UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA
CURSO DE ESPECIALIZAÇÃO EM TECNOLOGIA JAVA

MAICON STANGHERLIN

APLICATIVO ANDROID COM PORTAL WEB PARA ACOMPANHAMENTO E DIVULGAÇÃO DE PREÇOS DE PRODUTOS

MONOGRAFIA DE ESPECIALIZAÇÃO

PATO BRANCO
2017

MAICON STANGHERLIN

APLICATIVO ANDROID COM PORTAL WEB PARA ACOMPANHAMENTO E DIVULGAÇÃO DE PREÇOS DE PRODUTOS

Monografia de especialização apresentado ao IV Curso de Especialização em Tecnologia Java, do Departamento Acadêmico de Informática, da Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco, como requisito parcial para obtenção do título de Especialista.

Orientador: Prof.Robison Cris Brito

PATO BRANCO
2017

MINISTÉRIO DA EDUCAÇÃO
Universidade Tecnológica Federal do Paraná
Câmpus Pato Branco
Departamento Acadêmico de Informática
Curso de Especialização em Tecnologia Java

TERMO DE APROVAÇÃO

APLICATIVO ANDROID COM PORTAL WEB PARA ACOMPANHAMENTO E DIVULGAÇÃO DE PREÇOS DE PRODUTOS

por

MAICON STANGHERLIN

Este trabalho de conclusão de curso foi apresentado em 26 de agosto de 2017, como requisito parcial para a obtenção do título de especialista em Tecnologia Java. Após a apresentação o candidato foi arguido pela banca examinadora composta pelos professores Robison Cris Brito (orientador), Beatriz Terezinha Borsoi e Vinicius Pegorini, membros de banca. Em seguida foi realizada a deliberação pela banca examinadora que considerou o trabalho aprovado.

Robison Cris Brito
Prof. Orientador (UTFPR)

Beatriz Terezinha Borsoi
(UTFPR)

Vinicius Pegorini
(UTFPR)

Robison Cris Brito
Coordenador do curso

A Folha de Aprovação assinada encontra-se na Coordenação do Curso.

RESUMO

STANGHERLIN, Maicon Stangherlin. Aplicativo Android com portal web para acompanhamento e divulgação de preços de produtos. 2017. 36f. Monografia (Trabalho de especialização) – Departamento Acadêmico de Informática, Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco. Pato Branco, 2017.

A utilização da Internet para fins comerciais tem crescido exponencialmente e pode-se dizer que a compra de produtos online já é algo comum em muitos países. O presente trabalho apresenta o desenvolvimento de um aplicativo para divulgação e acompanhamento de preços de produtos, permitindo que os clientes visualizem facilmente as variações de preço dos itens desejados utilizando o QR Code. Para a construção do projeto foram utilizados vários frameworks que auxiliaram no desenvolvimento da parte visual até as camadas de persistência de dados e a comunicação cliente-servidor. A plataforma Android foi utilizada para a elaboração do aplicativo *mobile* por utilizar código Java. Como resultado, obteve-se um sistema que atende aos requisitos definidos. A utilização dos frameworks escolhidos foi essencial para o desenvolvimento do projeto.

Palavras-chave: Android. QR Code. Divulgação de Preços. Web.

ABSTRACT

STANGHERLIN, Maicon Stangherlin. Android application as web portal to publicize and follow up product's price. 2017. 36f. Monografia (Trabalho de especialização) – Departamento Acadêmico de Informática, Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco. Pato Branco, 2017.

The use of Internet for commercial purposes has grown exponentially, and can be said that buying products online is common in many countries.This work presents the development of a mobile application to dissemination and monitoring of prices, allowing customers to easily visualize item price changes, using the QR Code.For the development of the project, was made use offrameworks that help from the visual part to the layers of persistence, and client-server communication.The Android platform was used to develop the mobile application. This platform uses Java.As result, a mobile application was developed and the defined requirements were fulfilled. The frameworks used were useful to develop the project.

Keywords: Android. QR Code. Price publication. Web.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTAGENS DE CÓDIGOS

LISTA DE SIGLAS

ADT	*Android Development Tool*
API	*Application Programming Interface*
CSS	*Cascading Style Sheets*
DOM	*Document Object Model*
FGV-SP	Fundação Getúlio Vargas de São Paulo
HTML	*HyperText Markup Language*
JPA	*Java Persistence API*
MVC	*Model, View, Controller*
ORM	*Object-Relational Mapping*
REST	*Representation State Transfer*
SGBD	Sistema de Gerenciamento de Banco de Dados
SGBDOR	Sistema de Gerenciamento de Banco de Dados Objeto-Relacional
SQL	*Structure Query Language*
URL	*Uniform Resource Locator*

SUMÁRIO

1 INTRODUÇÃO

Visando economia, as pessoas buscam promoções e melhores preços para os produtos e serviços que adquirem. Com o crescimento do comércio eletrônico a utilização de *sites* com a função de informar o usuário sobre preços é cada vez mais comum. Outro aspecto relacionado às promoções é o *marketing* que é responsável por parte significativa do sucesso das vendas.

Mesmo com a popularização do *e-commerce* grande parte do que é consumido pela população ainda vem do comércio local - no próprio município. Nesses ambientes as empresas pequenas, médias e grandes usam algum tipo de mídia para a divulgação dos produtos ou serviços oferecidos. Nas empresas de maior porte o *marketing* torna-se mais acessível, inclusive pela quantidade de recursos que elas podem despender para investir em divulgação, e com isso, obtém-se vantagens de venda sobre as demais.

Semelhante ao crescimento da Internet, o número de aparelhos celularesteve um aumento significativo nos últimos anos. Pesquisa realizada pela Fundação Getúlio Vargas de São Paulo (FGV-SP), apontou que em 2016 o número de *smartphones* em uso no Brasil chegou a 168 milhões, que representa crescimento de 9% em relação a 2015 (CAPELAS, 2016). Assim como os celulares, a Internet móvel também teve um grande avanço. Por conta disso, esta plataforma pode ser vista como um ótimo meio de propagação para promoções, divulgação de preços e outras informações relacionadas ao *marketing*.

Push Notification é a tecnologia utilizada no envio de mensagens para aplicativos em geral em que, normalmente, as mensagens são entregues em tempo real. Esta tecnologia se transformou na forma de comunicação mais eficiente que uma empresa pode ter com seus clientes (FABRICA DE APLICATIVOS, 2015). Uma mensagem *push* alcança 96% das pessoas em 5 minutos e é lida por 92% das pessoas nos primeiros 2 minutos (FISBHEN, 2016).

O aplicativo desenvolvido como resultado deste projeto, nomeado de MyOFFs, propõe-se a trabalhar no nicho do *marketing* de preços, a fim de divulgar e alertar os usuários sobre alterações no valor de mercadorias ou serviços utilizando como base a tecnologia *web* e Android. Para tal o projeto foi fragmentado aos seguintes objetivos:

a) Geral:

   Desenvolver uma aplicação Android que alerte os clientes interessados em um produto quando o seu preço sofrer alteração em termos de redução de preço.

b) Específicos:

- Utilizar o *QR Code* como forma de identificação dos produtos, permitindo,assim, que o cliente passe a acompanhá-lo lendo-o com o aplicativo Android;
- No portal *web* efetuar a geração do *QR Code* de cada produto, para que o fornecedor possa exibi-lo juntamente com o produto;
- Permitir que o cliente defina um “valor meta”em cada produto para o recebimento da alerta, sendo avisado somente quando isso ocorrer;
- Permitir que o fornecedor insira uma data como prazo de validade do valor promocional;
- Disponibilizar ao fornecedor índices como: número de clientes que acompanham o produto e valor médio desejado pelos clientes em determinado produto.

2 FERRAMENTAS, TECNOLOGIAS E PROCEDIMENTOS

Neste capítulo são apresentadas as ferramentas e as tecnologias utilizadas no desenvolvimento do aplicativo.

2.1 FERRAMENTAS E TECNOLOGIAS

As ferramentas e as tecnologias utilizadas para a modelagem e o desenvolvimento do projeto estão listadas no Quadro 1.

Ferramenta / Tecnologia	Versão	Disponível em	Aplicação
Bizagi Process Modeler	2.9.0.4	http://www.bizagi.com/pt/produtos/bpm-suite/modeler	Modelagem de processos.
Eclipse	Neon Release (4.6.0)	http://www.eclipse.org/neon/	*Integrated Development Environment* (IDE) para desenvolvimento da aplicação*web.*
Android Studio	2.3.1	https://developer.android.com/studio/index.html?hl=pt-br	IDE para desenvolvimento da aplicação*web.*
Bootstrap	3.3.7	http://getbootstrap.com/	*Framework* de estilizações de páginas por meio de *Cascading Style Sheets*(CSS).
JQuery	1.12.4	http://jquery.com/	Biblioteca JavaScript utilizada no desenvolvimento da interface.
PostgreSQL	x64 9.5	https://www.postgresql.org/download	Sistema Gerenciador de Banco de Dados
Hibernate	5.0.12	http://hibernate.org/orm/downloads/	Mapeamento Objeto-Relacional para persistência dos dados.
Spring	1.4.1	https://projects.spring.io/spring-framework/	*Framework Open Source* para a plataforma Java
Rest (com Spring Rest)	2.5.3	https://spring.io/guides/gs/rest-service/	Transferência de Estado Representacional – utilizado para a transferência de dados via serviço *web*.
ZXing	3.2.1	https://github.com/zxing/zxing	Biblioteca para processamento de código de barra de código aberto 1D e 2D.

Quadro 1 – Tecnologias e ferramentas utilizadas na modelagem e na implementação do aplicativo
Fonte: Autoria própria (2017).

2.1.1 Bizagi Process Modeler

Para a visualização do modelo de processo estabelecido para o sistema MyOFFs, foi utilizado o *software* Bizagi, que trabalha com diagramas *Business Process Management*(BPM) ou Modelagem de processos de negócio. O objetivo principal da

metodologia é modelar o fluxo de processos organizacionais e também os processos interorganizacionais (MARCHAND, 2009).

O uso do BPM traz benefícios quando aplicado corretamente, pois se trata de uma forma de gerir as empresas, por meio de um conjunto de práticas de gerenciamento de processos. Pode-se dizer que o uso do método também impacta de forma positiva na geração de produtos e serviços – como *softwares*, por exemplo -, pois, é possível identificar falhas nos processos e posteriormente solucioná-las (PEREIRA, 2008).

O Bizagi foi desenvolvido para organizar processos, é possível trabalhar com regras de negócio, indicar desempenho, monitorar atividades, entre outros. A instalação básica do *software* é gratuita, dispensando os gastos iniciais com compras de licenças de uso(CONSULTOR, 2011).Na Figura 1 é apresentado um *print screen* da tela de trabalho da ferramenta.

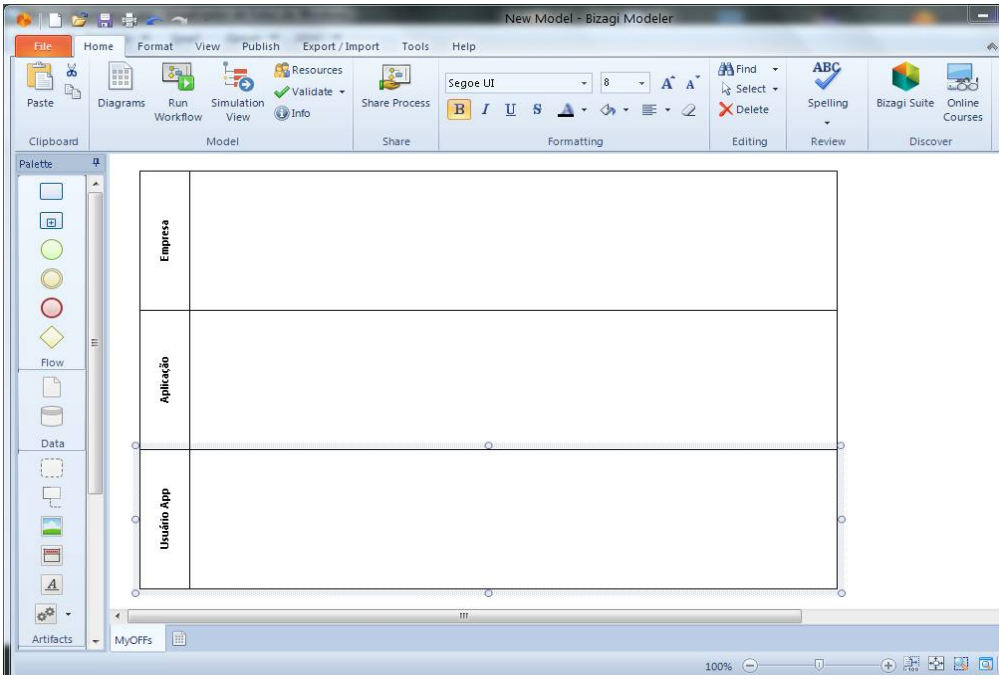


Figura 1 - Ambiente Bizagi: iniciando o modelo de processo
Fonte: Autoria própria (2017).

Para a construção do modelo de processo entre o aplicativo *mobile* e o portal *web*, desenvolvido como resultado deste trabalho, foram utilizados os recursos básicos do ambiente Bizagi. A versão utilizada do *software* foi a 2.6.0.4 (64 *bits*).

2.1.2Eclipse

O Eclipse é um Ambiente de Desenvolvimento Integrado –*Integrated Development Environment* (IDE). A flexibilidade e o fato de ser *open source*fazem com que o Eclipse seja uma das IDEs mais utilizadas na plataforma Java. Ele também é multilinguagem, ou seja, após instalar os devidos *plug-ins* ele suporta outras linguagens como, C, C++(FARIA, et al., 2010).

Com a tecnologia dos *plug-ins* utilizada no Eclipse é possível personalizar o ambiente de trabalho de acordo com o projeto que será desenvolvido, independentemente de sua complexidade(ANISZCZYK; GALLARDO, 2012). Na Figura 2 é apresentada a tela principal do Eclipse.

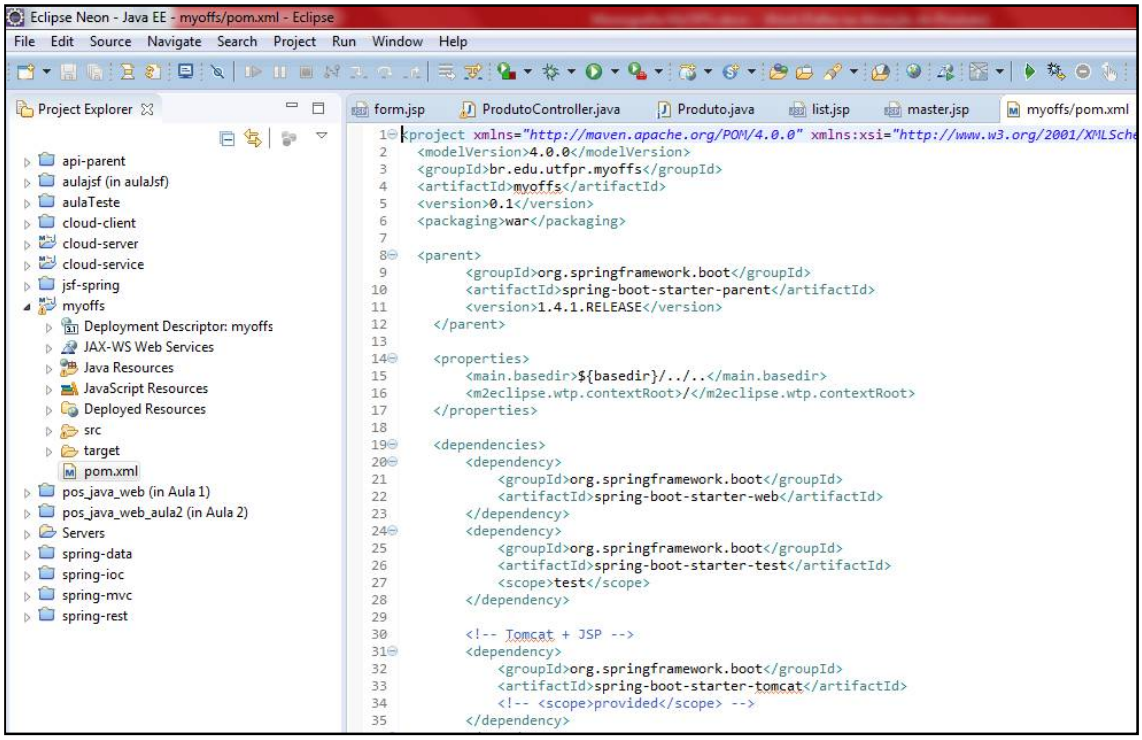


Figura 2 - IDE Eclipse
Fonte: Autoria própria (2017).

Toda a codificação *web* do projeto foi elaborada na IDEEclipse. Desde o início foram utilizados os recursos da estrutura de projeto Maven, que possibilita o gerenciamento das dependências do projeto, *builds*, mantendo um ambiente padronizado seguindo boas práticas possibilitando também um ganho de qualidade (SASSO, s.d.).

2.1.3Android Studio

O Android Studio foi lançado em maio de 2013 e é fruto da parceria entre a Google e JetBrains, já conhecida pelo desenvolvimento do IntelliJ IDEA. O ambiente de desenvolvimento integrado oficial para aplicativos Android é baseado no IntelliJ IDEA. Dentre as características deste software estão (SILVESTRE, 2015):

a) Programação baseada em Gradle, que é um sistema de automatização de *builds*;
b) Dispensa do uso de *plug-ins* como *Android Development Tool* (ADT), Ferramentas de Desenvolvimento Android;
c) Ambiente para desenvolvimento, *debug*, testes, emuladores;
d) Ferramenta para *preview* dos leiautes, entre outras.

A Figura 3 apresenta a tela principal do Android Studio.

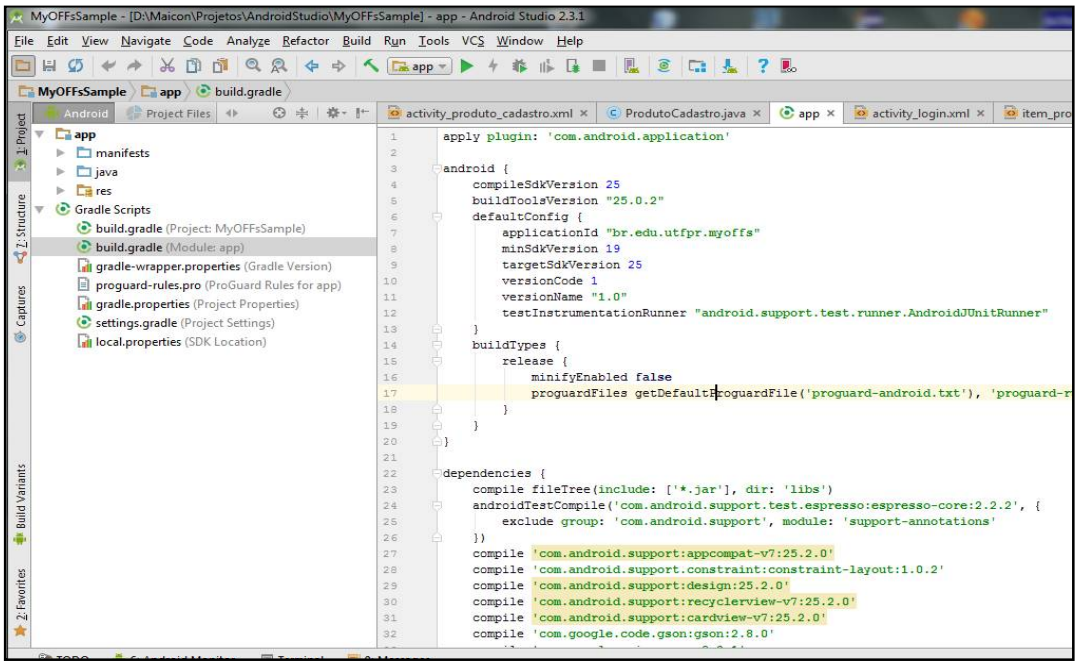


Figura 3 - IDE Android Studio
Fonte: Autoria própria (2017).

O Android Studio foi desenvolvido especialmente para a criação de aplicações Android e, por isso, disponibiliza uma gama de recursos facilitadores para os programadores. Ao incluir novas *activities*(similar a um formulário na programação *desktop*), por exemplo, é possível definir o *template* para esta *activity*. Assim, telas com menus, abas, mapas, definições de *login*, podem ser criadas com poucos cliques. Ou seja, muitos recursos foram adicionados a

IDE e apresentados a comunidade Android que passaram a utilizar o *software* como um dos principais motores de desenvolvimento da plataforma.

2.1.4Bootstrap

O Bootstrap é um *framework front-end* criado pela equipe de desenvolvedores do Twitter com a finalidade de otimizar o tempo para o desenvolvimento de páginas *web*. Nele são disponibilizados vários recursos visuais (CSS, ícones), estruturais (*grids*, navegação) e dinâmicos (JavaScript, Ajax) (PLATAS, 2013). Outra importante característica dessa tecnologia é a responsividade, que permite que leiautes de páginas *web*passam a ser realinhados/ajustados para as diferentes resoluções dos dispositivos eletrônicos.

Para desenvolver páginas com todas as vantagens da ferramenta é necessário seguir alguns padrões que constam na documentação do Bootstrap. Um dos principais é sistema de *grids*, sendo que a tela do navegador é dividida em 12 colunas e o tamanho dos componentes deve representar um número de colunas. É com base nesse sistema que o *framework* permite facilmente ajustar o leiautede acordo com o tamanho da tela (NASCIMENTO, 2013).

Mesmo tendo um papel importante na flexibilidade e aumento de produtividade do *front-end* ele traz algumas desvantagens. Entre elas pode-se citar: o código deverá seguir os padrões do Bootstrap e como este *framework* é utilizado em muitos *sites*, o leiaute tende a manter um modelo comum aos demais, sem uma identidade definida.

2.1.5jQuery

A jQuery é uma biblioteca JavaScript que foi desenvolvida para ter suporte a qualquer navegador *web*. Cada navegador possui um conjunto de características de implementação próprias o que dificulta a codificação, e por isso, tal biblioteca é importante para a simplificação e a transparência do código fonte. O objetivo da sua criação foi facilitar o desenvolvimento de aplicações que necessitam da linguagem JavaScript (RICARDO, s.d.).

Uma das características da jQuery é o suporte ao CSS3, sendo possível utilizar seletores CSS mesmo que o navegador não tenha suporte. Isso porque a própria biblioteca implementa esses seletores, fazendo com que independa do navegador. Além disso, a jQuery possui outras utilidades, como (RICARDO, s.d.):

- Adicionar efeitos visuais e animações;
- Acessar e manipular o Modelo de Objeto de Documento (*Document Object Model* – DOM);
- Carregar componentes Ajax;
- Prover interatividade;
- Alterar conteúdos;
- Simplificar tarefas JavaScript.

A biblioteca jQuery foi apresentada a comunidade no ano de 2006 e desde o início disponibilizada como *software* livre e aberto. (SILVA, s.d.).

2.1.6PostgreSQL

O PostgreSQL, normalmente chamado de Postgres, é um Sistema Gerenciador de Banco de Dados Objeto-Relacional (SGBDOR), desenvolvido pelo Departamento de Ciência da Computação da Universidade da Califórnia. Uma das características significativas desse banco de dados é o fato de ter seu código fonte aberto, podendo ser utilizado, modificado e distribuído por qualquer pessoa e para qualquer finalidade (POSTGRESQL, s.d.).

O PostgreSQL é um Sistema Gerenciador de Banco de Dados (SGBD) de código aberto considerando entre os mais avançados, contando com recursos como (REDAÇÃO OFICINA, 2008):

- Consultas complexas;
- Chaves estrangeiras;
- Integridade transacional;
- Controle de concorrência;
- *Trigger;*
- *Views;*

Esse SGBD otimizado para aplicações complexas, ou seja, aplicações que trabalham com grandes volumes de dados ou com informações críticas, sendo indicado, inclusive para sistemas de comércio eletrônico de médio e grande porte (ALECRIM, 2008).

2.1.7Hibernate

O Hibernate é um *framework*de Mapeamento Objeto Relacional (*Object Relational Mapping* - ORM), utilizado como ferramenta para persistência de objetos Java em um banco de dados relacional.

O presente trabalho utilizou a versão 4.1.7 Final do *framework*. Para utilizar dos recursos do Hibernate é necessário definir como os objetos são mapeados nas tabelas do banco. A partir disso o Hibernate faz todo o acesso ao banco, inclusive os comandos *Structure Query Langrage* (SQL) necessários.

Esse *framework* é um projeto *open-source* do grupo JBoss. Recentemente, o Hibernate e outros *frameworks* foram padronizados em uma especificação oficial do Java, API de Persistência Java (*Java Persistence API* - JPA)-(CAELUM, 2014).

2.1.8*Representation State Transfer* (REST)

Uma das maneiras mais fáceis de integração entre sistemas de informação é utilizando a tecnologia *Web Services*em conjunto com o *Representation State Transfer* (REST). Este é um modelo de comunicação baseado em requisições *Hypertext Transfer Protocol* (HTTP). A Figura 4 apresenta um gráfico comparando o uso dos principais modelos de comunicação.

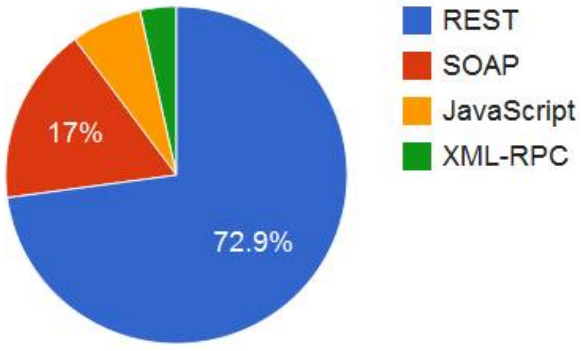


Figura 4 - Modelos de *Web Service* mais utilizados na *web*.
Fonte: (WAGNER, 2011, p.s.n.).

A estrutura REST é baseada no *design* do protocolo HTTP, que possui recursos para representação de cabeçalho, tipo de conteúdo e*status*. Um dos principais aspectos da arquitetura é a utilização de métodos do HTTP para comunicação, que são(PALIARI, 2012):

- *GET*: tem como função solicitar a representação de um determinado recurso;
- *POST*: tem como papel fazer processamentos que são relacionados a recursos, ou também, para utilizar as informações enviadas na requisição e criar um novo recurso;
- *DELETE*: tem como funcionalidade a remoção de um recurso;
- *PUT*: ele pode ser utilizado para atualizar um novo recurso.

A utilização dessa tecnologia em relação ao projeto em desenvolvimento é de extrema importância, uma que vez que, utilizando esse método, é possível fazer com que dados sejam transitadosentre plataformas como *mobile* e *web*,por exemplo.

2.1.9ZXing

ZXing é uma biblioteca *Open Source* desenvolvida em Java para a digitalização de código de barras unidimensional (1D) e bidimensional (2D). Ela tem suporte a vários códigos de barras como, por exemplo: EAN-8, EAN-13, Code 128, ITF, RSS-14, PDF 417 e o código QR Code, escolhido para o desenvolvimento deste projeto (THIENGO, 2014). A biblioteca está disponível para diversas plataformas. No entanto, neste projeto, a ZXing foi utilizada apenas na plataforma Android.

Para a utilização deste recurso é necessário adicionar a referência da biblioteca disponibilizada em um pacote da Google (‘com.google.zxing:core:3.2.1’). No Android, a biblioteca utiliza unicamente os recursos da câmera chamando posteriormente um método que fica aguardando a leitura ou o cancelamento do processo.

3 RESULTADOS

Este capítulo apresenta os resultadosdo presente trabalho. Este está dividido em: escopo do sistema, modelagem, apresentação do sistema e alguns dos principais códigos fontes.

3.1 ESCOPO DO SISTEMA

O aplicativo visa melhorar a acesso às promoções e aos reajustes em preços de produtos e serviços, além de disponibilizar um meio comum para divulgação de preços, independentemente do porte da empresa. Para cumprir essa tarefa o aplicativo receberá informações cadastradas pelo fornecedor e as divulgará aos clientes interessados, mantendo-os atualizados sobre a variação dos preços e de eventuais promoções.

O processo se inicia com o fornecedorcadastrando do produto ou serviço em um portal *web*. O cadastro é utilizado para a identificação do item, sendo informados dados pré-definidos, além de, gerar um *QR Code*. Esse *QR Code* é utilizado juntamente com o aplicativo para o sistema operacional Android. Após o usuário digitalizá-lo ele recebe todas as informações daquele produto e passa a acompanhar o seu preço, permitindo também informar um preço meta para o recebimento de alertas.

Na tela inicial do portal *web é* exibido um *dashboard* contendo informações de caráter gerencial como: número de clientes seguindo os produtos e os serviços da empresa, itens mais seguidos com a média de preço indicado nas alertas e marcação geográfica da densidade de clientes seguidores da empresa. A maioria das informações é enviada do aplicativo para o servidor *web*.

Por parte do aplicativo *mobile* o usuário tem basicamente as funcionalidades: ler o *QR Code* para obter as informações do produto e segui-lo, e cadastrar preço meta de produtos e serviços para recebimento de alertas. Para acessar o aplicativo, o usuário deve estar com o aparelho conectado à Internet e informar um *email* válido. Os dados cadastrados serão persistidos no servidor *web,* economizando memória interna do *device* Android.

3.2 MODELAGEM DO SISTEMA

Para o desenvolvimento do aplicativo *mobile* e *web* foram definidos alguns requisitos, listados nos Quadros 2.

Identificação	Nome	Descrição
RF001	Manter produtos – *web*	Exibir opção para incluir/editar/excluir produtos.
RF002	Manter serviços – *web*	Exibir opção para incluir/editar/excluir serviços.
RF003	*QR Code – web*	Gerar *QR Code* ao incluir um novo item no portal.
RF004	*QR Code* paraimpressão – *web*	Exibir botão para imprimir o *QR Code* em tamanho A4.
RF005	*Dashboard – web*	Na tela principal exibir índices sobre os itens publicados no portal.
RF006	Primeiro acesso - *mobile*	Exibir tela de autenticação (*login)* para preenchimento de um *email* válido.
RF007	Meta de preço – *mobile*	Na tela de principal exibir opção para o usuário cadastrar um preço meta para o recebimento de alertas.
RF008	Atualizar preços – *mobile*	Exibir botão para verificar se há novidades nos preços.

Quadro 2 - Requisitos funcionais
Fonte: Autoria própria (2017).

No Quadro 3 estão os requisitos não funcionais definidos.

Identificação	Nome	Descrição
RNF001	Persistência de dados	Todos os dados deverão ser armazenados no Banco de Dados PostgreSQL do servidor *web*.
RNF002	Texto *QR Code*	O texto do *QR Code* deverá receber o código do item no banco de dados.
RNF003	Autenticação no aplicativo	O *email* utilizado como *login* (conforme RF006), deverá ser armazenado na memória do celular (*shared preferences*), para que nos próximos acessos o aplicativo não exiba mais a tela de autenticação no sistema.

Quadro 3 - Requisitos não funcionais
Fonte: Autoria própria (2017).

Baseado nos requisitos funcionais e não funcionais selecionados para a criação do projeto, criou-se o diagrama de caso de uso conforme a Figura 5.

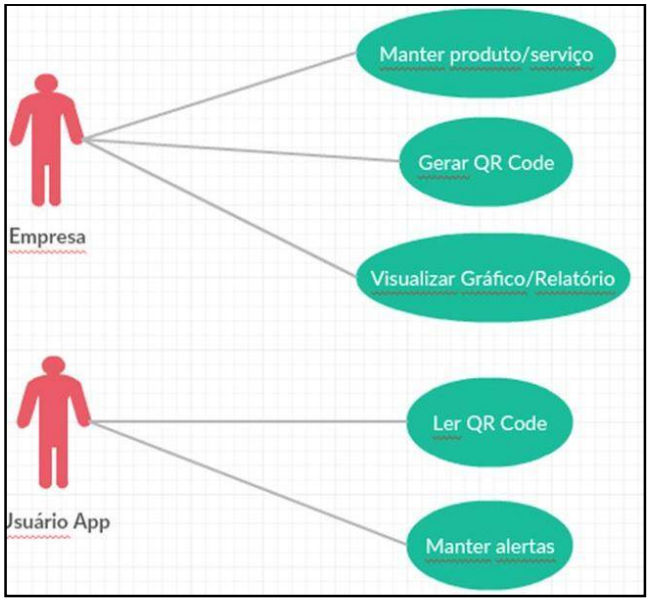


Figura 5 - Diagrama de caso de uso
Fonte: Autoria própria (2017).

Com o intuito de auxiliar na visualização do funcionamento da ferramenta como um todo foi utilizado o modelo de processo. O diagrama foi dividido em três partes, sendo dois atores e a aplicação, conforme trata a Figura 5. Esta figura apresenta alguns processos automatizados do sistema.

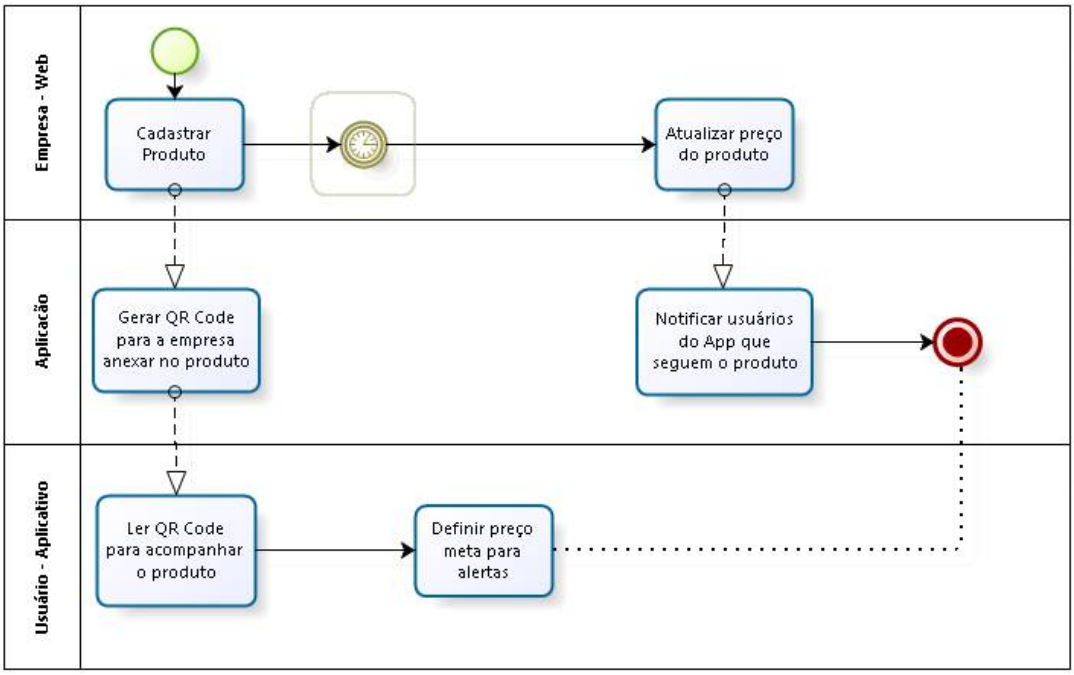


Figura 6 - Modelo de processo do MyOFFs
Fonte: Autoria própria (2017).

O primeiro quadro da Figura 5 representa o usuário do portal *web* sendo que as responsabilidades do usuário estão centradas no cadastramento de produtos/serviços e atualizações dos preços. Neste, após o usuário cadastrar as informações o sistema já permite que os clientes que utilizam o aplicativo possam digitalizá-los, isso porque, o segundo quadro foi inserido a fim de apresentar os processos principais que são executados automaticamente pela aplicação.

O processo realizado pelo cliente (usuário do aplicativo), será ler o *QR Code* de um anúncio e terá a opção de definir metas de preço para o recebimento de alertas. As alertas neste caso serão geradas por um processo manual do usuário, clicando em um botão de "Verificar atualizações", por exemplo, a fim de simular o recebimento de alertas no formato *Push*.

3.2.1Estrutura do Projeto

Para a elaboração do projeto *web* optou-se pelo desenvolvimento em três camadas: Modelo, Visualização e Controle (MVC). Seguindo esse modelo, a estrutura do código também foi organizada separando as classes de acordo com a função de cada uma. A Figura 7mostra como as classe foram distribuídas entre os pacotes.

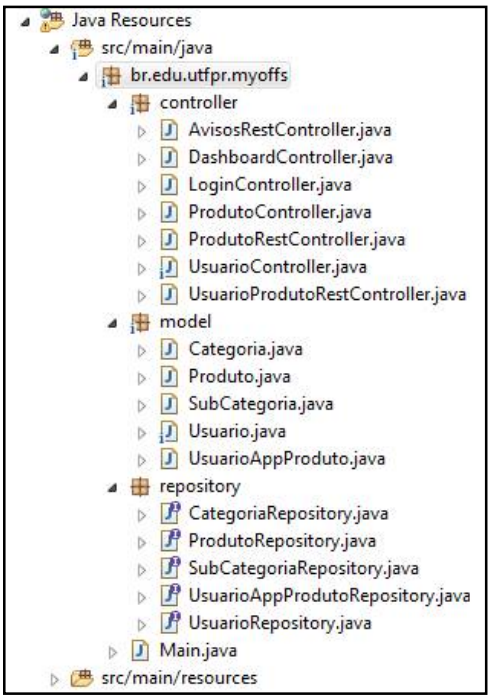


Figura 7 – Estrutura do código do projeto *web*
Fonte: Autoria própria (2017).

O pacote *Controller* contém as classes responsáveis por receber as requisições do sistema *web* e também do aplicativo *mobile*, representador pelas classes *Rest*. Os métodos declarados nestas classes definem qual *Model* usar e qual *View* será mostrado ao usuário.

No pacote *Model*há as classes responsáveis pelos modelos de objetos utilizados. No caso do projeto MyOFFs, as classes de modelo também possuem alguns métodos responsáveis por validações para a manipulação dos objetos antes de persisti-los no banco de dados, tarefa realizada pelas classes do pacote *Repository*.

A classe *Main* é a classe principal do projeto, pois, é a partir dela que a aplicação é inicializada. Para realizar esta função, a classe herda alguns métodos de configurações do *framework* Spring.

3.3 APRESENTAÇÃO DO SISTEMA

De modo a facilitar a visualização das funcionalidades do sistema MyOFFs a apresentação será feita em duas etapas, exibindo primeiramente as telas da aplicação *web* e na sequência as telas do aplicativo *mobile.*

3.3.1 MyOFFs - *Web*

O sistema irá trabalhar com dois tipos de usuário: a empresa distribuidora do produto ou serviço e os clientes interessados nos mesmos. Para essa empresa gerenciar as informações a serem divulgadas ela terá à disposição o portal *web*. Na primeira página do portal, será exibido um *dashboard* com alguns indicadores cadastrais e gerenciais. A tela principal do portal *web* é apresentada na Figura 7.

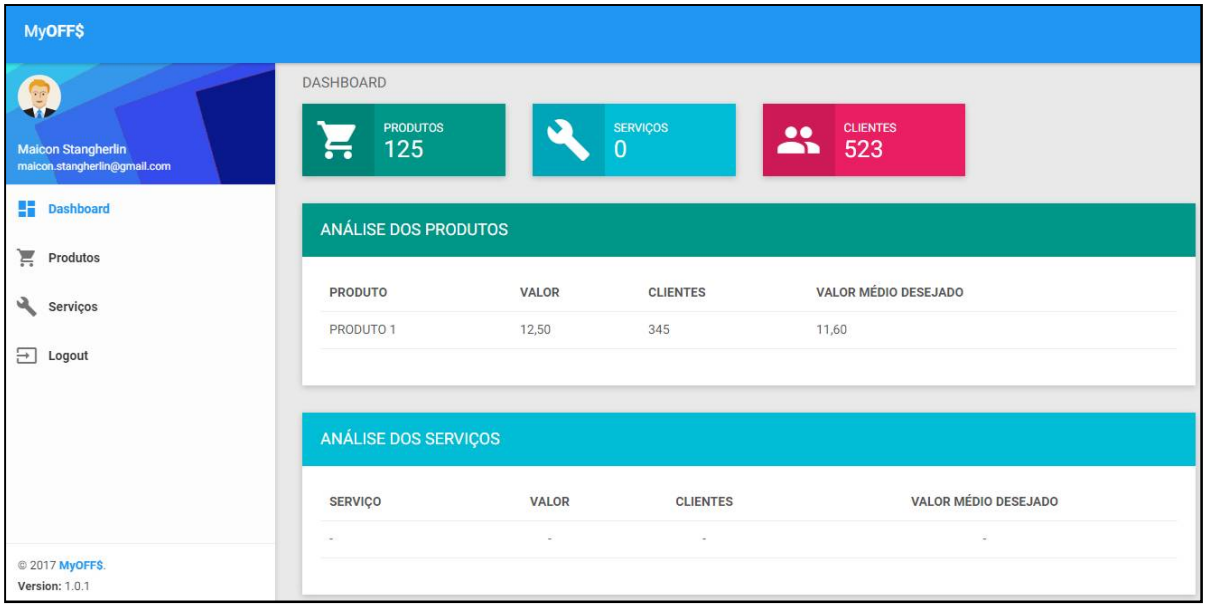


Figura 8 - *Dashboard* sistema *web*
Fonte: Autoria própria (2017).

Conforme mostra a Figura 7 (índices fictícios), os três primeiros indicadores exibem informações cadastrais, ou seja, apenas contagem de registros salvos no banco de dados. Abaixo são exibidas duas tabelas referentes à análise dos itens seguidos por cliente em cada segmento. Ambas as tabelas exibem informações estratégicas sobre os itens mais seguidos pelos clientes da empresa, sendo possível identificar a média de “preço meta” registrada pelos usuários do aplicativo.

A identificação da região e da localização dos clientes pode ser de fundamental importância para uma empresa, podendo direcionar o *marketing*para determinadas áreas de uma cidade, por exemplo. Para este tipo de análise foi inserido um mapa de densidade sendo exibidos círculos de acordo com o número de clientes da região. O mapa da Figura 8 contém dados ilustrativos. Ressalta-se que a coleta da localização do cliente por parte do aplicativo Android será tratada como implementação futura.

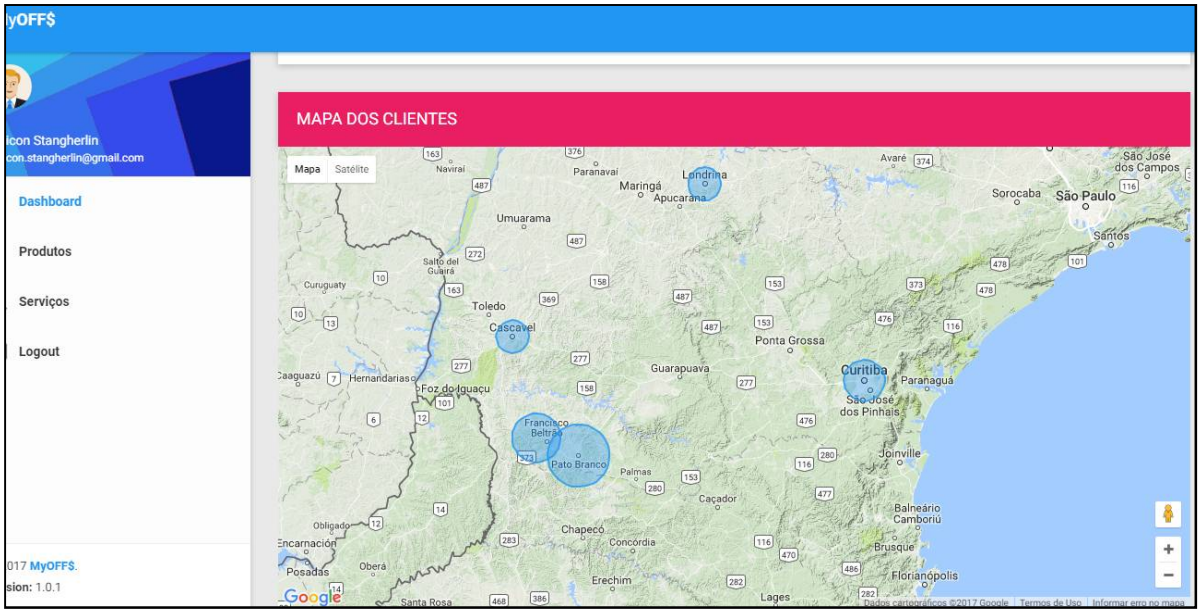


Figura 9 - Mapa *dos clientes (tela dashboard)*
Fonte: Autoria própria (2017).

Para inserção dos dados o usuário deverá acessar os menus de Produtos ou Serviços. Ambos seguem o mesmo padrão, utilizando um formato de *grid* com cartões. Como todo o desenvolvimento *front-end* foram respeitadas as normas da biblioteca Bootstrap, assim, todas as telas do sistema *web* são responsivas, permitindo que o leiautese ajuste em dispositivos com diferentes resoluções – Figura 9.

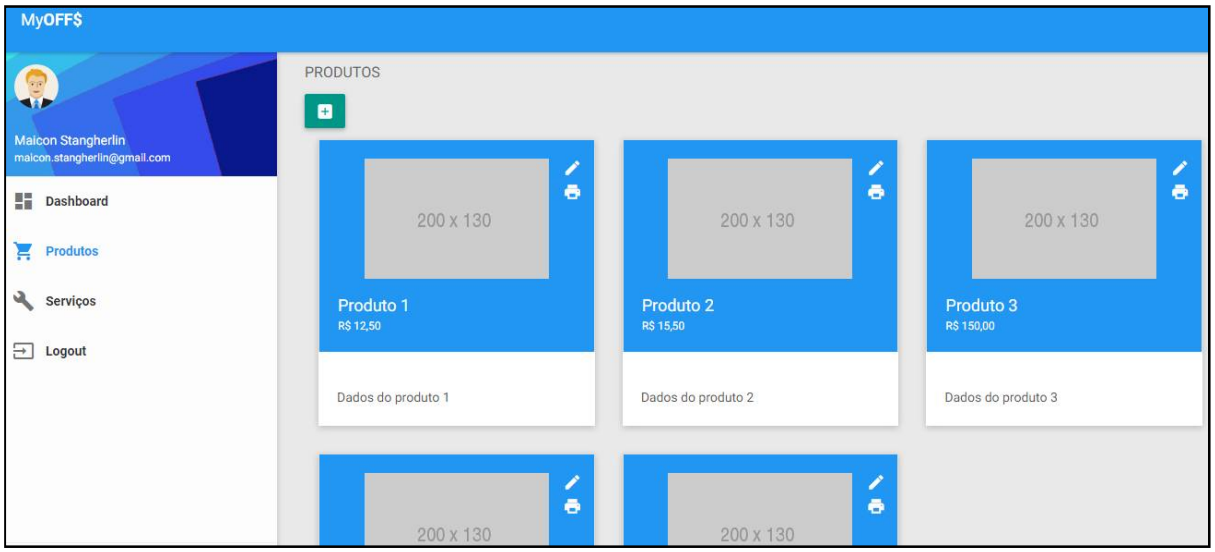


Figura 10 - Listagem de produtos
Fonte: Autoria própria (2017).

O cadastro dos itens é simples, deve-se informar alguns dados para que o sistema gere o *QR Code*. Em seguida o usuário da empresa terá a opção de imprimir o *QR Code*,

sendo que esse será exibido em tamanho A4 na tela padrão de impressão do navegador – Figura 10.

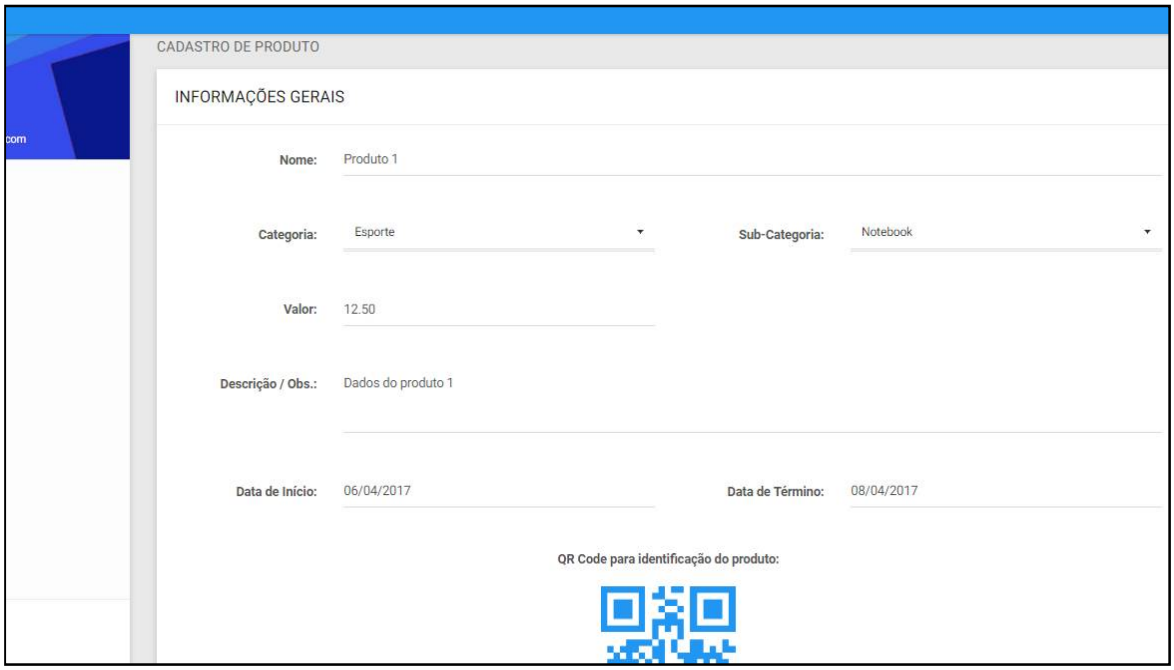


Figura 11 - Cadastro do produto
Fonte: Autoria própria (2017).

A impressão do *QR Code*em tamanho maior foi implementado a fim de melhorar a leitura do código por parte dos *smartphones* para produtos que ficam mais distante do cliente, como é o caso de vitrine de carros, por exemplo, assim, o código pode ser "estampado" no veículo que o cliente conseguirá lê-lo.

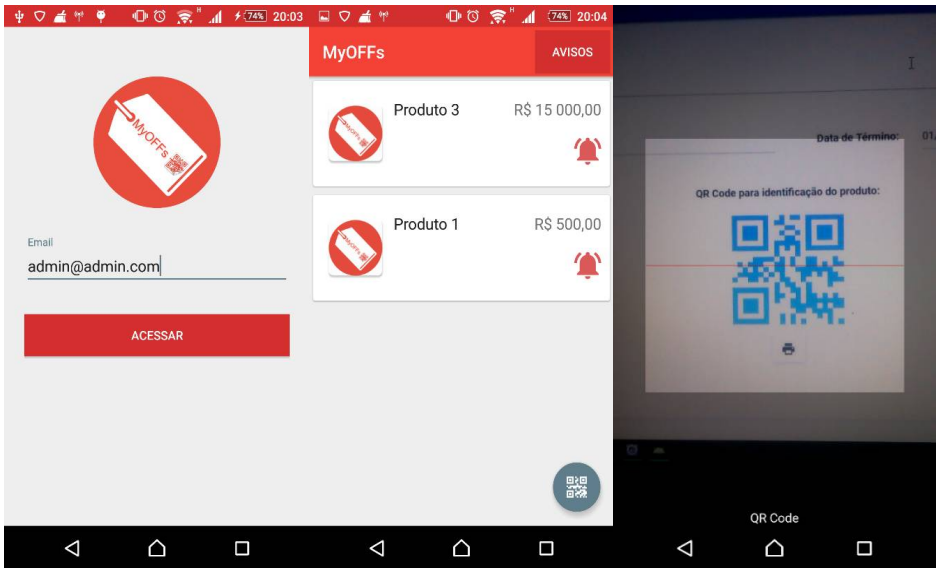
3.3.2MyOFFs - Aplicativo *mobile*

O aplicativo desenvolvido para o sistema operacional Android terá como função disponibilizar um meio para que os clientes das empresas cadastradas acompanhem seus produtos/serviços. A tela de *login* do aplicativo exige ao usuário apenas um *email* válido para acessá-lo, carregando logo em seguida a tela com a lista de produtos seguidos.

A leitura do *QR Code* fica por parte de um "botão flutuante" na tela principal do aplicativo, ao clicá-lo a câmera será acionada aguardando o usuário focar em uma imagem com o código do produto. Também é possível encerrar a câmera e voltar para à tela principal.

Outra função que pode ser visualizada na Figura 11 é a definição do "preço meta" para o recebimento de alertas dos itens – ícone de um sino, ou seja, o usuário pode definir

qual o preço que gostaria de pagar e quando o item chegar a esse preço o aplicativo emite uma notificação.

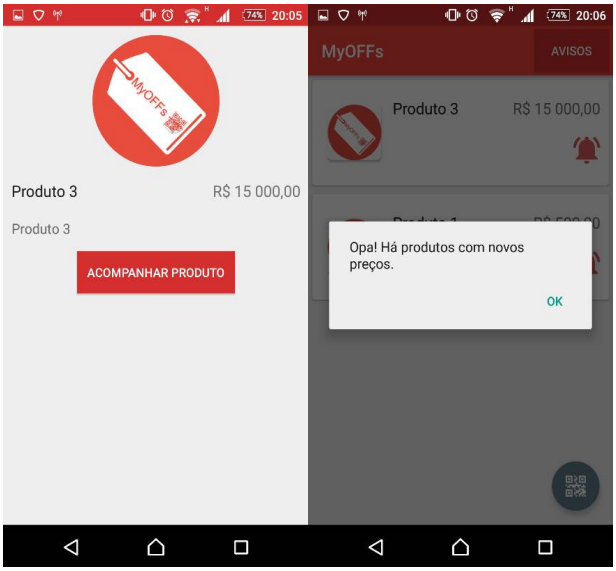


A **B** **C**

Figura 12 - Tela de *login*: listagem dos produtos/serviços e leitura do *QR Code* do *App mobite*
Fonte: Autoria própria (2017).

Inicialmente os avisos de alertas de preço serão verificados pelo sistema ao clicar no botão ‘Avisos’ na barra superior do aplicativo, futuramente este processo deverá ser substituído pelo método de *Push Notification*.

O protótipo do aplicativo também exibe ao usuário alguns detalhes do item após a leitura do código, conforme a primeira imagem da Figura 12. Nesta fase de prototipação optou-se por exibir os campos de imagem do produto, nome, valor e descrição.



A **B**

Figura 13 - Detalhes do item e exemplo da mensagem de alerta
Fonte: Autoria própria (2017).

3.4 IMPLEMENTAÇÃO DO SISTEMA

Nesta sessão serão apresentados alguns trechos de códigos considerados essenciais para a utilização do MyOFFs. Um dos itens que merece destaque no projeto é *QR Code*, sendo que, em ambas as plataformas (*mobile* e *web*) foi necessário utilizar de recursos de terceiros para realizar a leitura e a geração do mesmo.

Para a geração do *QR Code* na tela de cadastro dos produtos (*web*) foi utilizado uma classe JavaScript denominada ‘qrcode.js’. Ao instanciar essa classe pode-se configurar algumas opções de estilo do código a ser gerado como: cor de fundo, cor da linha, tamanho, qualidade do código, além do texto a ser codificado e local (*tagHyperText Markup Language* - HTML), a ser inserido o *QR Code* gerado.

Na Listagem 1 pode ser identificado a linha que recebe o ‘id’ do produto e faz com que esta variável seja utilizada para gerar o código bidimensional. Em testes aplicados durante o desenvolvimento notou-se uma leve “desvantagem” ao gerar o *QR Code* em cores diferentes de preto, pois, a leitura deste pelo *smartphone* é mais lenta.

```
function generateQrCode() {
    var id = $("#id").val();

    new QRCode(document.getElementById("qrcode"), {
        text : id,
        width : 150,
        height : 150,
        colorDark : "#2196f3",
        colorLight : "#ffffff",
        correctLevel : QRCode.CorrectLevel.H
    //levels zijn L, M, Q, H
    });

    new QRCode(document.getElementById("qrcodelarge"), {
        text : id,
        width : 800,
        height : 800,
        colorDark : "#2196f3",
        colorLight : "#ffffff",
        correctLevel : QRCode.CorrectLevel.H
    //levels zijn L, M, Q, H
    });
}
```

Listagem 1 - Geração de QR Code
Fonte: Autoria própria (2017).

Para as funcionalidades apresentadas no *dashboard* do sistema, uma das integrações utilizadas foi com o Google Maps APIs. Essa *Application Programming Interface*(API), interface de programação de aplicações, é responsável pela geração do mapa de densidade exibida na tela principal, onde vários parâmetros são disponibilizados permitindo a personalização do mapa.

Para poder fazer as requisições à API é necessário gerar um *token* para acesso. Este *token* é adicionado a *Uniform Resource Locator* (URL) da APIe validado de acordo com cada aplicação. Como o projeto está sendo inicialmente utilizado para fins acadêmicos não é preciso considerar o número de requisições diárias à API, mas, vale ressaltar que a utilização deste recurso poderá ser bloqueada após um determinado número de acessos, dependendo nestes casos da aquisição de planos pagos.

Na Listagem 2 é apresentado o trecho de código correspondente a função ‘InitMap()’ que tem como função instanciar a classe JavaScript para a geração do mapa. Para o tipo de mapa utilizado no projeto, além de algumas configurações básicas é necessário informar à API os valores referentes aos círculos que são exibidos sobre o mapa. No geral os principais atributos utilizados da APIforam: tipo de terreno, *zoom*, ponto central do mapa, lista de cidades, ponto central e raio de cada círculo baseado no número de clientes.

```
function initMap() {
  // Create the map.
  var map = new google.maps.Map(document.getElementById('map'), {
    zoom: 7,
    center: {lat: -25.390, lng: -51.462},
    mapTypeId: 'terrain'
  });

  // Construct the circle for each value in citymap.
  // Note: We scale the area of the circle based on the clientes.
  for (var city in citymap) {
    // Add the circle for this city to the map.
    var cityCircle = new google.maps.Circle({
      strokeColor: '#2196f3',
      strokeOpacity: 0.8,
      strokeWeight: 2,
      fillColor: '#2196f3',
      fillOpacity: 0.35,
      map: map,
      center: citymap[city].center,
      radius: Math.sqrt(citymap[city].clientes) * 1000
    });
  }
}
```

Listagem 2 - Mapa de densidade
Fonte: Autoria própria (2017).

No Android uma das principais funções do aplicativo é a leitura do *QR Code*. Para tal, foi utilizado a biblioteca *open-source* ZXing (lê-se ZebraCrossing), especializada em processamento de imagem de código de barras 1D e 2D (código de barras e *QR Code* por exemplo). Ao instanciar a classe 'IntentIntegrator' da biblioteca é disponibilizado algumas propriedades para configurar antes de iniciar a digitalização da imagem. Após ler o *QR Code* o aplicativo chama o método ouvinte 'onActivityResult' com o código do item.

O trecho de código responsável por chamar a câmera do *smartphone*pode ser visualizado na Listagem 3.

```
FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
fab.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        IntentIntegrator integrator = new IntentIntegrator(activity);
        integrator.setDesiredBarcodeFormats(IntentIntegrator.QR_CODE_TYPES);
        integrator.setPrompt("QR Code");
        integrator.setCameraId(0);
        integrator.setBeepEnabled(false);
        integrator.setBarcodeImageEnabled(false);
        integrator.initiateScan();
    }
});
```

Listagem 3 - Leitura do *QR Code*
Fonte: Autoria própria (2017).

Assim como o leitor de *QR Code* o *Web Service* também foi utilizado no projeto. Como não foi utilizado banco de dados no aplicativo (depende totalmente de conexão com a Internet), todo o material exibido ao usuário é buscado no servidor *web* via conexão REST, sendo que, nesse modelo as respostas do servidor costumam ser no formato JavaScript *Object Notation*(JSON).

Na Listagem 4 é exibido o método ‘doInBackgroud()’ correspondente a ‘AsyncTask’ ‘getProdutos()’. Essa função faz a chamada ao método *web* que retorna a lista dos produtos seguidos pelo usuário. Após receber essa lista em formato JSON, o Android percorre-a criando uma lista do tipo ‘Produto’ e envia para uma classe ‘Adapter’ que faz a organização das informações antes de gerar a lista visual.

GetProdutos | doInBackground()

```
@Override
protected Void doInBackground(Void... params) {
    try {
        ConexaoREST rest = new ConexaoREST();
        String jsonStr = rest.serviceCall_GET(URLs.getURL(URLs.URL_USER_PRODUCTS, email));
        Log.e(TAG, "Response from url: " + jsonStr);

        if (jsonStr != null) {
            try {
                JSONArray jsonArray = new JSONArray(jsonStr);
                JSONObject jsonObject;
                Produto produto;
                produtos = new ArrayList<Produto>();

                for (int i = 0; i < jsonArray.length(); i++) {
                    jsonObject = jsonArray.getJSONObject(i);

                    produto = new Produto();
                    produto.setId(jsonObject.getLong("id"));
                    produto.setNome(jsonObject.getString("nome"));
                    produto.setValor(jsonObject.getDouble("valor"));

                    produtos.add(produto);
                }

            } catch (final JSONException e) {
                Log.e(TAG, "Json parsing error: " + e.getMessage());
            }
        }
    } catch (Exception e) {
        pDialog.dismiss();
```

Listagem 4 - REST retornando JSON com lista de produtos
Fonte: Autoria própria (2017).

Há várias bibliotecas para Android que otimizam o manuseio em objetos JSON, como a biblioteca Retrofit, por exemplo, porém, no projeto do aplicativo optou-se por utilizar um formato mais manual (percorrendo um objeto JSON), devido a alguns contratempos em

relação ao estudo da biblioteca e ao prazo final do projeto. No entanto, recomenda-se a utilização deste recurso, uma vez que, ganha-se em produtividade e deixa o código mais “elegante”.

4 CONSIDERAÇÕES FINAIS

Este trabalho teve como objetivo demonstrar o desenvolvimento de um portal *web* para cadastro de promoções e ofertas integrando com um sistema *mobile* desenvolvido para a plataforma Android. Durante o projeto demonstrou-se a utilização de serviços como o Google Maps API, além de *frameworks* como Spring, JQuery, Hibernate, entre outros. Outra característica do projeto foi a utilização de códigos 2D (neste caso utilizando o *QR Code*), permitindo a identificação dos itens mais rapidamente.

Após a conclusão do sistema, MyOFFs, foi possível entender o processo que envolve a comunicação REST entre as aplicações *web* e *mobile*, a praticidade na utilização do Hibernate para a conexão com o banco de dados, assim como a usabilidade da biblioteca ZXing para leitura de *QR Code* nos *smartphones*.

Das dificuldades encontradas durante o desenvolvimento deste projeto pode-se destacar a dificuldade quanto às versões do *Software development kit* (SDK) do Android, sendo necessárias várias tentativas em diferentes versões para que a biblioteca ZXing fosse executada corretamente, bem como, componentes do *Material Designer*. No lado *web* a maior dificuldade foi relacionada ao leiaute das páginas responsivas utilizando a biblioteca Bootstrap e alguns componentes JavaScript.

O resultado deste trabalho possibilita futuras melhorias como a utilização de notificações *Push*, utilização de *framework* para requisições REST na plataforma Android, utilização do Spring Security no portal *web*, melhorias de interface e usabilidade principalmente no aplicativo *mobile*, além da adição de novos recursos para aprimorar a experiência do usuário.

Por fim, pode-se concluir que os objetivos propostos foram alcançados por meio do consumo do *QR Code* gerado a partir de um portal *web*, assimo cliente passa a acompanhar as alterações de preço do produto, permitindo que ele cadastre um preço mínimo para o recebimento de alertas. Os índices exibidos no *dashboard* do portal também possibilita ao proprietário da empresa a análise de algumas informações que podem ser essenciais na tomada de decisões.

REFERÊNCIAS

ALECRIM, E. **Banco de dados MySQL e PostgreSQL**. 2008. Disponível em Info Wester: <https://www.infowester.com/postgremysql.php />. Disponível em: 22 jun. 2017.

ANISZCZYK, C., GALLARDO, D. **Developer works.** 2012. Disponível em: <https://www.ibm.com/developerworks/br/library/os-eclipse-platform/ />. Acesso em: 09 mar. 2017.

CAELUM. **Caelum ensino e inovação**. 2014. Disponível em: <http://www.caelum.com.br/apostila-vraptor-hibernate/persistindo-os-dados-com-o-hibernate/#4-2-sobre-o-hibernate />. Acesso em: 09 mar. 2017.

CAPELAS, B. **Link**. Disponível em: <http://link.estadao.com.br/noticias/gadget,brasilchegaa168milhoesdesmartphonesemuso />. Acesso em: 14 abr. 2016

CONSULTOR, P. **Bizagi: o software para organizar processos**. 2011. Disponível em: <https://pauloconsultor.wordpress.com/2011/11/27/bizagi-o-software-para-organizar-processos>. Acesso em: 27 mai. 2017.

Fabrica De Aplicativos. **O que é a notificaçãopush?** 2015. Disponível em: <http://fabricadeaplicativos.com.br/fabrica/o-que-e-e-como-funciona-a-notificacao-push/ />. Acesso em: 6 mar. 2017.

FARIA, F. B., LIMA, P. d., DIAS, L. G., SILVA, A. A., COSTA, M. P., BITTAR, T. J. **Evolução e principais características do IDE Eclipse**, 2010.

FISBHEN, B. **Porque aplicativo é a mídia que mais cresce**. 2017 Disponível em: <http://blog.panrotas.com.br/traveltech/index.php/2016/11/24/porque-aplicativo-e-a-midia-que-mais-cresce/ />. Acesso em: 24 nov. 2017.

MARCHAND, R. *BPM* – **Abordagem conceitual**. 2009. Disponível em: <http://www.linhadecodigo.com.br/artigo/2502/bpm-abordagem-conceitual.aspx />. Acesso em: 02 mar 2017,

NASCIMENTO, T. **Desenvolvendo com Bootstrap 3: um framework front-end que vale a pena!** 2013. Disponível em: <http://thiagonasc.com/desenvolvimento-web/desenvolvendo-com-bootstrap-3-um-framework-front-end-que-vale-a-pena />. Acesso em: 20 mar. 2017.

PALIARI, M. **Como funciona um webservice rest**. 2012. Disponível em: <http://www.matera.com/br/2012/10/22/como-funciona-um-webservice-rest/ />. Acesso em: 20 jun. 2017.

PEREIRA, O. C. **Por que usar BPM?** 2008. Disponível em: <http://www.informazione4.com.br/cms/opencms/desafio21/artigos/gestao/organizando/0017.html />. Acesso em: 20 jun. 2017.

PLATAS, E. **Introdução a Bootstrap Framework**. 2013. Disponível em: <http://www.ericplatas.com.br/artigos/introducao-bootstrap-framework/ />. Acesso em: 20 mar. 2017

POSTGRESQL. **Documentação do PostgreSQL 8.2.0**. Disponível em PostgreSQL: <http://pgdocptbr.sourceforge.net/pg82/intro-whatis.html />. Acesso em: 22 jun. 2017.

REDAÇÃO OFICINA. **PostgreSQL o que é?** 2008. Disponível em: <https://www.oficinadanet.com.br/artigo/746/postgresql_o_que_e />. Acesso em: 12 jul. 2017.

RICARDO, J.**Introdução a JQuery**. Disponível em: <http://www.devmedia.com.br/introducao-a-jquery/27299 />. Acesso em: 13 jun. 2017.

SASSO, E. **Gerenciando projetos com Maven**. Disponível em: <http://www.devmedia.com.br/gerenciando-projetos-com-maven/10823 />. Acesso em: 13 jun. 2017.

SATO, D., FERREIRA, J. E. **Academia.edu**. 207. Disponível em : <http://www.academia.edu/8911744/Dicas_e_Estrat%C3%A9gias_Sincroniza%C3%A7%C3%A3o_Estrat%C3%A9gia_Vantagens_Desvantagens_Triggers />. Acesso em: 09 mar. 2017.

SILVA, M. S. **Introdução à jQuery**. Disponível em: <http://www.linhadecodigo.com.br/artigo/2068/introducao-a-jquery.aspx />. Acesso em: 13 jun. 2017.

SILVESTRE, M. P. Desenvolvimento de um sistema de apoio ao tratamento de pascientes com desvios fonológicos para plataforma Android. **Desenvolvimento de um sistema de apoio ao tratamento de pascientes com desvios fonológicos para plataforma Android.** 2015,.Disponível em: <http://aberto.univem.edu.br/bitstream/handle/11077/1393/MONOGRAFIA%20MATHEUS%20POLTRONIERI%20SILVESTRE.pdf?sequence=1 />. Acesso em: 05 jun. 2017.

THIENGO, V. **Integrando o leitor de QRCode ZXing no Android.** 2014.Disponível em <https://www.thiengo.com.br/integrando-o-leitor-de-qrcode-zxing-no-android />. Acesso em: 17 ago. 2017.

WAGNER, J. **The increasing importance of APIs in web development**.2011. Disponível em: https://code.tutsplus.com/articles/the-increasing-importance-of-apis-in-web-development--net-22368 /. Acesso em: 08 jun. 2017.