resultado (ANDROID DEVELOPERS, 2018c).

A Figura 22 apresenta o código da utilização da biblioteca Volley para realizar a verificação do estado do Web Service. Dessa forma, se as credenciais de *login*, preenchidas na aplicação, corresponderem ao cadastro de um usuário no sistema QRmesse, será exibida a tela de boas-vindas do aplicativo. Nota-se que, na linha 106, são repassadas as credenciais de *login* utilizadas para acessar o sistema por meio do método de autenticação básica HTTP do Spring Security.

```
82      private void createQRmesseAPI() throws InterruptedException {
83          String path =  new QRmesseAPI().getInstance().getUri() + "/hello";
84          StringRequest stringRequest = new StringRequest(Request.Method.GET,
85                  path,
86                  new Response.Listener<String>() {
87                      @Override
88                      public void onResponse(String response) {
89                          Intent intent = new Intent(MainActivity.this, InfoLoginActivity.class);
90                          startActivity(intent);
91                      }
92                  }, new Response.ErrorListener() {
93              @Override
94              public void onErrorResponse(VolleyError error) {
95                  Log.e("Rest response", error.toString());
96                  if (error instanceof TimeoutError){
97                      Toast.makeText(getApplicationContext(), "O servidor demorou muito para respo
98                  }else if(error instanceof NoConnectionError){
99                      Toast.makeText(getApplicationContext(), "Falha ao conectar com " + QRmesseAP
100                 }
101             }
102         }) {
103             @Override
104             public Map<String, String> getHeaders() throws AuthFailureError {
105                 HashMap<String, String> params = new HashMap<String, String>();
106                 String creds = String.format("%s:%s", username, password);
107                 String auth = "Basic " + Base64.encodeToString(creds.getBytes(), Base64.DEFAULT)
108                 params.put("Authorization", auth);
109                 return params;
110             }
111         };
112         VolleyContext.getInstance(this).addToRequestQueue(stringRequest);
113     }
```

Figura 22: Solicitação de *login* no aplicativo QRmesseL via Web Service.

# 4 RESULTADOS

Esta seção apresenta algumas telas das aplicações Java Web e Android desenvolvidas para realizar a interface entre os usuários e o sistema. Telas comuns a quase todo tipo de sistema como *login*, cadastro de usuários, cadastro de produtos, cadastro da instituição não estão presentes, por não serem consideradas relevantes para este trabalho. Portanto, apresentam-se apenas as telas que permitem a realização das funcionalidades particulares ao sistema QRmesse. O último tópico deste capítulo relata o resultado e as considerações de um evento teste, realizado em um estabelecimento comercial, durante um torneio de truco.

## 4.1 AMBIENTE DO PROJETO

A Figura 23 simula o cenário de um evento comunitário utilizando a solução computacional QRmesse. Nessa simulação, há um caixa para a realização da venda e ativação dos cartões dotados de QR Code, três barracas de entrega de produtos mediante baixa nos créditos do cartão, um terminal para consulta de saldo e uma área com o servidor onde está instalada a aplicação Java EE e o Web Service.

Os dispositivos móveis e o servidor precisam estar interligados pela rede *wireless*. Portanto, para a implantação do sistema, é necessário um roteador, cuja área de abrangência do sinal alcance a localização de todos os dispositivos. A partir disso, dois *softwares* são essenciais para a instalação do sistema no computador servidor: o Apache Tomcat Server, para hospedar a aplicação Java EE e o SGBD PostgresSQL, para gerenciar o banco de dados.

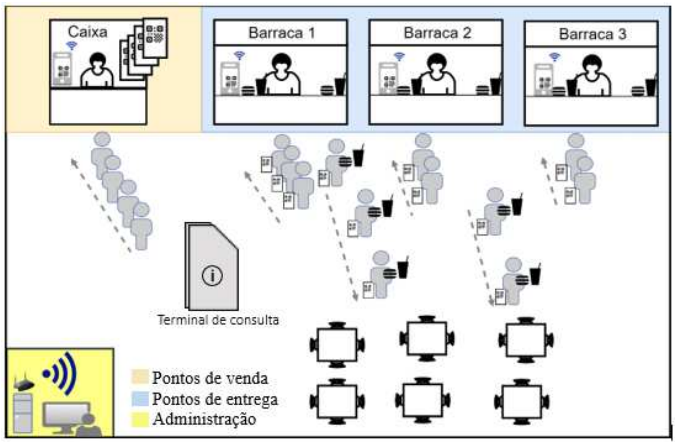


Figura 23: Ambiente de um evento comunitário utilizando o sistema QRmesse.

## 4.2 TELAS DA APLICAÇÃO JAVA EE

O desenvolvimento da interface na aplicação Java *web* foi facilitado pela utilização do *framework* PrimeFaces, que disponibiliza mais de 100 componentes para utilização em aplicações JSF. O Cadastro de Eventos (Figura 24) permite ao operador do sistema cadastrar os eventos e ter acesso às demais telas de cadastro, como Barracas, Caixas e Dispositivos, que pertencem ao evento selecionado. Além disso, pode-se visualizar a tela de Monitoramento do Evento (Figura 25).


Figura 24: Cadastro de eventos e menu das opções que pertencem ao evento selecionado.

Durante o evento, podem ser visualizados alguns números que refletem a situação geral do consumo de produtos. A tela de monitoramento mostra um gráfico com a quantidade vendida de cada produto e um gráfico com um ponteiro, indicando a quantidade total arrecadada, tendo como valor máximo a meta de arrecadação estipulada para o evento.

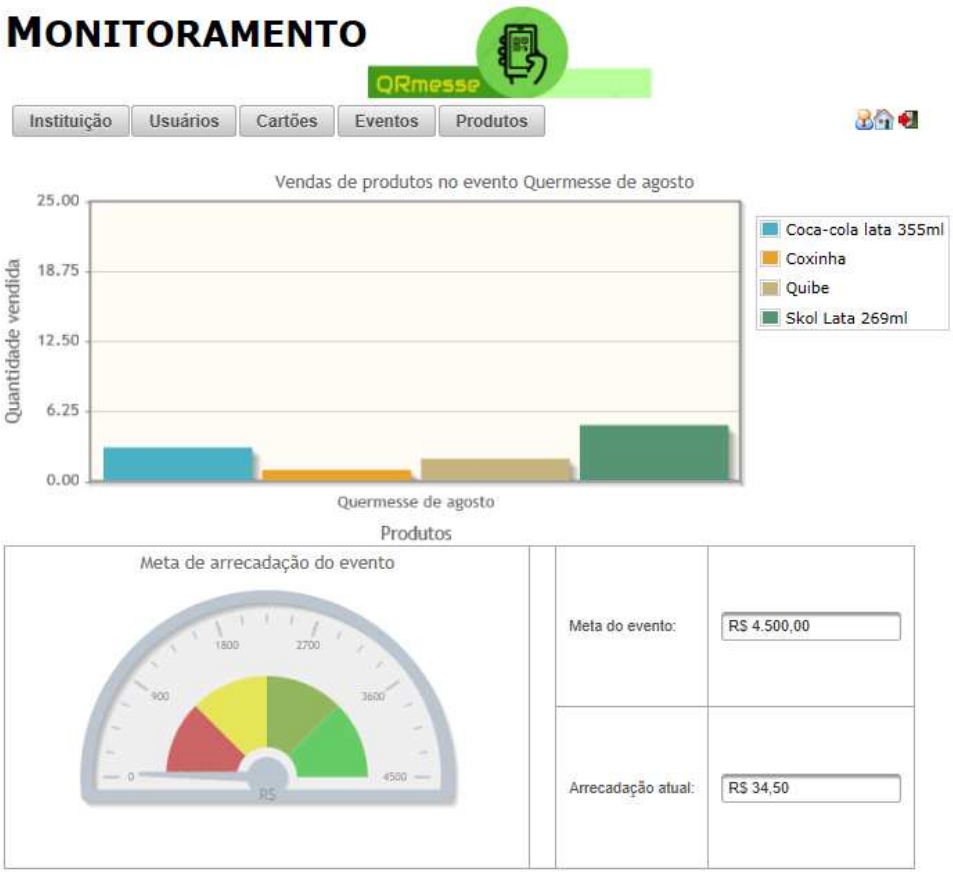


Figura 25: Tela de monitoramento do evento.

A tela de Cadastro de Cartões (Figura 26) permite ao operador do sistema gerar os cartões e, posteriormente, visualizar o saldo e o QR Code de cada um dos cartões gerados.


Figura 26: Cadastro de cartões.

No Cadastro de Barracas (Figura 27), além do nome da barraca devem ser definidos o gerente, que é um usuário cadastrado no sistema, e os produtos que a barraca disponibilizará aos participantes do evento. Nessa tela, os dispositivos podem ser apenas visualizados, pois a definição do local em que o dispositivo irá operar durante o evento é realizada na tela de Cadastro de Dispositivos.


Figura 27: Cadastro de barracas.

Para que somente dispositivos pertencentes ao evento realizem comunicação com o sistema, é necessário que eles sejam cadastrados. Portanto, é no Cadastro de Dispositivos (Figura 28) que o papel do dispositivo é definido, ou seja, se o dispositivo se comportará como uma ferramenta para a carga de saldo no cartão ou para o consumo de produto. Os pedidos realizados pelos dispositivos móveis podem ser acompanhados na tela de pedidos (Figura 29). Nela eles são filtrados conforme seu *status*. Pedidos entregues provêm de produtos cuja entrega ocorre no momento da leitura do cartão. Já os pedidos realizados ocorrem quando um produto não está pronto para a entrega, ou seja, necessita de preparo. Desse modo, o pedido entra em uma fila de espera. O *status* Preparado é opcional, ele seria utilizado quando no evento houvesse garçons. Assim, estes poderiam saber que há pedidos que devem ser entregues. Portanto, os

pedidos podem passar diretamente do *status* Realizado para Entregue.

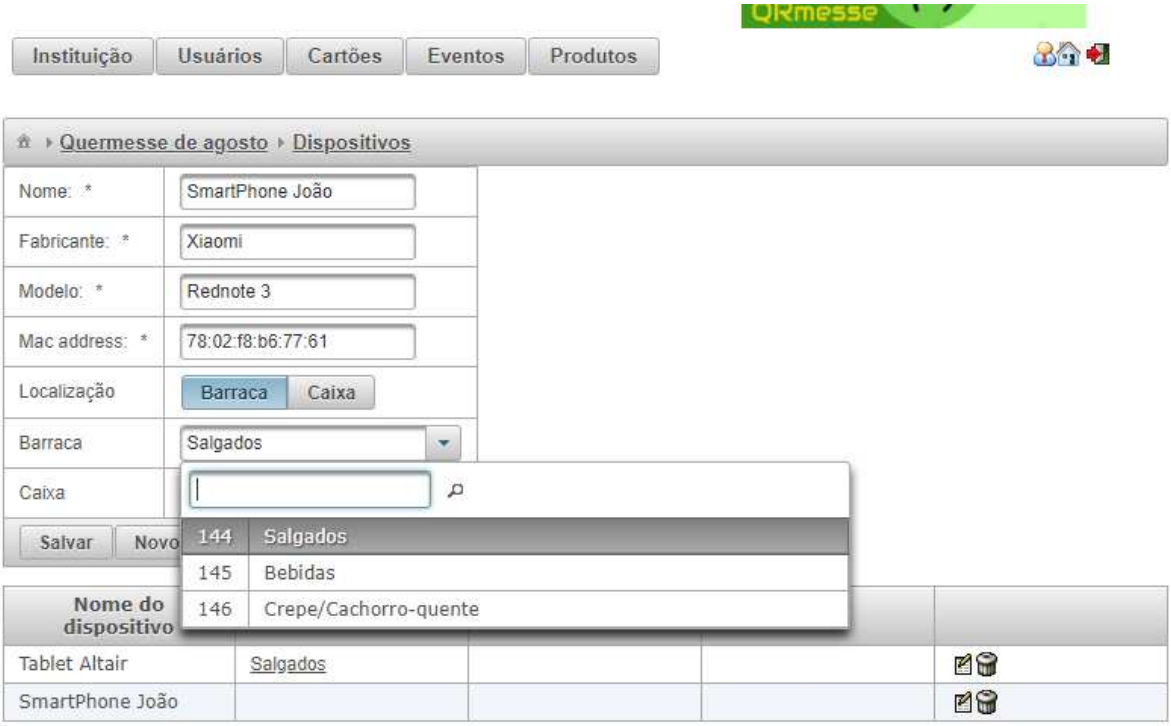


Figura 28: Cadastro de dispositivos.

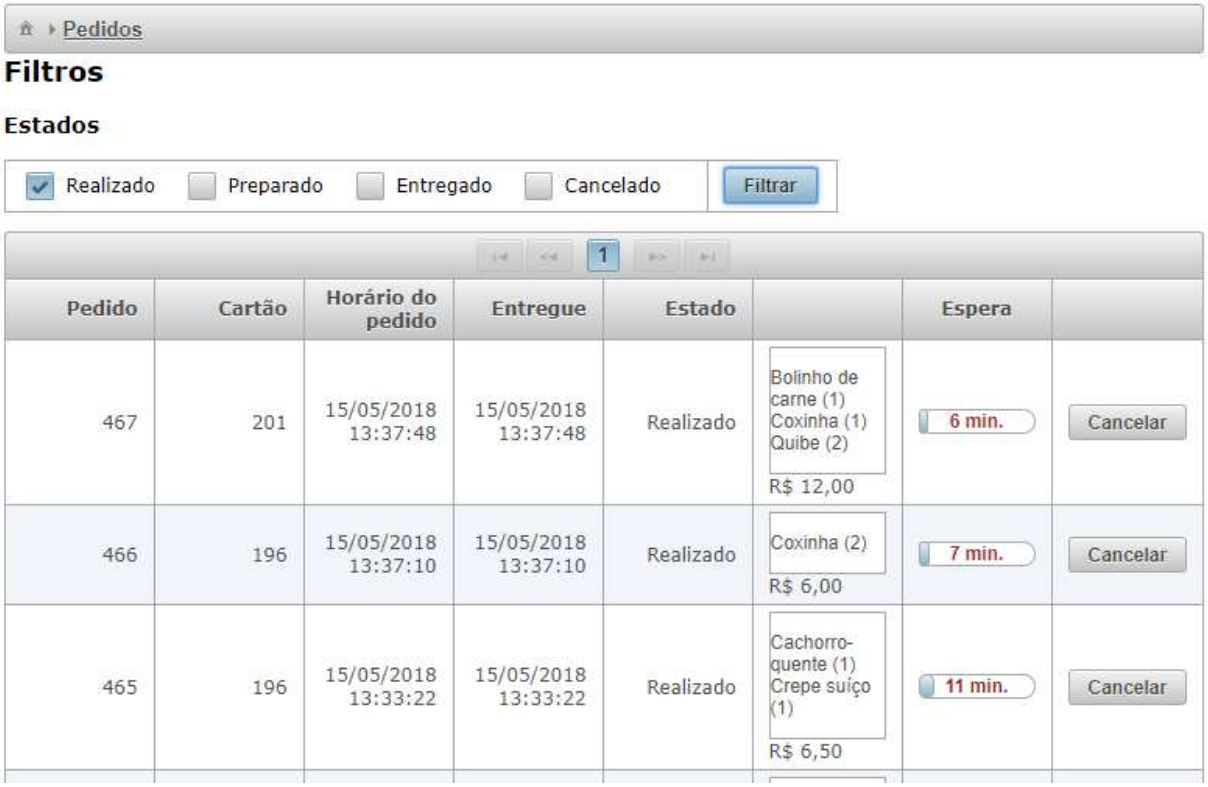


Figura 29: Acompanhamento de pedidos do evento.

## 4.3 APLICAÇÃO ANDROID

A aplicação para dispositivos móveis, desenvolvida no IDE Android Studio, tem como principal função permitir que aparelhos como s*martphones* ou *tablets* realizem o papel de leitores de QR Code, presente nos cartões dos participantes do evento. Como explicado, o dispositivo pode ser usado para duas finalidades: ou para a carga de saldo ou para o consumo de produtos. Após o processo de *login* bem-sucedido (Figura 30), a primeira tela que aparece para o portador do dispositivo, presente na Figura 31, informa qual é o evento, o local e quem fez o *login*.

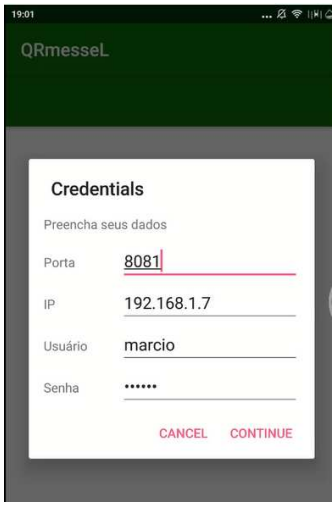


Figura 30: Credenciais de acesso ao sistema.

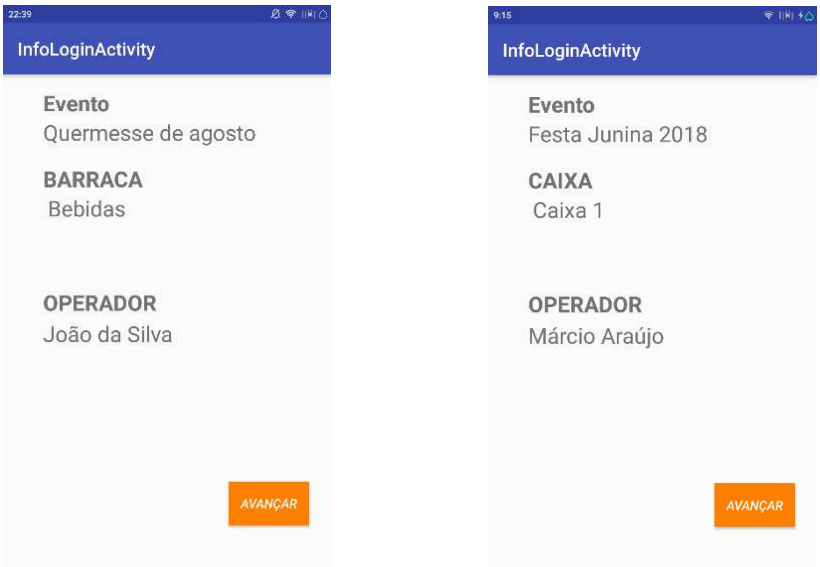


Figura 31: Tela inicial com informações sobre o dispositivo.

Após o clique no botão AVANÇAR, a próxima tela contém um menu com 3 opções. A primeira opção é diferente, conforme o local em que o dispositivo foi alocado. Se for dispositivo de Caixa, será Carga, caso seja dispositivo de Barraca, será Entregar (Figura 32). As próximas duas opções retornam informações sobre o saldo e o consumo do cartão.


Figura 32: Menu de opções para os dispositivos.

Nos dispositivos configurados para operar nos pontos de caixa do evento, deve-se primeiramente entrar com o valor de carga para em seguida ler o cartão. A Figura 33 exibe a tela para a entrada do valor de saldo que será carregado. Essa tela aparece após a escolha da opção Carga. Basta ao operador pressionar os botões + e – correspondentes a cada valor, e o sistema exibirá o valor da carga que será realizada. Para compor o valor de R$ 25,00, exibido na Figura 16, basta ao operador clicar uma vez no botão + da nota de R$ 20,00 e uma vez no botão + da nota de R$ 5,00.


Figura 33: Tela para entrada do valor de carga de saldo.

Ao pressionar o botão LER CÓDIGO, é solicitado ao operador do dispositivo que realize a leitura do cartão. Nesse momento, a câmera do dispositivo é acionada e então ele deve colocar o cartão em frente a ela, conforme exibido na Figura 34.


Figura 34: Leitura do cartão.

Nos dipositivos configurados para operar em barracas de produtos, após selecionar a opção Entregar, aparecerá uma tela (Figura 35) mostrando apenas imagens dos produtos associados à barraca do dispositivo. Abaixo de cada uma dessas imagens aparecem dois botões, + e -, para permitir a entrada da quantidade. Com isso, o sistema calcula o valor total do pedido e exibe-o na parte superior da tela. Em seguida, o operador clica no botão ENTREGAR ou no botão PREPARAR. Se for um produto de pronta entrega, escolhe-se a opção ENTREGAR, se ainda é necessário preparar o produto, deve ser escolhida a opção PREPARAR.


Figura 35: Tela para escolha de produtos para entregar ou preparar.

Após a leitura do cartão e o envio da solicitação ao servidor, é exibida uma das duas telas da Figura 36. Assim, o produto pode ser entregue imediatamente; colocado na fila de pedidos e entregue; ou o saldo do cartão é insuficiente.


Figura 36: Resultados da solicitação de consumo.

Ainda na tela de produtos, o operador do dispositivo pode clicar no botão PEDIDOS e ver a lista de pedidos que estão pendentes (Figura 38), ou seja, ainda não foram entregues. Após o clique no botão ENTREGAR da tela da Figura 38, o cartão deve ser lido novamente, a fim de confirmar que esse é o cliente que solicitou aquele pedido, se não for, será exibida a tela da Figura 38.


Figura 38: Lista de pedidos pendentes.

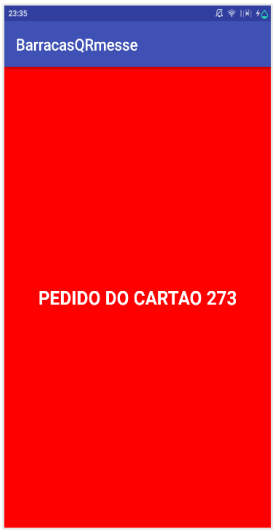


Figura 38: Erro na entrega de pedido.

Outra informação que pode ser obtida pelos dispositivos móveis é a lista de pedidos relizados para o cartão lido. A Figura 39 exibe um exemplo dessa lista.

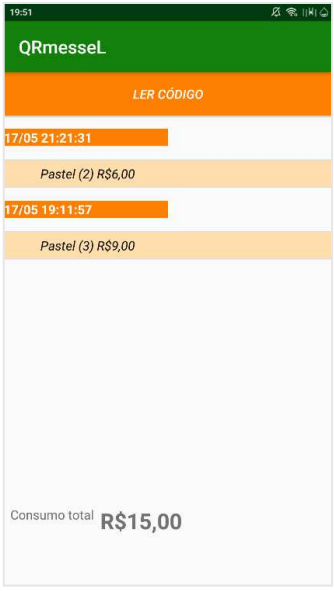


Figura 39: Pedidos efetuados pelo cartão.

## 4.4 VALIDAÇÃO DO SISTEMA

Na data do dia 12/05/2018, ocorreu um torneio de truco em uma lanchonete. Durante esse evento, foram comercializados produtos somente para clientes que utilizavam cartões com QR Code gerados pelo sistema QRmesse. Para poder consumir produtos durante o evento, os clientes precisavam adquirir um cartão ao custo de R$ 20,00, pois este era o valor do saldo incial para cada cartão.

No ambiente do estabelecimento foi ativado um roteador TP-LINK, modelo TR-WR740N e nele foi ativada uma rede *wireless* em que um *notebook* ASUS S46C, com os sistemas QRmesse instalado, além do SGBD PostgreSQL 9.6, e 4 dispositivos Android com o aplicativo QRmesseL instalado. Desses 4 dispositivos, um foi configurado para operar como caixa e os outros três como barraca, entretanto, para facilitar a entrega, todos com os três dispositivos estavam cadastrados para a mesma barraca.

Durante o evento foram realizados 78 consumos, para 26 cartões diferentes. Além do uso do sistema QRmesse, o operador do caixa também fez um controle manual, anotando tanto os saldos dos cartões quanto os pedidos realizados. Cada vez que um cliente saía do local

verificava se ainda havia saldo nos cartões e conferia com suas anotações, todas as conferências foram bem sucedidas. Mesmo sem prévio treinamento, os garçons que operaram os 3 dispositivos configurados para entrega de produtos conseguiram utilizar o sistema sem dificuldade.

Houve a necessidade de intervenção do desenvolvedor do sistema apenas uma vez, quando um dos garçons realizou desconto de saldo de cartões trocados. Nesse caso, o cliente A pediu o produto X e o cliente B o produto Y, porém o garçon inverteu o desconto. Para resolver a situação sem a intervenção direta no sistema, já que não existe a opção de cancelamento de consumo, foi sugerida a realização de carga para cada um no valor do produto incorreto e, em seguida, o desconto do produto correto.

Ao final do evento, foi solicitada a opinião dos operadores. Os garçons elogiaram o sistema, o operador de caixa sugeriu a implantação de tranferência de saldo entre cartões, pois a soma dos saldos de dois clientes possibilitariam a compra de um produto cujo valor era superior ao saldo de ambos. Além disso, para formalizar a validação do sistema, os usuários responderam um questionário para avaliação do uso do QRmesse, a partir do qual foram gerados os dados a seguir:

Quadro 7: Respostas do questionário de avaliação de usuários do QRmesse.

| **Quanto à facilidade de uso do QRmesse, você considera:** | | | | | |
|---|---|---|---|---|---|
| Opções | Muito fácil | Fácil | Moderadamente fácil | Difícil | Muito difícil |
| Respostas | 3 | 2 | 0 | 0 | 0 |
| **Quão rápida é a leitura de cartões QR Code no sistema?** | | | | | |
| Opções | Muito rápida | Rápida | Normal | Lenta | Muito lenta |
| Respostas | 2 | 3 | 0 | 0 | 0 |
| **A agilidade no atendimento ao consumidor utilizando o sistema QRmesse se comparado ao atendimento convencional torna-se:** | | | | | |
| Opções | Muito mais ágil | Mais ágil | Não se altera | Mais lenta | |
| Respostas | 1 | 3 | 1 | 0 | 0 |
| **Quanto às cores, tamanhos de fontes e disposição dos elementos nas telas do sistema, você considera que estão adequadas?** | | | | | |
| Opções | Totalmente adequadas | Adequadas | Normais | Inadequadas | Totalmente inadequadas |
| Respostas | 3 | 2 | 0 | 0 | 0 |
| **Qual é a sua percepção em relação à maioria dos clientes que realizaram consumo de produtos utilizando o QRmesse?** | | | | | |
| Opções | Elogiaram o uso do sistema com QR Code | | Mostraram-se indiferentes | Criticaram o sistema de cartões | |
| Respostas | 4 | | 1 | 0 | |
| **Você acha útil a aplicação desse sistema para agilizar pedidos em um evento?** | | | | | |
| Opções | Muito útil | Útil | | | Indiferente |
| respostas | 2 | 3 | | | 0 |

## 4.5 LIMITAÇÕES IDENTIFICADAS

Na versão atual do sistema QRmesse, não é possível o acesso por parte dos participantes do evento às informações sobre seus próprios cartões. Todavia, para implementar essa funcionalidade, seria adequado desenvolver outro Web Service, com acesso restrito a determinadas informações. Nesse caso, também poderia ficar disponível o acesso à posição do pedido na fila e até mesmo a realização de pedidos.

A impossibilidade de transferência de saldos, identificada na validação do sistema, também é uma limitação que pode ser resolvida. Entretanto, como o sistema foi desenvolvido para substituir fichas, e não comandas, tal funcionalidade pode não ser requerida em um evento no qual o seja utilizado um cartão por família.

# 5 CONSIDERAÇÕES FINAIS

Este trabalho apresentou tecnologias atuais utilizadas para o desenvolvimento de aplicações Java Web com Web Service, para realizar a integração com dispositivos móveis Android utilizados como leitores de QR Code.

A opção por desenvolver um banco de dados PostgreSql agregou ao sistema as funções PLpgSQL que facilitaram o desenvolvimento das duas aplicações, Java EE e Android, pois reduziu a necessidade de soluções como o cálculo do saldo de cartões, por exemplo.

Mesmo tendo a etapa de desenvolvimento mais dispendiosa, pois é nela que fica o núcleo do sistema, a aplicação Java Web foi considerada bem-sucedida. Ela gerencia de maneira satisfatória todos recursos do evento, como: os usuários, os dispositivos, os pontos de venda e a entrega e os produtos.

O uso de *frameworks* como o Spring Security para restringir o acesso apenas a usuários cadastrados, o EclipseLink para mapeamento objeto-relacional, o Primefaces para tornar a interface com o usuário mais amistosa e o Jersey para os recursos do Web Service agilizou o desenvolvimento da aplicação *web*. A implantação do Web Service RESTful também foi satisfatória, pois os pedidos podem ser realizados e visualizados pela aplicação cliente Android.

A utilização da biblioteca ZXing, na aplicação QRmesseL, permite a leitura de QR Code de forma eficiente. O envio dos pedidos e a solicitação de informações ao Web Service, com a inclusão da biblioteca HTTP Volley também cumpre o objetivo proposto de permitir a comunicação entre os dispositivos móveis e a aplicação Java Web. Todavia, essa aplicação é a que mais pode evoluir. Funcionalidades como visualização da lista de pedidos ou até mesmo a própria realização do pedido pelos participantes do evento podem ser implementadas.

Conforme os resultados apresentados, considera-se que recursos presentes em dispositivos móveis, como a facilidade de conexão com redes *wi-fi* e a utilização da câmera para realizar a leitura de QR Code, permitem a criação de um sistema computacional que gerencie de maneira ágil e organizada os processos de compra e venda de produtos em eventos comunitários, sem a necessidade de altos investimentos. Estima-se que o sistema pode melhorar a experiência dos participantes desses eventos, tanto dos que fazem parte da organização quanto dos consumidores.

## 5.1 TRABALHOS FUTUROS

Como possíveis trabalhos futuros podem-se apontar novas funcionalidades nos quatro componentes do sistema:

No banco de dados podem ser criadas funções PLpgSQL que realizem a replicação de dados de um evento para outro. Isso agilizaria a criação de novos eventos, importando barracas, produtos, dispositivos, gerentes de barracas e operadores de caixa.

Na aplicação *web* podem ser implementados recursos que aprimorem o monitoramento do evento, exibindo informações por meio de filtros.

Na estrutura Web Service podem ser inseridas funções que disponibilizem informações para os participantes do evento e realizem a transferência de saldo.

Na aplicação Android, além de funcionalidades que permitam o acesso aos dados pelos participantes do evento para verificação de saldo e itens consumidos, podem ser desenvolvidos recursos que permitam o autoatendimento e possibilitem ao participante a realização de pedidos pelo seu próprio *smartphone*.

# REFERÊNCIAS

ALMEIDA HOLANDA, M. I.; SENA DO NASCIMENTO, G. Um estudo das novas tecnologias a serviço da hotelaria de Fortaleza-CE. **Revista Ciências Administrativas**, v. 18, n. 2, 2012.

ANDROID DEVELOPERS. Conheça o Android Studio. **Android Developers**, 2018a. Disponível em: <https://developer.android.com/studio/intro/>. Acesso em: 30 maio 2018.

ANDROID DEVELOPERS. Configure variantes de compilação. **Android Developers**, 2018b. Disponível em: <https://developer.android.com/studio/build/build-variants#dependencies>. Acesso em: 20 maio 2018.

ANDROID DEVELOPERS. Conheça o Android Studio. **Android Developers**, 2018c. Disponivel em: <https://developer.android.com/training/volley/simple>. Acesso em: 01 jun. 2018.

BARROS, G. P.; LEMOS, A.; AMARAL, H. F. Solução em dispositivo móvel para atendimento de restaurantes e lanchonetes em Viçosa-mg. **ANAIS SIMPAC**, Viçosa-MG, v. 5, n. 1, 2015.

BEZERRA, A. C. A. Festa e cidade: entrelaçamentos e proximidades. **Espaço e Cultura**, n. 23, p. 7-18, 2012.

CHEONG, S. N.; CHIEW, W. W.; YAP, W. J. Design and development of Multi-touchable E-restaurant Management System. **Science and Social Research** (CSSR), 2010, International Conference on. IEEE, 2010. p. 680-685.

CHUANG, J.-C.; HU, Y.-C.; KO, H.-J. A novel secret sharing technique using QR code. **International Journal of Image Processing**, v. 4, n. 5, p. 468-475, 2010. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=E41525789B894D6CA2E4C19526FCD8BA?doi=10.1.1.740.3429&rep=rep1&type=pdf>. Acesso em: 6 nov. 2017.

CONTI, G. **Arquitetura e implementação de sig móvel embasado em conceitos da internet das coisas**. 2015. Dissertação (Mestrado) – Universidade Estadual de Ponta Grossa. Ponta Grossa.

DB-ENGINES. DB-Engines Ranking of Relational DBMS. **DB-Engines**, 2018. Disponível em: <https://db-engines.com/en/ranking/relational+dbms>. Acesso em: 18 maio 2018.

DEITEL, P.; DEITEL, H.; WALD, A. **Android 6 para Programadores**: Uma Abordagem Baseada em Aplicativos. 3.ed. Bookman, 2016.

DHORE, V. B. E. A. Digital Table Booking and Food Ordering System Using Android Application. **International Journal of Emerging Engineering Research and Technology**, v.

2, n. 7, p. 76-81, 2014.

ERL, T. **Service-oriented architecture:** concepts, technology, and design. Pearson Education India, 2005.

FÉLIX, L. D. O. **Proposta de identificação de Prontuário Eletrônico do Paciente (PEP) via smartphone usando QR-CODE**. Universidade de Brasília. Brasília. 2016.

FERNANDES, C. E. P.; MARCILIO, F. D. O.; MARQUES, F. C. **Coleta de dados para controle de estoque utilizando dispositivos móveis**. Pindamonhangaba: FUNVIC, 2015.

FERNANDEZ, N. D. N. **Escolas de Samba:** Sujeitos celebrantes e objetos celebrados. Rio de Janeiro: Arquivo Geral da Cidade do Rio de Janeiro, v. 3, 2001.

FIELDING, R. T. I. **Architectural Styles and the Design of Network-based Software Architectures**. University of California, 2000.

GAO, J. Z.; PRAKASH, L.; JAGATESAN, R. Understanding 2d-barcode technology and applications in m-commerce-design and implementation of a 2d barcode processing solution. **Computer Software and Applications Conference**, COMPSAC, 2007.

GARTNER, INC. IT GLossary. **Technology Resource | Gartner, Inc**. 2017. Disponível em: <https://www.gartner.com/it-glossary/service-oriented-architecture-soa>. Acesso em: 10 nov. 2017.

GONÇALVES, A. **Beginning Java EE 7**. Nova Iorque: Apress, 2013.

GS1 BRASIL. História. **Bem-vindo à GS1 Brasil**, 2017. Disponível em: <https://www.gs1br.org/sobre-a-gs1/Paginas/historia.aspx>. Acesso em: 1 nov. 2017.

HUI, M. K.; TSE, D. K. O que dizer a consumidores em esperas com durações diferentes: um modelo integrativo de avaliação de serviços. **Marketing de serviçoes**, Porto Alegre, p.226-237, 2001.

HURWITZ, J. et al. **Service oriented architecture for dummies**. Indianapolis: Wiley Publishing, 2007.

KRILL, P. JetBrains readies JVM-based language. **InfoWorld from IDG**, 2011. Disponível em: <https://www.infoworld.com/article/2622405/java/jetbrains-readies-jvm-based-language.html>. Acesso em: 02 jun. 2018.

KUMAR, B. V. **Implementing SOA Using Java EE**. India: Pearson Education, 2010.

KUMAR, B. V.; NARAYAN, P.; NG, T. **Implementing SOA Using Java EE**. Pearson Education, 2009.

LAYKA, V. **Learn java for web development:** Modern java web development. Apress,

2014.

MATTHEW, N.; STONES, R. **Beginning Databases with PostgreSQL.** 2. Berkeley, CA: Apress, 2005.

MAXIMIANO, A. C. A. **Administração para empreendedores:** fundamentos da gestão e da criação de novos negócios. São Paulo: Pearson Prentice Hall, 2006.

MEIRELLES, F. D. S. 28ª Pesquisa Anual do Uso de TI, 2017. **Fundação Getúlio Vargas**, 2017. Disponível em: <http://eaesp.fgvsp.br/sites/eaesp.fgvsp.br/files/pesti2017gvciappt.pdf>. Acesso em: 25 out. 2017.

MOMJIAN, B. **PostgreSQL:** introduction and concepts. New York: Addison-Wesley, 2001.

MONARIM, L. H. **Cardápio digital para restaurantes, bares e similares-MM+**. UTFPR. Londrina. 2013.

OLIVEIRA, C. D. M. D. **Dinâmicas das Festas Populares:** Sagradas, Profanas e Turísticas. Anais do II Colóquio Nacional do NEER. 2007.

ORACLE. **Agile PLM Core Web Services User Manual**. Oracle. 2010.

ORACLE. Introduction to Java EE. **Java Enterprise Edition**, 2017. Disponível em: <https://javaee.github.io/tutorial/overview001.html>. Acesso em: 3 nov. 2017.

ORACLE. The Java EE 6 Tutorial. **ORACLE Help Center**, 2017. Disponível em: <https://docs.oracle.com/javaee/6/tutorial/doc/gijqy.html>. Acesso em: 19 nov. 2017.

OSUNA, E. E. The psychological cost of waiting. **Journal of Mathematical Psychology**, v. 29, n. 1, p. 82-105, 1985.

PARRA, F. A influência do QR Code na reconfiguração da interação com o ciberespaço. **Revista de Estudos de Gestão, Informação e Tecnologia**, Itaquaquecetuba, v. 2, n. 4, p. 50-61, jul. 2015.

PATEL, K. J.; PATEL, U.; OBERSNEL, A. PDA-based Wireless Food Ordering System for Hospitality Industry – A Case Study of Box Hill Institute. **Wireless Telecommunications Symposium**. IEEE: 2007. p. 1-8.

POSTGRESQL. PL/pgSQL Overview. **PostgreSQL:** The world's most adcanced open source database, 2017. Disponível em: <https://www.postgresql.org/docs/9.6/static/plpgsql-overview.html#PLPGSQL-ADVANTAGES>. Acesso em: 13 nov. 2017.

POSTGRESQL. PostgreSQL 10.4 Documentation: Chapter 41. Procedural Languages. **PostgreSQL:** The World's Most Advanced open source relational database, 2018. Disponível em: <https://www.postgresql.org/docs/10/static/xplang.html>. Acesso em: 05 maio 2018.

QR CODE.COM. History of QR Code. **QR Code.com**, 2017. Disponível em: <http://www.qrcode.com/en/history/>. Acesso em: 1 nov. 2017.

ROMANO, R. R. **Os impactos do uso de tecnologia da informação e da identificação e captura automática de dados nos processos operacionais do varejo**. FGV-EAESP. São Paulo. 2011.

RUBIN, A. Where's my Gphone? **Google Official Blog**, 2007. Disponível em: <https://googleblog.blogspot.com/2007/11/wheres-my-gphone.html>. Acesso em: 04 abr. 2018.

SARKAR, S. E. A. Integration of Touch Technology in Restaurants using Android. **International Journal of Computer Science and Mobile Computing**, v. 3, n. 2, p. 721-728, 2014.

SENHORAS, E. M. O varejo supermercadista sob perspectiva. **Revista Eletrônica de administração**, v. 9, n. 3, mai-jun 2003.

STONES, R.; MATTHEW, N. **Beginning databases with PostgreSQL:** from novice to professional. Apress, 2006.

TANPURE, S. S.; SHIDANKAR, P. R.; JOSHI, M. M. Automated food ordering system with real-time customer feedback. **International Journal of Advanced Research in Computer Science and Software Engineering**, v. 3, n. 2, 2013.

THE ECLIPSE FOUNDATION. **Solutions Guide for EclipseLink (Beta Draft)**. 2014.

VANZ, N. M. **Um estudo sobre a evolução do código de barras linear até o qr code e sua aplicação em um estudo de caso**. Instituto Federal de Educação, Ciência e Tecnologia Sul-riograndense. Passo Fundo. 2012.

VARANASI, B.; BELIDA, S. **Spring REST**. Apress, 2015.

ZXING PROJECT. ZXing Project. **ZXing Project**, 2017. Disponível em: <https://github.com/zxing>. Acesso em: 13 nov. 2017.

# APÊNDICE I – Diagrama Entidade Relacionamento da base de dados