UNIVERSIDADE DE SÃO PAULO

ESCOLA DE ENGENHARIA DE SÃO CARLOS

DEPARTAMENTO DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO

Aplicativo android e website interativos para busca de menores preços de produtos com código de barras

Autor: Rafael Alexandre Freiberger Hellmann

Orientador: Prof. Dr. Evandro Luis Linhari Rodrigues

São Carlos

2016

Rafael Alexandre Freiberger Hellmann

Aplicativo android e website interativos para busca de menores preços de produtos com código de barras

Trabalho de Conclusão de Curso apresentado
à Escola de Engenharia de São Carlos, da
Universidade de São Paulo

Curso de Engenharia Elétrica

ORIENTADOR: Prof. Dr. Evandro Luis Linhari Rodrigues

São Carlos

2016

Hellmann, Rafael Alexandre Freiberger
H476a Aplicativo android e website interativos para busca de menores preços de produtos com código de barras / Rafael Alexandre Freiberger Hellmann; orientador Evandro Luis Linhari Rodrigues. São Carlos, 2016.

Monografia (Graduação em Engenharia Elétrica com ênfase em Eletrônica) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2016.

1. Android. 2. Aplicação Web. 3. Código de barras. 4. Base de dados. 5. Big Data. 6. Cloud Computing. I. Título.

FOLHA DE APROVAÇÃO

Nome: Rafael Alexandre Freiberger Hellmann

Título: "Aplicativo android e website interativos para busca de menores preços de produtos com código de barras"

Trabalho de Conclusão de Curso defendido e aprovado em 29/11/2016,

com NOTA 9,7 (NOVE, SETE), pela Comissão Julgadora:

Prof. Associado Evandro Luis Linhari Rodrigues - Orientador - SEL/EESC/USP

Prof. Associado Ivan Nunes da Silva - SEL/EESC/USP

Mestre André Luís Martins - (Doutorando - SEL/EESC/USP)

Coordenador da CoC-Engenharia Elétrica - EESC/USP:
Prof. Associado José Carlos de Melo Vieira Júnior

Resumo

De redes sociais, fóruns e avaliações de usuários nota-se um crescimento no número de sistemas nos quais usuários acessam informações disponibilizadas ao público apenas por outros usuários. Assim surgiu a ideia de um sistema para encontrar a melhor oportunidade de compra de produtos no varejo por meio do código de barras e entradas dos próprios compradores, pois assim os dados se mantêm atualizados e permitem a geração de uma série de análises de tendências. A implementação foi dividida em um *website* e um aplicativo Android com a mesma função: adicionar, atualizar ou buscar lojas e produtos em uma base de dados. Tanto o aplicativo como *website* desenvolvidos se provaram funcionais, e a base de dados é escalonável.

Palavras chave: 1.Android 2.Aplicação Web 3.Código de Barras 4.Base de dados 5.Big Data 6. Cloud Computing

Abstract

From social networks, forums and user reviews, it is possible to notice a tendency in the newest systems where users access information delivered only by other users. From this context came the idea of a system to find the lowest price of retail products using its barcode and entries from other users, because this way the database would stay updated and allow many different statistical analysis. To acheive this two susbsystems werde developed, a website and an android application, with the same functionality: add, update or search for products and stores in a database. Both the application and the website proved to be functional, and the database is scalable.

Keywords: 1.Android 1.Web Application 3.Barcode 4.Database 5.Big Data 6. Cloud Computing

Lista de Figuras

Siglas

CDB	Código de barras
IDE	*Integrated development environment,* - Ambiente integrado de desenvolvimento
GUI	*Guided User Interace*
UI	*User Interace*
RAM	A memória RAM de um dispositivo, *Random Access Memory*
PC	*Personal Computer*, computador pessoal
ioT	*Internet of Things*, a internet das coisas
Web	Também como *Wold Wide Web*, a Internet
Browser	Navegador Web
GPS	*Global Positioning System*
BD	Banco de dados
DBMS	*Database Management System*
SQL	*Structures Query Language*
HTTP	*Hypertext Transfer Protocol*
HTML	*HyperText Markup Language*

Sumário

Capítulo 1

Introdução

Um dos aspectos mais importantes quando se compara alternativas de compra para produtos similares é o preço. Isso ocorre porque no contexto dos mercados onde existe livre concorrência há geralmente muitas opções que satisfazem o desejo do comprador de maneira semelhante. Assim, o fator de decisão dentro desse subconjunto das opções de compra é muitas vezes o gasto de tempo e dinheiro para a sua aquisição.

Entretanto, comparar essas alternativas não é uma tarefa tão simples pois normalmente não há uma fonte de informação centralizada contendo as opções nos diferentes estabelecimentos. Esse tipo de pesquisa de mercado é tipicamente realizada pelo consumidor, e as ferramentas usadas na maioria das vezes não são as mais avançadas.

Dentre os sistemas existentes destaca-se o "Buscapé", o maior comparador de preços do brasil [14]. O site foca, contudo, em produtos de maior valor e não possui um recurso para reconhecimento de código de barras. Há também o "Meu carrinho", que além de comparar preços possibilita a criação de uma lista de compras, mas também não é um sistema no qual os usuários entram com os dados. Por último pode-se citar o aplicativo "Zoom"que faz o reconhecimento usando CDB, mas como os demais não aceita entradas dos usuários.

Nesse sentido a proposta desse projeto é a criação de um sistema para melhoria da eficiência nas compras de produtos de baixo valor agregado. A abordagem é semelhante a das alternativas para os demais mercados: cria-se uma base de dados contendo alternativas para se adquirir um produto, com seus preços e localizações, e faz-se com que esta seja acessível de maneira intuitiva.

Para cumprir com esses objetivos é necessário que a base de dados esteja sempre atualizada, o que é uma dificuldade grande já que os preços flutuam diariamente sem referenciar a um controle central disponível ao público. Como é inviável armazenar toda a informação em

todos os aparelhos clientes ao mesmo tempo, é necessário uma base de dados central.

Assim surge a idéia de um aplicativo para celular apoiado por um servidor, como na figura 1.1. A mobilidade do aparelho permite aos usuários alertarem uns aos outros de boas oportunidades rapidamente atualizando a base de dados, enquanto o servidor se encarrega de armazenar e buscar essas grandes quantidades de dados.

Figura 1.1: Visualização do funcionamento do sistema

1.1 Motivação

Uma boa ideia quando se fala em inovação é tentar entender as tendências atuais das necessidades e vontades das pessoas com o intuito de prever o que há por vir e, com essa informação, tomar medidas cabíveis para adiantar-se em relação aos concorrentes. Essa é uma abordagem que por muitas vezes se mostrou eficiente, e faz o sucesso da inovação em certo grau independente das condições iniciais de seu surgimento. Uma outra abordagem, por exemplo, é a criação de uma demanda juntamente com a inovação, mas para isso deve-se apelar a outros recursos que não são intrínsecos do produto, como é o caso de qualidade, preço, abrangência, etc.

No contexto da segunda década do século XXI e no universo da eletrônica, há alguns conceitos que se destacam pela velocidade com que cativam o público. Um deles é a chamada Internet das Coisas (chamada também de ioT, vindo do inglês *Internet of Things*), que prevê o aumento da variedade de aparelhos eletrônicos que se conectam à Internet. [15] O uso dessa rede toma um outro rumo, não mais somente explorando as possibilidades de transporte rápido de informação entre pessoas ou máquinas, mas sim apoiando-se fortemente

na capilaridade dela para conectar equipamentos que até então não se beneficiavam dessa estrutura.

Uma outra tendência no uso da eletrônica e computação é a aglutinação e interpretação de grandes quantidades de dados para determinação de tendências ou padrões. A denominação dessa moda é o termo vindo do inglês "Big Data", na qual tipicamente muitas fontes diferentes abastecem um ponto central com grande quantidade de dados [16]. São então utilizados algoritmos computacionais para gerar como resultado padrões na população amostrada.

Esses dois conceitos, muito atrelados, ainda se relacionam com um terceiro: contribuição comunitária. Essa idéia se baseia no conceito de que as melhorias ou correções necessárias do software são melhor percebidas pelo usuário [17] . Com esse propósito se dispõe a ele informações e ferramentas para fazer esse julgamento das inovações: se é útil ou inútil, verdadeiro ou falso, bom ou ruim.

Isso motiva a criação de sistemas como o proposto nesse trabalho, que tem sua funcionalidade muito mais atrelada às interações dos usuários do que com a estrutura proposta da solução.

1.2 Objetivo

Desenvolver um sistema acessível por meio de um website e um aplicativo android para busca dos menores preços de produtos com código de barras, possibilitando ao usuário buscar por produtos e lojas, bem como interagir com a base de dados adicionando ou atualizando suas entradas.

1.3 Justificativa

A escolha do segmento de produtos de baixo valor agregado se deu pelo fato das opções disponíveis para busca dos preços de produtos não cumprirem de maneira satisfatória esse ramo de atuação, sendo geralmente focadas em produtos mais caros ou não permitirem ao usuário enviar informações. A interação comunitária mantém a base de dados atualizada e contendo produtos que efetivamente são comprados pelo público.

O uso de código de barras como identificador único desses produtos ocorreu naturalmente pois já é usado pelas lojas com sucesso.

Capítulo 2

Embasamento Teórico

Este capítulo tem por objetivo sumarizar os conceitos necessários para o entendimento do sistema desenvolvido.

Algumas vezes os termos usados na língua portuguesa para se referir as tecnologias relacionadas à internet são levemente ambíguos. Nesse trabalho, a convenção utilizada foi: o termo "servidor"se refere à máquina ou conjunto de máquinas responsáveis pelos serviços de internet. Engloba, portanto, tanto hardware e software. Quando os termos "servidor Web"são usados, é refenciado somente o software específico para responder a requisições da Internet. Como explicado posteriormente, as expressões "sistema de banco de dados"e "banco de dados"são usadas de forma intercambiável a não ser quando especificado.

2.1 Internet como estrutura

O sistema proposto faz parte de uma tendência moderna de se usar a internet como principal estrutura no intuito de criar uma aplicação para melhorias para a vida cotidiana. Com conexões e interfaces cada vez mais rápidas, o que limita o tempo de resposta e consequentemente a qualidade desse tipo sistema são os servidores. O desempenho desses, por sua vez, depende de uma combinação de fatores que envolvem diversos níveis diferentes de software, suas interações (mostrados na figura 2.1) e sua capacidade de se adequar às requisições dos clientes. [1].

No escopo desse projeto, otimizar o desempenho do servidor se resume a escolher uma plataforma para suporte da aplicação adequada, pois uma reconfiguração da estrutura é uma tarefa extremamente complexa [18].

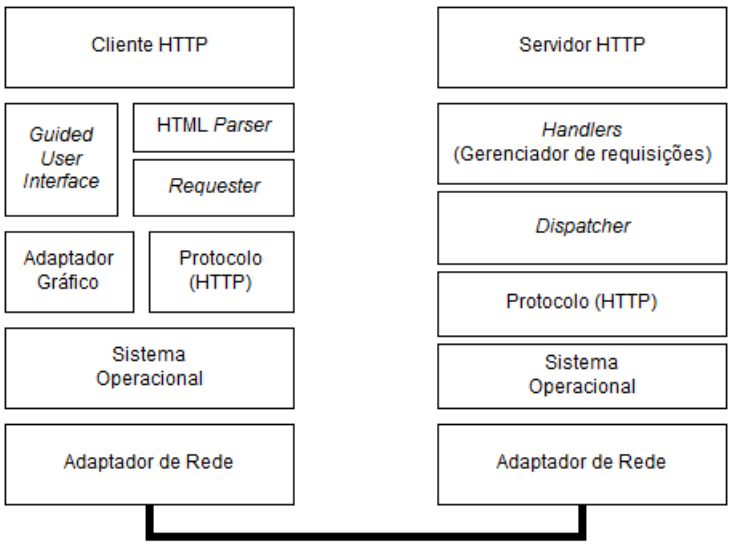


Figura 2.1: Exemplo mostrando diferentes níveis da comunicação com um servidor (adaptado de [1])

2.2 *Clusters*

Devido a formação desse limitante no lado do servidor, uma prática nova e comum é a utilização de "clusters". Esse conjunto de computadores interconectados por meio de de uma rede de alta velocidade realizando as tarefas de forma coordenada são relativamente simples de implementar, e oferecem várias vantagens como altas disponibilidade e performance.

Contudo, o que se torna um desafio é gerenciar esse recurso computacional de maneira eficiente no contexto extremamente dinâmico da internet. Isso é feito usando sistemas "DARC"(*Dynamic Architecture for Recognition of Web Server Clusters*) [2] . Esses sistemas funcionam usando três principais componentes: servidores Web, *switches* de classe e *gateways* do cluster como mostrado na figura 2.2.

O *switch* de classe é quem recebe as requisições dos clientes. Ele as repassa para os clusters de acordo com regras pré-estabelecidas de prioridade e atuais níveis de carregamento do sistema. Esse, por sua vez, envia a requisição ao servidor que esteja mais apto a recebê-la. Dessa forma os recursos são alocados de maneira muito eficiente e dinâmica.

Considerando que a aplicação proposta tem como objetivo criar uma estrutura capaz de se expandir rápida e dinâmicamente de acordo com a demanda, esse tipo de servidor é muito apropriado. Mais uma vez a sua implementação de maneira personalizada foge do âmbito para o projeto proposto, mas a escolha de uma plataforma capaz dessas funcionalidades é muito importante.

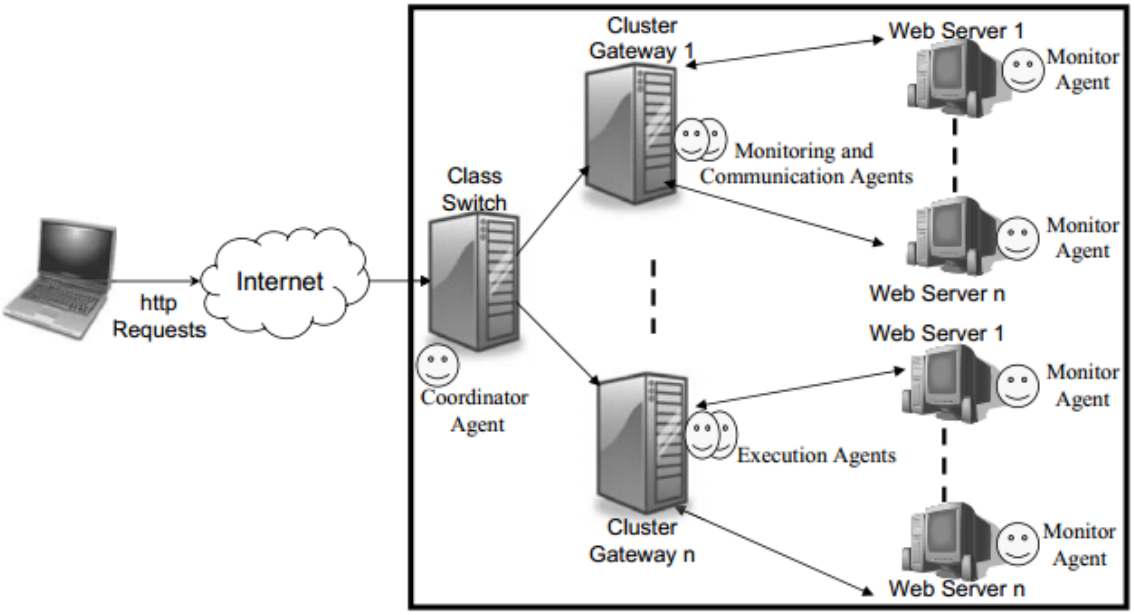


Figura 2.2: Estrutura de um "cluster"de Web Servers [2]

2.3 Sistema de banco de dados

É quase impossível se pensar em uma grande organização que não utilize um banco (também chamado de "base") de dados computadorizada para complementar o controle de seus registros, planejamentos ou recursos. No passado o controle era feito por meio de documentos físicos e arquivos, mas as vantagens de se transpor esses documentos para um sistema computadorizado são claras e os chamados DBMS (*Database Management Systems*) vêm ganhando cada vez mais importância.

Tecnicamente, se diferenciam três conceitos importantes quando se faz referência a sistemas de base de dados, como exposto na figura 2.3. Um sistema de banco de dados é, na verdade, composto por dois elementos:

- Banco de dados: é o conjunto de unidades de informação, os chamados dados, que são importantes para o controle ou execução da tarefa ou atividade.
- Sistema de gerenciamento de banco de dados (DBMS): a associação de programas para o acesso, controle e modificação da base de dados (Ex. MySQL).

Contudo, essa divisão não é sempre considerada e muitas vezes apenas uma das expressões, "base de dados"ou "DBMS", é usada para se referir ao sistema todo, o sistema de banco de dados. Essa convenção apesar de errônea é muito prática e se entende facilmente o sen-

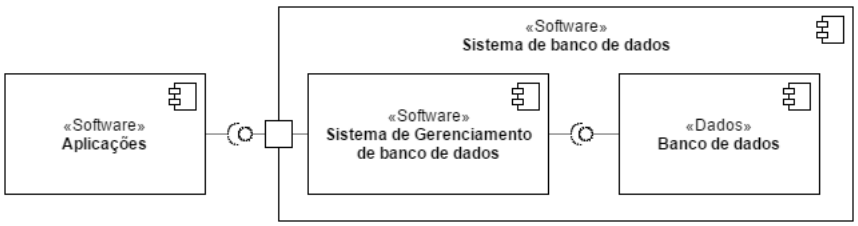


Figura 2.3: Sistema de banco de dados

tido usado pelo contexto. A parte dessa introdução, a convenção também será utilizada nesse trabalho.

2.3.1 Tarefas de um Sistema de Banco de Dados

Um sistema de gestão de banco de dados deve cumprir algumas tarefas específicas, e existem algumas características e operações que são desejáveis. Essas são listadas a seguir: [19]

- Integração dos dados: É possível armazenar dados nos mais diversos formatos, de números, textos até imagens e sons. Uma tarefa importante de um banco de dados é suportar o tipo de dado desejado e fazer com que as operações de busca, exclusão ou edição destes sejam possíveis e independentes do formato em que os dados são salvos, ou seja, permitir que os diferentes dados sejam gerenciados de uma maneira padrão.

- Permitir operações: Tarefas como busca, modificação, etc. devem estar acessíveis em um nível elevado de abstração. Isso quer dizer que as informações passadas ao sistema não precisam ser muito detalhadas, mas a execução deve ocorrer utilizando-se algoritmos apropriados visando fatores de qualidade como tempo e segurança.

- Descrição apropriada: Um banco de dados deve conter as informações apropriadas para a aplicação desejada, bem como conectá-las de maneira eficiente. Para que fique claro a maneira com que os dados reais são representados no modelo, pode ser usado o chamado "dicionário de dados"contendo essas descrições.

- Realizar camadas: Nem todos os usuários devem ter acesso a todas as informações da base de dados. Por exemplo, um usuário deve ter acesso a seus dados pessoais mas não aos de outros usuários. Quando é realizada uma busca, é importante que só sejam exibidos dados apropriados a esse usuário (ou tipo de usuário). Essa diferenciação recebe o nome de "camadas".

- Consistência: É muito importante que os dados em um banco de dados não sejam contraditórios ou ambíguos. Por exemplo, numa lista de produtos e seus códigos de barras não deve existir dois produtos com o mesmo código. É essencial para um DBMS manter a consistênca do banco de dados mesmo depois de realizadas diversas alterações.

- Definir "transações": Há tarefas a serem executadas no banco de dados que exigem mais de uma etapa, mas que só devem ser executadas se todas as etapas forem possíveis e aceitáveis. Por exemplo, numa transferência bancária é necessário que antes de se retirar dinheiro de uma conta se verifique a existência da conta destino, evitando um erro que poderia fazer a quantia "desaparecer". É necessário, portanto, que as etapas sejam verificadas antes da execução, bem como realizá-las de maneira que seus dados possam ser recuperados caso um erro inesperado ocorra.

- Sincronizar alterações: Em um sistema com diversos usuários, fica claro que comandos contraditórios para alteração do mesmo dado podem ser dados ao mesmo tempo. É tarefa do DBMS gerenciar essa sincronização, considerando a devida prioridade.

- Recuperação de dados: No caso de haver um desligamento inesperado do sistema, o DBMS deve ter um mecanismo para recuperação dos dados para o último estado estável do sistema para manter a consistência de seus dados.

2.3.2 Estrutura de um Banco de dados: Abstração em camadas

Quando se deseja analisar um sistema de banco de dados a respeito de sua estrutura, é possível entendê-lo como divido em três camadas: física, lógica e de apresentação (figura 2.4) [20]. Cada uma delas têm diferentes funções e interage com a camada seguinte através de uma interface determinada, portanto o nível de abstração é maior quanto superior for a camada.

A primeira camada, a física, representa a forma com que os dados são gravados. Geralmente é usado um disco rígido ou sistema semelhante, sendo a forma com que os dados são salvos e indexados incluídos nessa representação. Essa camada não é acessada pelo usuário e nem pelo programador do do banco de dados, somente pelos chamados "administradores do banco de dados". O principal desafio é organizar as informações de maneira com que possam ser localizadas rapidamente quando requisitado.

A camada intermediária, chamada de Lógica, é onde se determina a estrutura com que os dados são salvos. Aqui são definidos os Objetos, suas propriedades e relações que definirão o

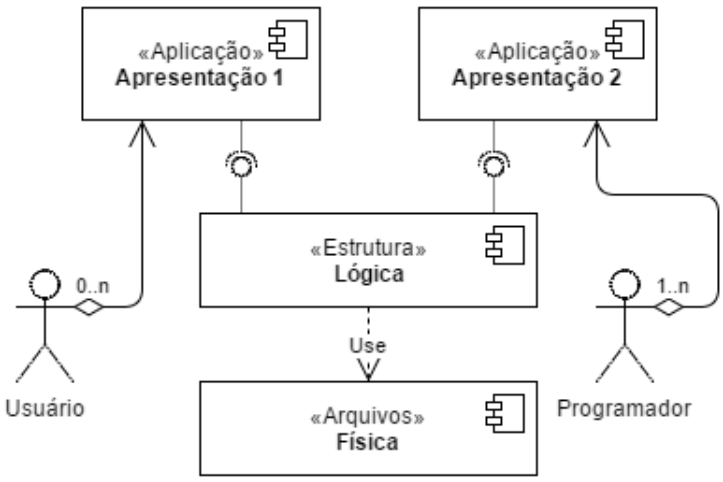


Figura 2.4: Banco de dados

banco de dados. Pode ser pensado como a criação das tabelas num banco de dados relacional: se define quais tabelas, colunas e chaves haverão.

No terceiro nível se especifica quais acessos cada usuário (ou grupo de usuários) terá, ou seja, as operações que lhe cabem e os dados que lhe são relevantes para o seu determinado uso do banco de dados.

2.3.3 Tipos de banco de dados

Há diferentes maneiras com que os dados podem ser armazenados em um banco de dados. A forma com que estes são interconectados nos permite separá-los em diferentes tipos de bancos de dados, sendo os principais: hierárquico, em rede, relacional, orientado a objeto e objeto-relacional [20] [19] [21]

- Hierárquico

  Organiza os dados em uma estrutura em forma de árvore, em que diferentes níveis se relacionam (são apontados) em uma estrutura denominada "pai-filho". Essa relação conecta um elemento pai a "n"elementos filhos (1:N), agrupando os filhos de acordo com uma propriedade comum expressa no elemento pai como exemplificado na figura 2.5. Já foi muito popular e é util quando se pretende armazenar dados de uma estrutura que já tem uma hierarquia definida, como por exemplo um departamento de uma empresa como mostra a figura. Uma grande vantagem desse modelo é que a busca pode ocorrer

apenas num subconjunto do banco de dados. Por exemplo, para se buscar um funcionário de uma empresa os dados dos fornecedores não seriam revistados, pois encontram-se em um outro ramo da árvore.

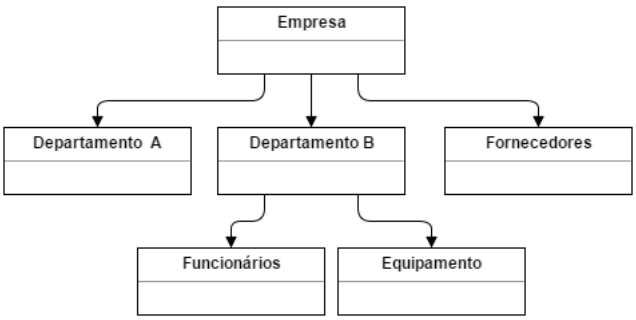


Figura 2.5: Modelo hierárquico

- Modelo em Rede

  É uma versão mais genérica do modelo hierárquico. Nele, um elemento filho pode ser apontado por mais de um pai (relação M:N, vide figura 2.6). Isso faz com que o acesso a informação ocorra de maneira rápida, pois as múltiplas conexões fazem com que seja fácil de navegar entre as relações mais diversas na base de dados. Contudo, um grande ponto negativo dessa forma de armazenar informação é que se a estrutura precisa ser alterada de alguma forma, é necessário uma revisão de toda a base de dados devido às múltiplas conexões.

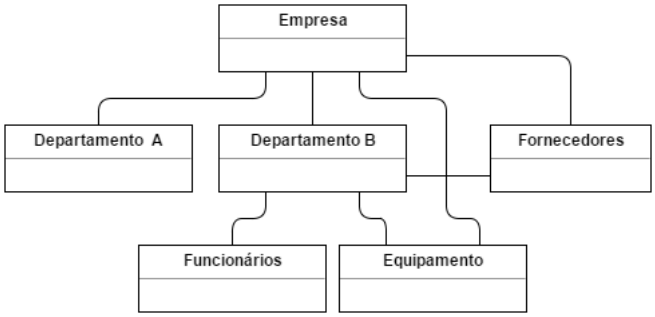


Figura 2.6: Modelo em rede

- Relacional

Definitivamente o tipo mais difundido de banco de dados, o banco relacional é muito diferente dos modelos hierárquicos e em rede, pois foi projetado sem se considerar a estruturação dos dados para determinar as relações. Ele se baseia na teoria matemática dos conjuntos, considerando tabelas como elementos bidimensionais (linhas e colunas) e apenas as relações entre as diferentes dimensões das diferentes tabelas (vide figura 2.7). Dessa forma não há informações redundantes, mas uma série de regras específicas precisa ser seguida para que as relações entre tabelas façam sentido. Por exemplo, cada registro (linha) de uma tabela precisa ser único e conter apenas um elemento. Uma grande vantagem é que uma alteração em uma tabela não afeta a maneira com que ela se relaciona com as outras, permitindo que a modificação possa ser feita com poucas considerações.

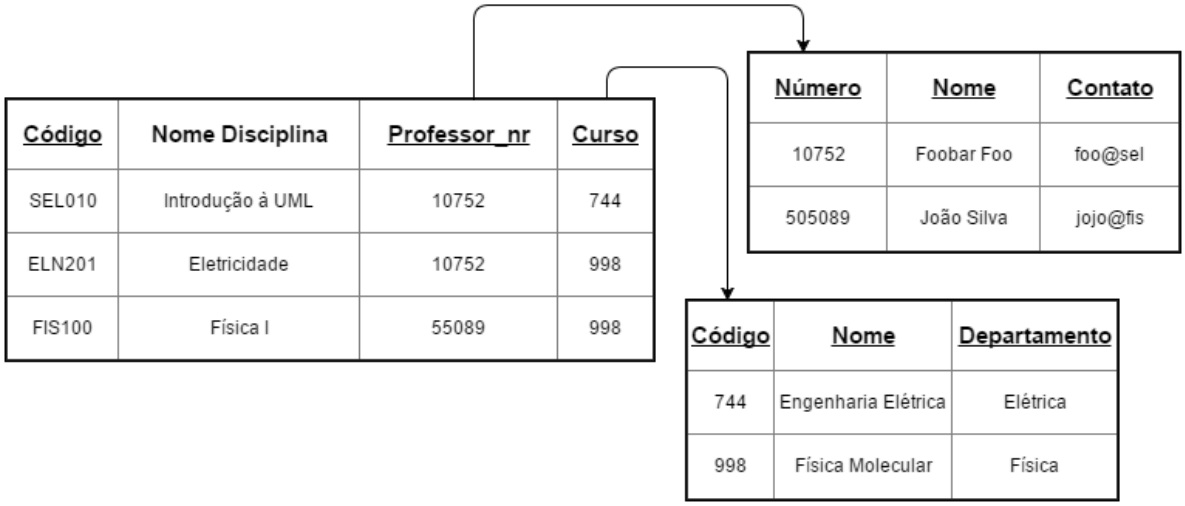


Figura 2.7: Modelo Relacional

- Orientado a Objeto

  É um tipo relativamente novo e uma evolução do modelo em rede. As informações no modelo Orientado a Objeto são salvas na forma de estruturas de objetos complexas. O objetivo é trazer as vantagens da lógica da orientação a objeto como a criação de classes, atributos, métodos, etc. para a base de dados.

- Objeto-Relacional

  Há ainda um híbrido entre o modelo orientado a objetos e o modelo relacional. Claramente o objetivo é unir as vantagens dos dois sitemas em um, através da representação de objetos na forma de tabelas.

2.4 Programação Web

Desde a criação da internet, surgiram diversas linguagens muito diferentes para programação de aplicações web. Isso ocorre porque existem tarefas distintas que devem ser cumpridas pela aplicação, e cada uma delas requer uma lógica ou organização diferente. Linguagens apropriadas para gerenciamento de um banco de dados, por exemplo, não descrevem a interface com o usuário ou o funcionamento da aplicação. A seguir são detalhadas algumas linguagens usadas para web.

2.4.1 PHP

Criada em 1995 por Rasmus Lerdorf released PHP, se mostrou sendo a linguagem dinâmica mais usada em uma pesquisa em 2007 [22]. PHP originalmente significa "página inicial pessoal"(*personal home page*), é *open-source*, multi plataforma, orientada a objeto e executada no lado do servidor.

Assim como JS, PHP pode ser inserido no código HTML da página e torná-la dinâmica. PHP ainda conta com os "PHP-PDOs"(*PHP Data Objects*) oferecem uma interface para base de dados relacionais permitindo à página acessar um banco de dados desse tipo. [22]

2.4.2 Javascript

Javascript, geralmente abreviada para JS, é uma linguagem interpretada e orientada a objeto. É muito conhecida por gerar conteúdo dinâmico em páginas web, sendo incorporada ao código HTML e respondendo à rolagem da tela e posição do mouse, por exemplo. Isso se dá pelo fato da linguagem ser interpretada no lado do cliente.

Ao contrário do que pode parecer, não é diretamente relacionada com Java. A semelhança se limita ao fato de JS ter sua sintaxe intencionalmente parecida com Java, para que seja fácil aprendê-la quando se está familiarizado com a segunda, mas tem também fortes influências

da sintaxe C++. [23] [22]

2.4.3 Python

Python foir criado em 1990 por Guido van Rossum. É uma linguagem *open-source*, interpretada e orientada a objeto. Seu nome é cursiosamente derivado de um show de humor britânico, *Monty Python's Flying Circus*, porque o criador da linguagem é um grande fã.

A linguagem é conhecida por ser clara, ter sintaxe simples, ser de fácil entendimento e portável entre sistemas operacionais. É muito difundida em aplicações web, mas também é usada para gerenciar bases de dados, criar *GUIs*, aplicações científicas e jogos. [22]

2.4.4 Ruby

Criada em 1995 por Yukihiro Matsumoto, é *open-source*, interpretada e totalmente orientada a objeto. É conhecida por ser limpa, concisa, consistente, bem estruturada, extensível e portável entre sistemas operacionais.

É muito difundida para programação do lado do servidor, devido ao *framework* "Ruby on Rails". Com esse recurso a liguagem disponibiliza estruturas para base de dados, páginas Web e serviços Web.

2.4.5 SQL

SQL, que significa *Structured Query Language*, era originalmente chamada de *SEQUEL* significando *Structured English Query Language* quando a empresa IBM desenvolveu um DBMS chamado "System R"na década de 70 e criou a linguagem. Logo ela foi renomeada para SQL, e a partir da decada de 80 já se tornou o termo mais apropriado. Atualmente a linguagem é padronizada pela ANSI e ISO [24]

Ela pertence a uma classe de linguagens chamadas "linguagens de consulta"apesar dessa denominação ser um pouco inadequada, pois ela contém também comandos para definição e manipulação dos dados. Dessa maneira, a linguagem SQL pode ser analalisada como tendo três partes: uma linguagem de definição de dados (DDL, *Data Definition Language*); outra linguagem para manipulação (DML, *Data manipulation Language*); e por último uma para controle de acesso (DCL, *Data Control Language*). [21]

Uma característica desse tipo de linguagem é que são "declarativas", ou seja, o usuário descreve quais dados são de seu interesse sem descrever como a análise dos dados é feita

para se obter os resultados. Isso garante um nível maior de abstração, de maneira que a tarefa geralmente complexa de busca e seleção dos dados não precisa ser considerada. [20]

2.5 Aplicação Android

Esta seção tem como objetivo descrever uma aplicação Android acerca de sua estrutura, navegação dentro da estrutura e *design*, mas primeiramente é necessário familiarizar-se com o conceito de atividade.

2.5.1 Conceito: Atividade

Atividade é uma "ação"que o usuário deseja fazer dentro do aplicativo, uma determinada tarefa que será realizada. É cada diferente alteração, busca, edição, seleção que ocorre dentro de um aplicativo, como por exemplo preencher um formulário, selecionar uma música, tirar uma foto, etc.

Geralmente cada atividade interage com o usuário por meio de uma UI (*User Interface*) específica para ela, e essa por sua vez cobre toda a tela do dispositivo. Quando se navega para uma próxima atividade, a atual não é mais exibida e a nova toma toda a tela. Esse tipo de navegação é preferida por ser intuitiva.

Contudo, esse formato não é uma regra e a interface com o usuário pode ser dada através de janelas flutuantes ou dentro de outras atividades. Uma atividade tem um determinado ciclo de vida e diferentes estados possíveis. O aplicativo muda de estado dependendo de ações do usuário ou do sistema operacional sobre o aplicativo, como por exemplo abrir o aplicativo, iniciar nova atividade ao clicar um botão, etc. Os quatro estados essenciais de uma atividade podem ser vistos na 2.8, e são eles: [3]

- Quando a atividade está sendo exibida e recebe o foco do usuário, ela está ativa (*running*)
- Quando ela está visível mas perdeu o foco do usuário, como por exemplo uma janela que não cobre toda a tela aparece ou uma atividade transparente a encobre, a atividade está em pausa (*paused*). Ela mantém toda a suas informações e variáveis, mas pode ser fechada pelo Android caso falte memória ao sistema.
- Se a atividade está completamente oculta por outra atividade, ela está parada (*stopped*). As variáveis e outras informações são mantidas, mas a sua UI será primeiramente oculta

e com frequência é terminada pelo sistema operacional para livrar memória ao sistema.

- A atividade pode ser encerrada (*shut down*) pelo sistema operacional pedindo-lhe que termine suas atividades ou simplesmente terminando seu processo. Quando for restaurada, ela deve recuperar o estado anterior.

Como visto na figura 2.8, ao mudar de estado, uma subrotina apropriada é executada. Essas são muito importantes para manter o aplicativo funcionando corretamente e ocupar somente os recursos necessários, mesmo que o usuário saia e retorne ao aplicativo ou notificações do sistema apareçam durante sua execução.

2.5.2 Estrutura

A estrutura de uma aplicação pode ter a forma que o seu desenvolvedor desejar. Contudo, existe uma estrutura genérica que persiste entre diferentes aplicações e se mostra eficiente, além de ser um padrão entre aplicações [4] . Mantendo-se a navegação dentro do aplicativo dessa forma, o usuário reconhece rapidamente como navegar entre os diferentes níveis e atividades quando busca por uma função do aplicativo. Essa estrutura é mostrada na figura 2.9:

Topo

As atividades no nível mais alto são a primeira imagem que o usuário terá quando abrir o aplicativo. Elas se tratam de diferentes representações dos mesmos dados ou expõe as diferentes funções do aplicativo, como exemplificado na figura 2.10. É necessário se ter atenção especial com o *design* dessas atividades, bem como escolher a maneira adequada de navegação entre elas (tabs, barra de menu, etc), pois essas características junto com as espectativas do usuário influenciam muito a sua experiência com o aplicativo.

Categorias

Essas atividades são intermediárias entre o topo e as atividades detalhadas. Elas geralmente representam um grupo dos dados a ser trabalhados, com o intuito de organizar a navegação. É recomendável evitar essas etapas intermediárias e criar "atalhos"que vão direto do nível alto para a vista detalhada, se possível (vide 2.11). Entretanto, muitas vezes essas atividades podem oferecer um gerenciamento de vários itens ao mesmo tempo, contribuindo para uma melhor experiência do usuário.

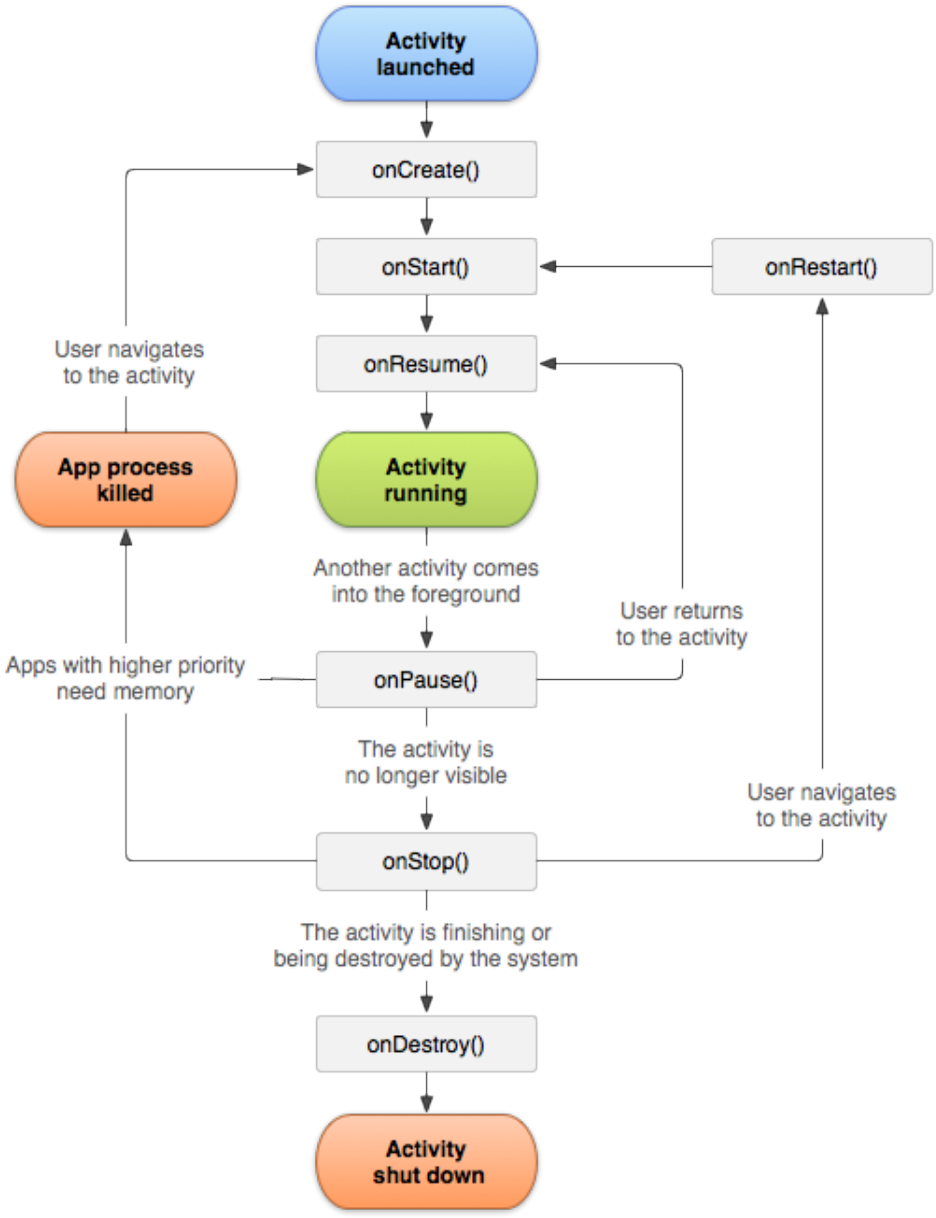


Figura 2.8: Ciclo de vida de uma atividade [3]

Figura 2.9: Estrutura genérica de um aplicativo [4]

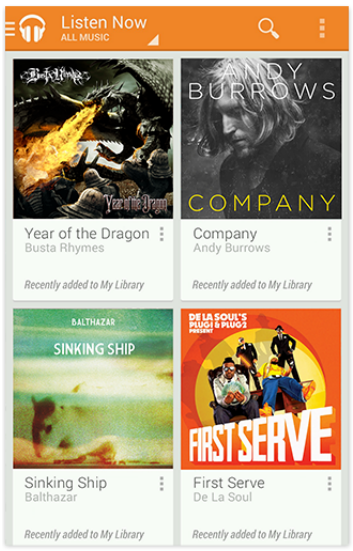


Figura 2.10: Atividade "topo"[4]

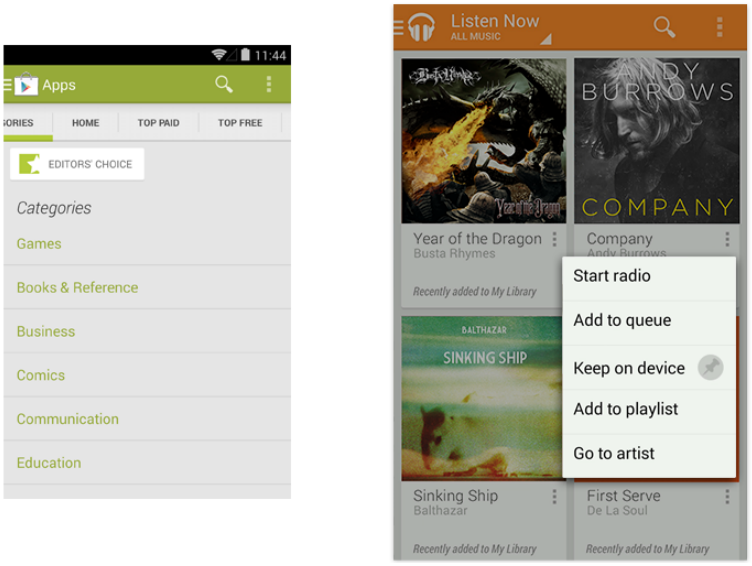


Figura 2.11: Atividade "categorias"(esquerda) e um "atalho"(direita) [4]

Detalhe

Esse é o nível mais baixo, onde o usuário atua diretamente sobre uma unidade de dado, como por exemplo toca ou para uma música, edita um contato (figura 2.12), redige uma anotação, etc. Aqui, o mais importante é a maneira com que os dados e características são apresentados e trabalhados.

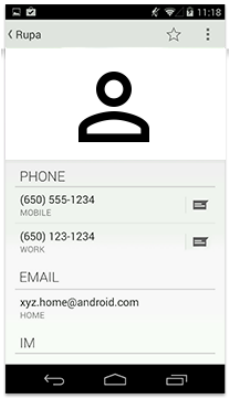


Figura 2.12: Atividade tipo "detalhe"(Adaptado de [4])

2.5.3 Navegação

A navegação dentro de um aplicativo é o tipo de esforço que se bem executado passa despercebido, mas se houverem problemas a experiência do usuário é arruinada. É a forma com que se acessa os diferentes conteúdos oferecidos pelo aplicativo, semelhante a uma navegação

entre websites.

Navegação para cima

Tendo em vista a hierarquia explicada na seção anterior, navegar para cima é dizer que se retorna para um nível superior de um nível mais baixo na hierarquia como mostrado na figura 2.13. Por exemplo, voltar de uma vista detalhada para uma de categorias. [5] Desde a introdução da versão Android 3.0 (Honeycomb, em 2011), o padrão é usar barras de navegação e não se exige mais que o aparelho tenha um botão "voltar". Entretanto este ainda deve ser suportado, ou seja, deve haver um botão na barra de menu com a mesma função do botão "voltar"do aparelho.

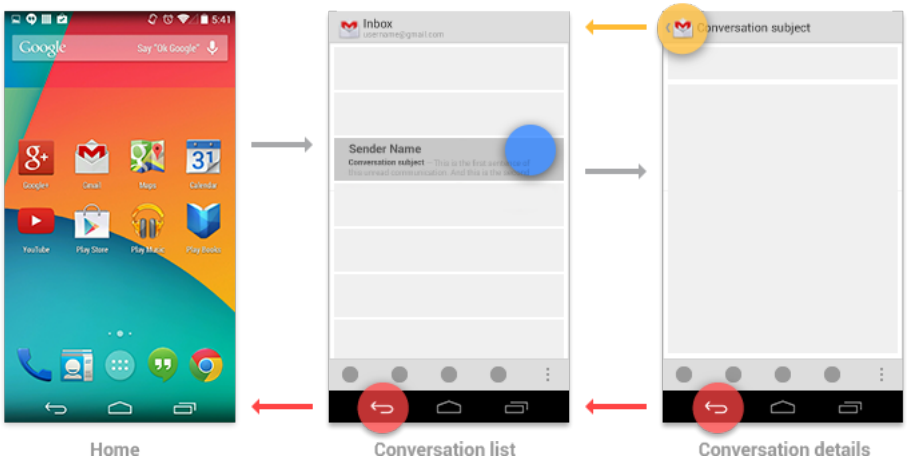


Figura 2.13: Navegação para cima [5]

É importante ressaltar que o botão voltar não tem somente a função de navegação entre telas como é o caso do menu de navegação. Ele pode, por exemplo, descartar janelas flutuantes (caixas de diálogo, *pop-ups*), descartar barras de ação contextuais, remover o destaque dos itens selecionados, ocultar o teclado da tela ou sair do aplicativo.

Navegação para o lado e para baixo

A navegação para baixo ocorre quando o usuário se desloca de um nível elevado para um nível mais baixo na hierarquia de atividades (vide figura 2.14). Isso geralmente é feito selecionando-se um botão dentro de um menu de opções ou um item de uma lista. Como o número de camadas necessárias para e ter acesso a um item individual deve ser pequeno para que se tenha uma boa experiência com o aplicativo, o número de níveis nesse tipo de navegação também deve ser pequeno. [6]

Já a navegação para o lado oferece uma maneira prática de se navegar por diferentes opções do aplicativo. Uma maneira muito usada para realizá-la são as chamadas *"tabs"*. Essas linguetas contém o título das diferentes seções do aplicativo, e ao se selecionar ou deslizar o dedo sobre a tela uma nova seção ganha o foco do usuário

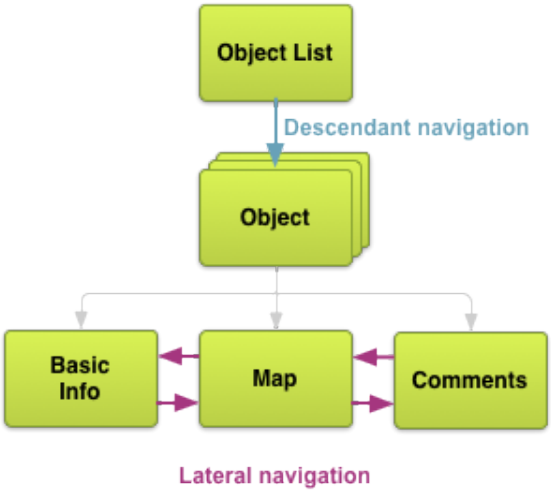


Figura 2.14: Navegação para baixo e para o lado [6]

2.5.4 *Design: Layouts*

O que define uma estrutura para interface com o usuário no sistema android são os chamados "layouts". Existem duas principais maneiras de se criar elementos de *layout*: declará-los em arquivos "xml"ou dinamicamente criar seus elementos usando objetos instanciados no codigo fonte. Esses dois métodos podem inclusive ser utilizados em conjunto, por exemplo criar uma estrutura básica em xml e adicionar itens em tempo de execução. [7]

Há várias vantagens em se declarar elementos em arquivos de *layout*. Uma delas é separar a apresentação do aplicativo do seu comportamento, em outras palavras, separar a lógica de funcionamento da forma com que os dados são exibidos. Isso se extende de tal forma que é possível efetuar modificações no *design* e testá-las sem recompilar o código-fonte do aplicativo (se esse não for alterado).

Outra grande vantagem é que se pode cirar facilmente arquivos diferentes que serão usados para diferentes orientações de tela, tamanhos de tela, idiomas, entre outros. Como aparelhos android possuem os mais variados formatos de tela, é muito importante considerar essas diferenças para que todos os usuários possam ter uma boa experiência.

A estrutura desses arquivos é muito semelhante a usada em arquivos HTML: são usados

uma série de elementos encadeados e devem conter exatamente um elemento base (*root*) do tipo *View* ou *ViewGroup*. Elementos view são os blocos básicos de visualização, como *Textview, Space, Imageview, WebView, KeyboardView*, entre outros. *ViewGroup* por sua vez são compostos de outros "View filhos", entre eles *LinearLayout, RelativeLayout, Toolbar e Adapterview*, como mostra a figura 2.15.

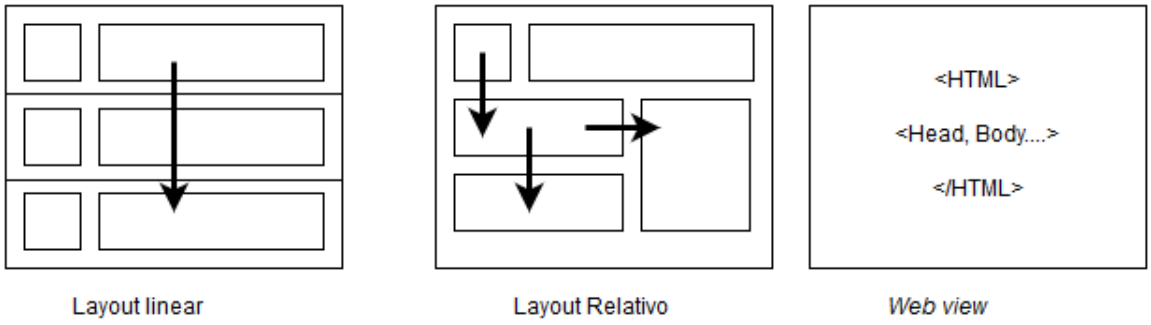


Figura 2.15: Exemplos de elementos tipo ViewGroup e View (adaptado de [7])

2.6 Portabilidade em aparelhos android

Quando se publica um aplicativo na google Play, ele fica visível para mais de 1 bilhão de usuários em mais de 190 países [25] . Por isso, é muito importante se levar em consideração as diferentes propriedades dos aparelhos sendo usados no mercado como tamanho de tela, densidade de pixels, língua local, etc. Por padrão, os aplicativos são tidos como próprios para uso em aparelhos celular e "tablets", mas há a possibilidade de desenvolvimento para aparelhos "portáteis"(*Wearables*), televisores e automóveis. Algumas dessas diferenças são discutidas a seguir:

Telas

Uma notável diferença entre aparelhos físicos é o tamanho de suas telas. Fazer um *design* que possa ser escalável, ou seja, que tenha todos os elementos acessíveis mesmo variando-se o tamanho da tela é a primeira opção. Contudo, mais adequado é o desenvolvimento de interfaces com o usuário específicas para cada configuração.

Os exemplos 2.16 e 2.17 a seguir mostram como as diferentes densidades de pixels (pixels por unidade de área) afetam a interface com o usuário:

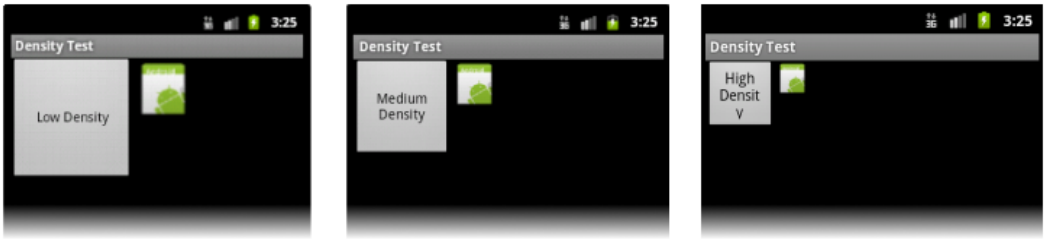


Figura 2.16: Exemplo de aplicação sem suporte a diferentes densidades de pixels [8]

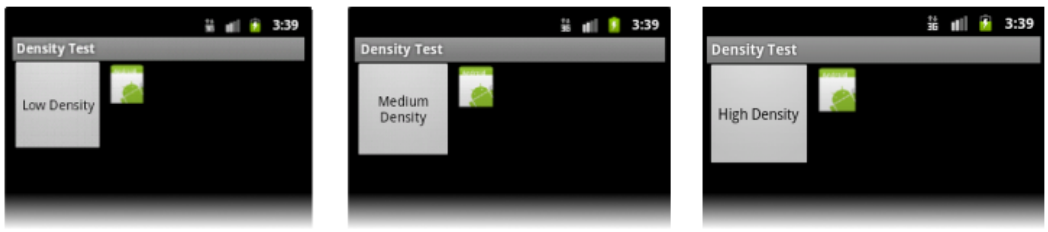


Figura 2.17: Exemplo de aplicação com suporte a diferentes densidades de pixels [8]

Idioma

Claramente, um outro aspecto que afeta diretamente a experiência do usuário é a apresentação do conteúdo num idioma que lhe seja familiar. Dessa maneira, é importante ao se desenvolver um aplicativo fazer com que a interface com o usuário responda de acordo com a configuração de idioma local.

Velocidade da conexão

Deve-se levar em consideração que aproximadamente metade dos usuários ainda usa conexão 2G em seus aparelhos móveis [26]. Isso implica em um aumento na importância de práticas que visam o uso mais eficiente desse recurso como a utilização de formatos compactos para imagens, redimensionamento dinâmico de imagens, armazenamento temporário de dados que possam ser reutilizados ou indentificação de conexões ruins e correspondente diminuição de seu uso.

Bateria

Aplicações que utilizem muitos sensores do aparelho drenam rapidamente a energia contida na bateria. Por isso é importante que esses sejam desligados e seus dados temporariamente salvos assim que sejam suficientes, administrar o uso de conexão com a internet e minimizar tarefas que sejam executadas em segundo plano é essencial. Na 2.18 são expres-

sos dois exemplos de aplicações com o intuito de comparar a potência utilizada relativo a conexão com a internet: a primeira (*unbounded transfers*) utiliza a conexão por um segundo a cada 18 segundos; e a segunda (*bundled transfers*) por 3 segundos a cada minuto. Os gráficos mostram o tempo em que o rádio do aparelho consome alta, baixa e nenhuma potência (*High Power, Low Power* e *idle* respectivamente).

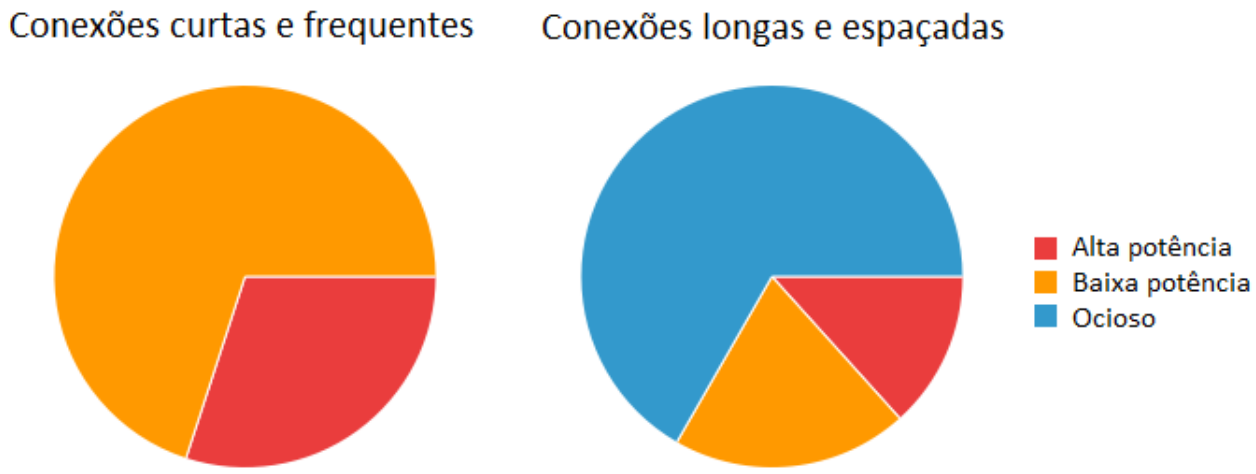


Figura 2.18: Diferença no consumo de potência usando conexões mais curtas e frequentes (esquerda) e conexões longas e espaçadas (direita) (adaptado de [9])

Memória: RAM

Processos que possam se manter por longos períodos de tempo durante o uso da aplicação também devem ser evitados pois geram atrasos ao se navegar dentro e fora da aplicação por usar recursos de memória e processamento. No geral, uma aplicação deve ser planejada de maneira que ela seja iniciada, realize as tarefas respectivas ao estado atual e se deslige novamente.

Tamanho da aplicação

Geralmente a parte que mais contribui no tamanho final de uma aplicação são os elementos gráficos. Portanto, as medidas citadas para diminuição do tamnho desses recursos no intuito de dimiuição do uso da conexão com internet também são validas aqui.

Outro cuidado importante que se deve ter é importar somente as bibliotecas necessárias para a sua aplicação, e ao se usar bibliotecas externas, verificar se elas são apropriadas para aplicativos móveis.

Facilitar o gerenciamento de dados pelo usuário, como fazer possível mover a aplicação para o cartão SD ou apagar arquivos antigos, é muito importante, pois assim é menos provavél que o aplicativo seja desisntalado pelo usuário quando a memória do aparelho estiver escassa.

2.7 Considerações finais

O projeto desenvolvido se baseia no uso da internet para proporcionar elevada praticidade e abrangência ao sistema. Ele se baseia na suposição de que os aparelhos eletrônicos têm acesso a ela todo o tempo, e por extensão seus usuários.

Fazendo uso de desenvolvimento Web e *Cloud Computing* essa solução se torna estável, confiável e com escalabilidade.

Programando-se com foco em aparelhos Android tem-se uma elevada abrangência em número de aparelhos que podem usar o aplicativo móvel, principalmente se as diferenças em hardware forem levadas em consideração e a experiência do usuário for boa independente de seu aparelho.

Assim, espera-se que o sistema seja uma solução muito acessível e que possa aumentar de tamanho e incorporar funcionalidades com o aumento do número de usuários e lançamento de novas versões.

Capítulo 3

Materiais

Esse capítulo descreve os materiais usados no desenvolvimento do projeto. São descritos basicamente os softwares e linguagens de programação utilizados

3.1 Servidor

No intuito desse projeto, é necessário um servidor-web com interface com usuários, uma pequena aplicação para gerenciar suas instruções e uma base de dados a ser acessada por essa aplicação, conforme descrito no capítulo anterior. Nota-se que as buscas e interações que os usuários farão no banco de dados também devem ser executadas pelo servidor, cabendo aos clientes somente enviar pedidos e receber os dados correspondentes. Assim, a velocidade com que a aplicação realiza suas tarefas é importante para o desempenho do sistema como um todo.

3.1.1 Criando o próprio servidor

Nessa primeira etapa de desenvolvimento, o servidor usado foi um computador portátil no qual foi instalado sistema operacional Linux.

A vantagem dessa abordagem é que o servidor é totalmente controlado pelo desenvolvedor, e os dados ficam fisicamente acessíveis a todo instante. Contudo, conforme discutido posteriormente, as vantagens de terceirizar a implementação do servidor são maiores. Entre elas pode-se destacar a escalabilidade, pois os serviços atuais de servidor ampliam suas capacidades de armazenamento e processamento conforme exigido pelos usuários, enquanto que no caso de se usar um computador novo hardware precisa ser instalado toda vez que se quiser aumentar sua capacidade.

Last 12 months			Last 6 months			Last 3 months			Last 1 month		
1	Mint	3001▼	1	Mint	2887▼	1	Mint	2706▼	1	Mint	2608▼
2	Debian	1852▼	2	Debian	1673▼	2	Debian	1681▼	2	Debian	1682▼
3	Ubuntu	1588▼	3	Ubuntu	1353▼	3	Ubuntu	1376▼	3	openSUSE	1549▲
4	openSUSE	1282▼	4	openSUSE	1219▲	4	openSUSE	1340▲	4	Ubuntu	1497▼
5	Manjaro	1083▼	5	elementary	1085▬	5	elementary	1221▼	5	elementary	1343▼
6	Fedora	1061▼	6	Manjaro	1043▼	6	Manjaro	996▲	6	Manjaro	1173▼
7	elementary	884▲	7	Fedora	997▼	7	Fedora	962▬	7	Zorin	1141▼
8	Zorin	875▲	8	Zorin	914▬	8	Zorin	943▲	8	Fedora	1028▼
9	CentOS	836▬	9	CentOS	777▬	9	deepin	747▲	9	CentOS	783▲
10	Arch	793▬	10	deepin	727▲	10	CentOS	737▬	10	Antergos	748▲
11	Mageia	733▼	11	Arch	725▼	11	Arch	708▼	11	Solus	741▲
12	deepin	727▲	12	PCLinuxOS	610▬	12	Antergos	641▬	12	Arch	734▼
13	PCLinuxOS	673▲	13	Ubuntu MATE	580▼	13	Ubuntu MATE	582▬	13	Lite	721▼
14	Android-x86	654▬	14	Antergos	575▲	14	PCLinuxOS	580▲	14	deepin	709▲
15	Ubuntu MATE	650▬	15	Mageia	569▼	15	Solus	568▲	15	Ubuntu MATE	645▼

Figura 3.1: Ranking das distribuições baseadas em Linux [10]

Sistema operacional

A distribuição foi escolhida por sua popularidade, sucesso e conhecida grande rede de colaboradores: Debian. Apesar de ser difícil de afirmar com precisão, Debian é provavelmente a segunda distribuição mais usada do sistema Linux e é conhecida pela estabilidade e controle de qualidade para suas atualizações [10].

O ranking das diferentes distribuições baseadas em Linux é mostrado a seguir na figura 3.1, acessada em 10 de novembro de 2016. A terceira coluna de cada tabela mostra o número de acessos à pagina usando cada uma das distribuições.

Servidor Web

Instalar um servidor Web em um sistema Linux é uma tarefa muito fácil: basta um comando (apt-get no caso do Debian) e a máquina já responde a pedidos HTTP na porta 80 com uma página web.

Há diversas opções de servidores web para sistema linux. As mais comuns e que foram consideradas para esse projeto foram: Apache, Lighttpd e Nginx.

Lighttpd (chamado de "*lighty*") oferece uma boa escalabilidade e um eficiente uso de memória se comparado aos seus concorrentes diretos, como pode ser visto na figura 3.2. É *open-source* e adequado para uso de páginas estáticas, Ruby on Rails e PHP.

Apache tem a vantagem de ser antigo e muito popular, pois isso faz com que exista muita documentação, suporte a seus usuários e compatibilidade com demais programas. Contudo, o uso de memória não é muito eficiente quando o número de requisições aumenta, como visto na figura 3.2.

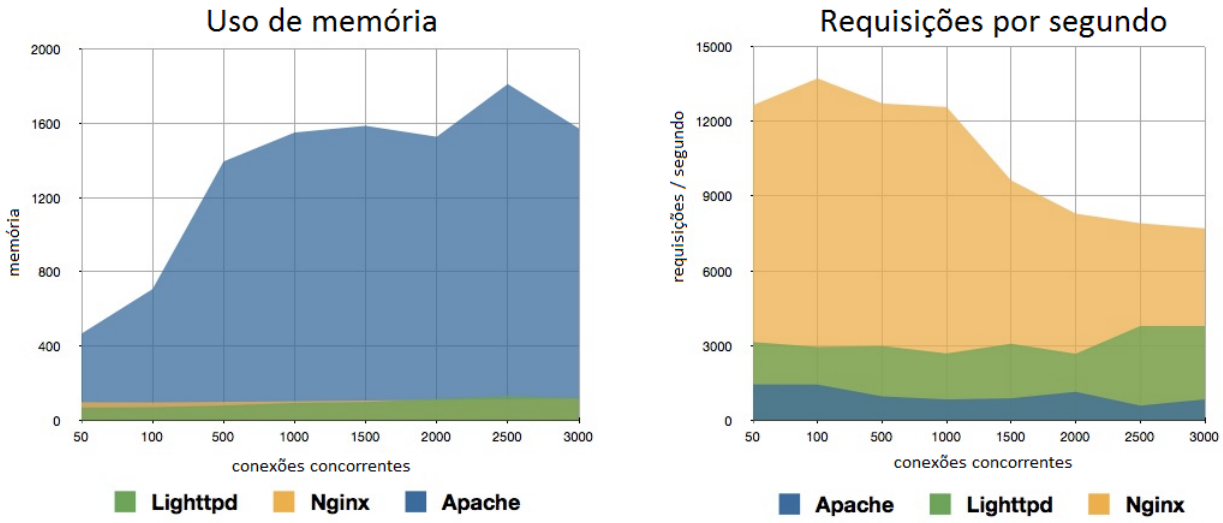


Figura 3.2: Comparação entre web servers, adaptado de [11]

Nginx foi desenvolvido como sendo servidor web e servidor proxy. Também é leve como lighttpd e consegue boa escalabilidade com pouco hardware. Contudo, não trata conteúdo dinâmico nativamente. Para gerenciar requisições PHP, por exemplo, é necessário repassá-las a um processador externo que retorna para o servidor web o conteúdo *"renderizado"*. Isso faz com que seja necessário configurar essa interface.

Um novo concorrente desses servidores Web é o chamado Flask. Ele não é um servidor Web propriamente dito, e sim um "*web framework*". Isso quer dizer que ele não possui diversos recursos como acesso a bases de dados de forma nativa, validação de formulários, autenticação de usuários, entre outros. Também trata uma requisição de cada vez, de forma serial, fazendo-o ineficiente no caso de diveros usuários tentarem acessar o servidor ao mesmo tempo.

Devido às vantagens citadas, a sua popularidade e a familiaridade do desenvolvedor com o programa, foi escolhido para esse projeto o servidor web Lighttpd (chamado de "Lighty"). Depois de instalado, basta colocar as páginas web dentro do diretório /var/www e elas são automaticamente carregadas quando se acessa o endereço do servidor.

3.1.2 Site e aplicação

Para construir os sites que serão acessados pelos usuários (por meio do Browser ou aplicativo), é necessário um conjunto de arquivos html. Essa linguagem de hipertexto é uma forma padrão para descrição de páginas web.

Essas páginas devem ser carregadas com informações de acordo com as instruções rece-

bidas do usuário e informações na base de dados, para então serem enviadas ao usuário. Sua programação é descrita na seção seguinte.

Programação

A tarefa do site e da aplicação não pode ser realizada apenas com páginas estáticas HTML. Para executar tarefas dinâmicas, usa-se outras linguagens como as descritas na seção "Programação Web"2.4 do capítulo "Embasamento Teórico".

As instruções são passadas do usuário para o servidor usando o método POST do HTTP e carregadas para a aplicação dentro da página HTML. Esta aplicação usa essas informações para buscar na base de dados e responder com código HTML correspondente.

Para programação do site deste projeto, as páginas Web baseadas em HTML foram extendidas com código PHP e com os chamados "PHP PDOs"para acessar a base de dados MySQL. A criação de uma conexão com a base de dados ocorre de maneira simples, como mostrado na figura 3.3.

```php
<?php
$servername = "localhost";
$username = "username";
$password = "password";

try {
    $conn = new PDO("mysql:host=$servername;dbname=myDB", $username, $password);
    // set the PDO error mode to exception
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    echo "Connected successfully";
    }
catch(PDOException $e)
    {
    echo "Connection failed: " . $e->getMessage();
    }
?>
```

Figura 3.3: Acessando a base de dados com PHP [12]

Em seguida, usa-se esse objeto para interagir com a base de dados executando comandos SQL dentro do código PHP. Os resultados das buscas, inserções e alterações na base de dados são então tratados usando PHP, em seguida traduzidos em HTML para serem exibidos no site.

3.1.3 Base de dados - MySQL

Para a construção da base de dados a princípio foi escolhido um dos sistemas mais aceitos de bancos de dados relacionais: MySQL. As razões que levaram a essa escolha são semelhantes às que pesaram na decisão da distribuição Linux: é um sistema considerado confiável e estável. Essas qualidades são muito apreciadas quando se procura elementos para um sistema que exigirá bom desempenho das interfaces ente seus componentes.

A instalação ocorre sem mais problemas em um sistema com capacidade suficiente para o programa, e o acesso ocorre utilizando-se chamadas específicas nas páginas HTML que contém código PHP.

3.1.4 Usando *Cloud Computing*

O formato final do sistema usa um servidor Web do tipo *Cloud Computing*. Isso quer dizer que a hospedagem do site e da base de dados ocorre dentro de um contexto dinâmico no qual recursos são alocados de acordo com a demanda.

A plataforma escolhida, "Firebase"da empresa Google, conta com os recursos de um servidor Web como "host"para sites, base de dados, autenticação de clientes, entre outros. Além disso, há diversas ferramentas para monitoramento, gerenciamento, segurança e monetização da aplicação e seu uso. Uma visão geral desses é esquematizado na figura 3.4.

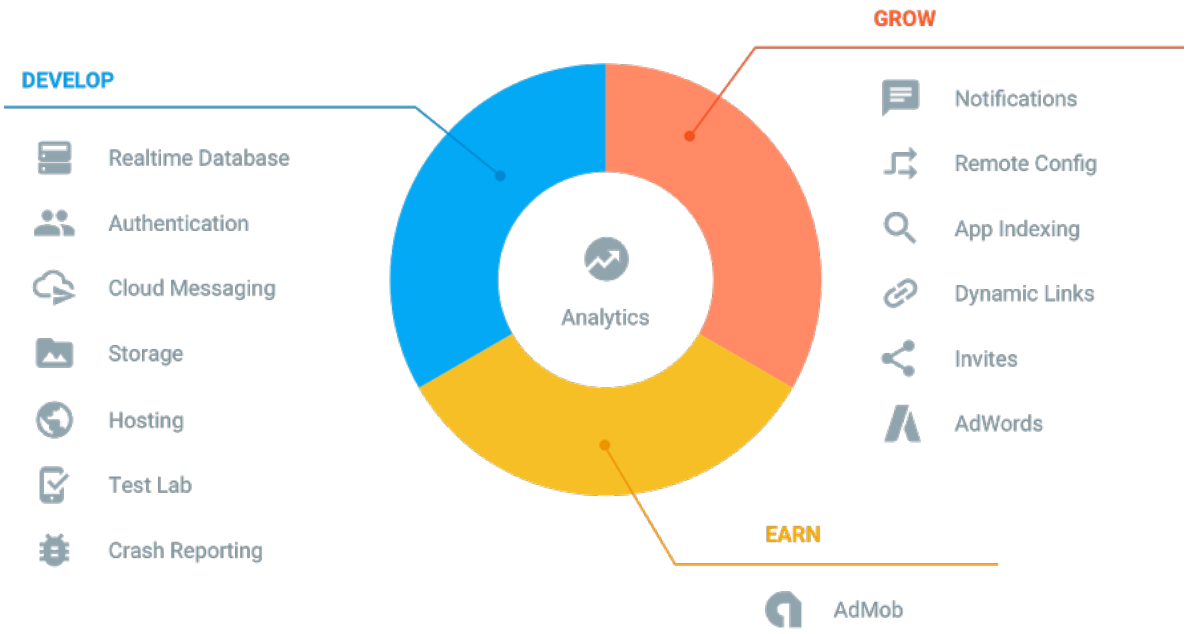


Figura 3.4: Recursos do servidor *Cloud* [13]

A principal vantagem no uso dessa plataforma foi que ela é apropriada para o uso em conjunto com aplicações móveis Android, já possuindo uma interface adequada para sua comunicação.

Além disso, essa plataforma pode ser incorporada a um sistema desenvolvido que use a chamada "Google Cloud". Esta é outra infraestrutura online que contém ferramentas para desonvolvimento e gerenciamento de aplicações. A "Google Cloud"conta com ferramentas de aprendizado de máquina, análises de Big Data, *networking*, vários tipos de base de dados, suporte online, entre outros.

3.2 Escolha da plataforma móvel

A escolha do desenvolvimento do aplicativo como sendo voltado para aparelhos do sistema operacional Android se deu principalmente pelo fato de se tratar de um sistema livre. Como o desenvolvedor não possuía experiência em programação de aplicações móveis ("programação *mobile*"), um fator de peso para a escolha da plataforma foi a quantidade de suporte gratuito disponível.

A empresa Google disponibiliza uma grande quantidade de materiais, tutoriais e até suporte gratuitamente e sem a necessidade de cadastro. A principal referência é o site "www.developer.android.com", que possui uma enorme quantidade de informações organizadas em diferentes níveis de detalhe. Além disso o ambiente de desenvolvimento apropriado, o Android Studio (detalhado na seção a seguir), é disponibilizado de maneira gratuita pela empresa.

Dessa maneira todos os recursos necessários para se desenvolver aplicações Android estão disponíveis gratuitamente e com fácil acesso. Somente quando se deseja publicar o aplicativo na loja virtual "Google Play"é que se torna necessário um cadastro e o pagamento de uma taxa.

3.3 Android Studio

O aplicativo foi criado usando IDE oficial da empresa Google, Android Studio. Esse ambiente de desenvolvimento possui diversas funcionalidades como por exemplo preenchimento automático de comandos, auto-identação, navegador de pastas, simulação de aparelho android, e muitas outras.

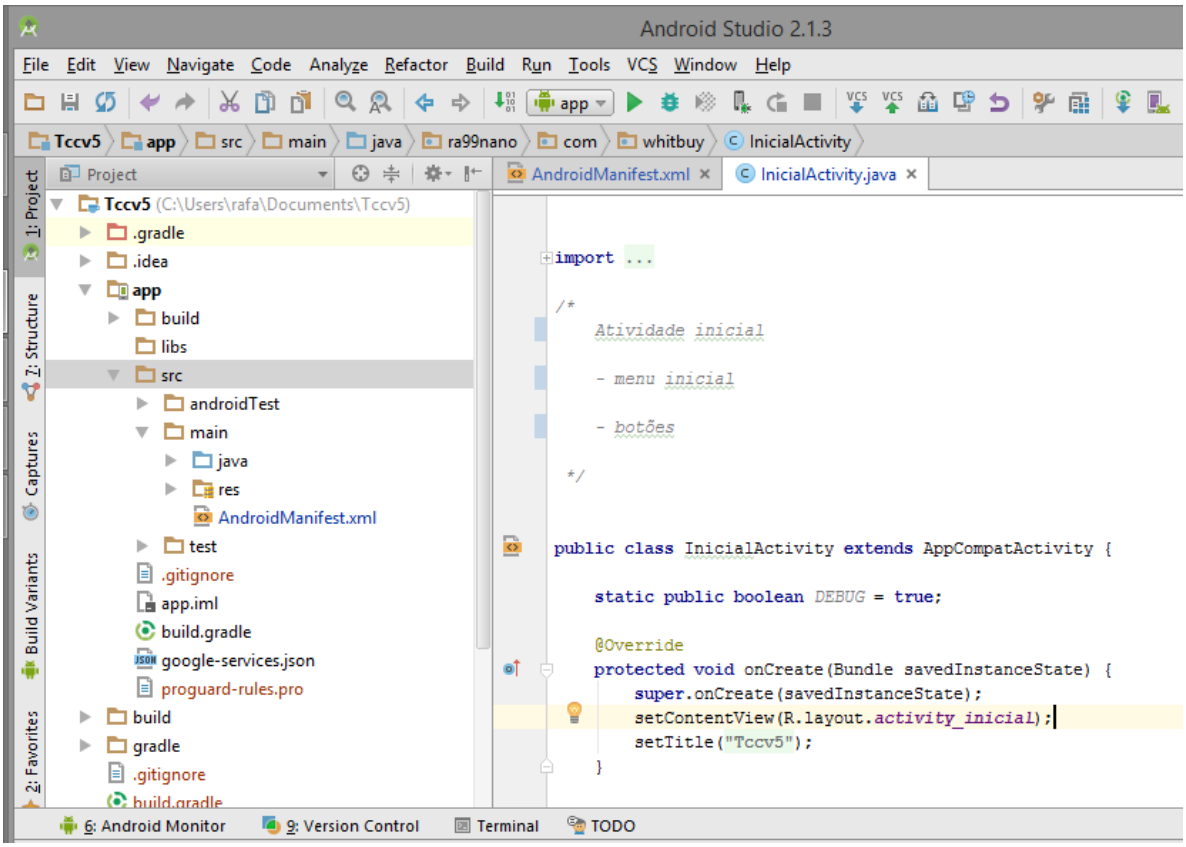


Figura 3.5: Ambiente de desenvolvimento Android Studio

Num projeto do Android Studio, o aplicativo é composto por diferentes arquivos separadas em três grandes pastas: manifests, java e res, detalhadas a seguir. A figura 3.5 mostra o ambiente de desenvolvimento. Nota-se do lado esquerdo a organização das pastas e arquivos que compõe o projeto.

3.3.1 Manifesto

Na pasta manifests é colocado um arquivo precisamente nomeado "AndroidManifest.xml"no qual são declaras as especificações gerais do aplicativo: quais recursos do aparelho são usados, versão, nome do pacote Java, versão mínima do Android que suporta o aplicativo, entre outros. [27]

No aplicativo desenvolvido, o manifesto conta apenas com as declarações de nome, versão mínima do Android, classe inicial e permissão para acessar a internet.

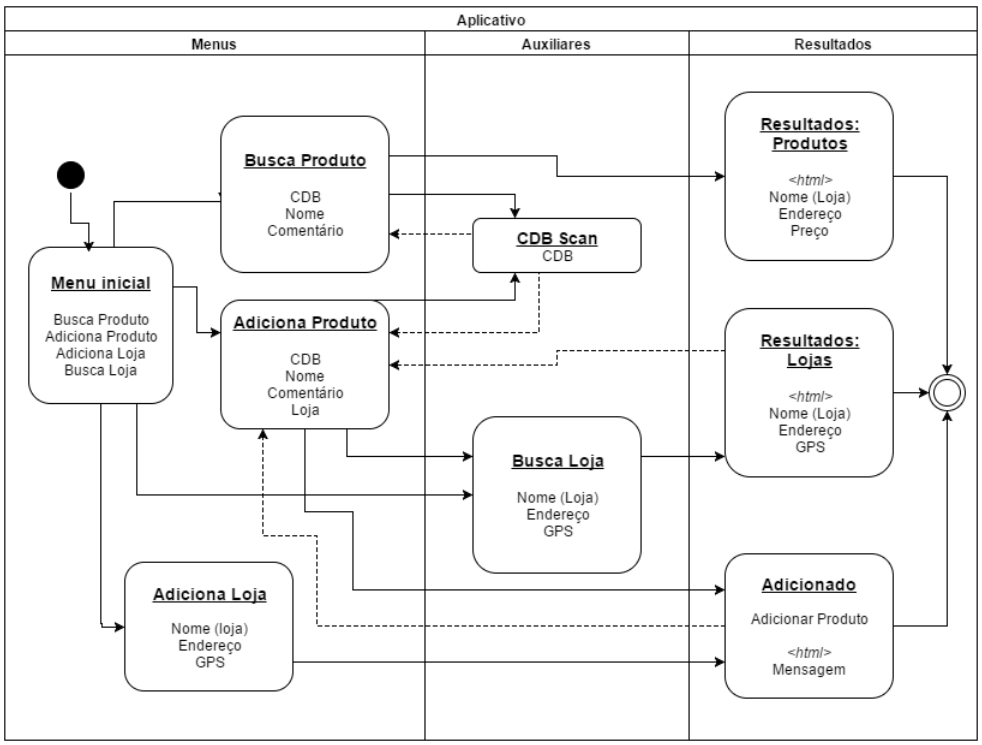


Figura 3.6: Atividades do aplicativo

3.3.2 Diretório Java

Na pasta com o nome sugestivo "java"são colocados os arquivos em linguagem java que descrevem o aplicativo e seu funcionamento, ou seja: o pacote java, suas classes e demais elementos. É onde fica realmente o código do aplicativo, separado em diferentes atividades. Cada atividade, que pode ser pensada como um estado de uma máquina de estados, contém variáveis, funções, e está conectada a uma interface com o usuário. De acordo com os comandos do usuário se inicia a próxima atividade ou se retorna à anterior levando novos dados.

A estrutura escolhida para as atividades do aplicativo está exposta a na figura 3.6, e um fluxograma representando o código de cada uma delas consta em anexo. De maneira muito análoga à separação clássica em topo, categorias e detalhe, atividades no projeto desenvolvido foram separadas em três grandes grupos: menus, auxiliares e resultados. Essa separação não tem reflexo nenhum sobre a implementação do aplicativo, tem por fim somente classificá-las de acordo com seu funcionamento. Atividades marcadas com a tag <html> possuem uma interface com o servidor.

As atividades agrupadas como sendo parte do "menu"tem como principal função proporcionar uma navegação coerente dentro do aplicativo conforme a intenção do usuário. É nessas

atividades, também, que ficam salvos os dados a serem enviados ao servidor.

As atividades denominadas como "auxiliares"são responsáveis por buscar dados antes de realizar a busca ou inserção de produtos ou lojas, ou seja, auxiliar o usuário no preenchimento dos campos necessários. Por exemplo, a atividade "CDB Scan"é responsável por escanear o código de barras do produto e retornar seu valor ao campo adequado, para que não seja exigido do usuário digitar o código manualmente.

Por último, as atividades denominadas "resultados"tem como principal objetivo exibir os valores da base de dados correspondentes à busca do usuário. Contudo, também é esperado dessas atividades que seja possível selecionar valores da busca pelo usuário, retornando dados para atividades anteriores. Dessa forma é possível ao final de uma busca selecionar um produto ou loja para a próxima já retornando à atividade correspondente.

3.3.3 Diretório res

No diretório res são colocados os recursos que o aplicativo usará, ou seja, os layouts descrevendo a interface om o usuário, imagens, ícones, textos e demais elementos estáticos usados pela aplicação.

No aplicativo em desenvolvimento usa-se imagens para melhor interface com o usuário e clareza na navegação. Além disso, todo texto usado no aplicativo é salvo na forma de recurso, pois otimiza a a memória usada pela aplicação. Também é definido um estilo para a aplicação usando-se um tema padrão do Android, uma prática recomendada pois caso haja lançamento de novas versões do sistema operacional, o aplicativo é automaticamente atualizado e mantém-se em conformidade com o restante das aplicações. Por último também são definidas as interfaces com o usuário, os chamados layouts, geralmente correspondendo a uma atividade.

3.4 Considerações finais

Os materiais utilizados no desenvolvimento desse projeto estão em acordo com tendências atuais para desenvolvimento de aplicações que usam a Internet como estrura de apoio para um serviço.

Android Studio, o ambiente de desenvolvimento (*IDE*) da empresa Google, está constantemente sendo atualizado e tem sempre novas funcionalidades incorporadas que estão em acordo com as tendências em desenvolvimento de aplicativos móveis.

O servidor escolhido se baseia em *Cloud Computing*, uma tendência forte na atualidade. Uma grande vantagem é que este pode ser incorporado ao sistema "Google Cloud", e assim fazer uso dos mais diversos recursos para análise, interpretação e gerenciamento dos dados coletados, bem como permitir a inclusão das novas funcionalidades que vão sendo adicionadas a essa plataforma periodicamente.

Capítulo 4

Métodos

Esse capítulo descreve primeiramente como foi o planejamento do projeto, sua estrura, organização e em seguida sua execução.

4.1 Planejamento

O primeiro passo no planejamento do sistema foi a definição das partes que viriam a compô-lo, descrito na próxima seção "Estrutura planejada". Ficou claro que o projeto se divide em duas grandes frentes: desenvolvimento de um aplicativo e de um servidor. O planejamento desses componentes ocorreu em seguida, e consta nas respectivas seções.

4.1.1 Estrutura planejada

A estrutura prevista para esse projeto é muito semelhente à tipica para aplicações web descrita no capítulo 1 em que clientes acessam através da internet um servidor, e este gerencia um banco de dados e as respostas para os clientes.

No caso específico do sistema implementado, são previstas duas interfaces diferentes de acesso a base, uma adequada ao usuário que utilizar um navegador web (em seu PC ou celular) e outra para o aplicativo desenvolvido especificamente para esse fim. No lado do servidor, um Web Server gerencia essas interfaces e repassa instruções para uma aplicação, que tem acesso ao banco de dados e responde apropriadamente. Essa estrutura pode ser expressa como na figura 4.1:

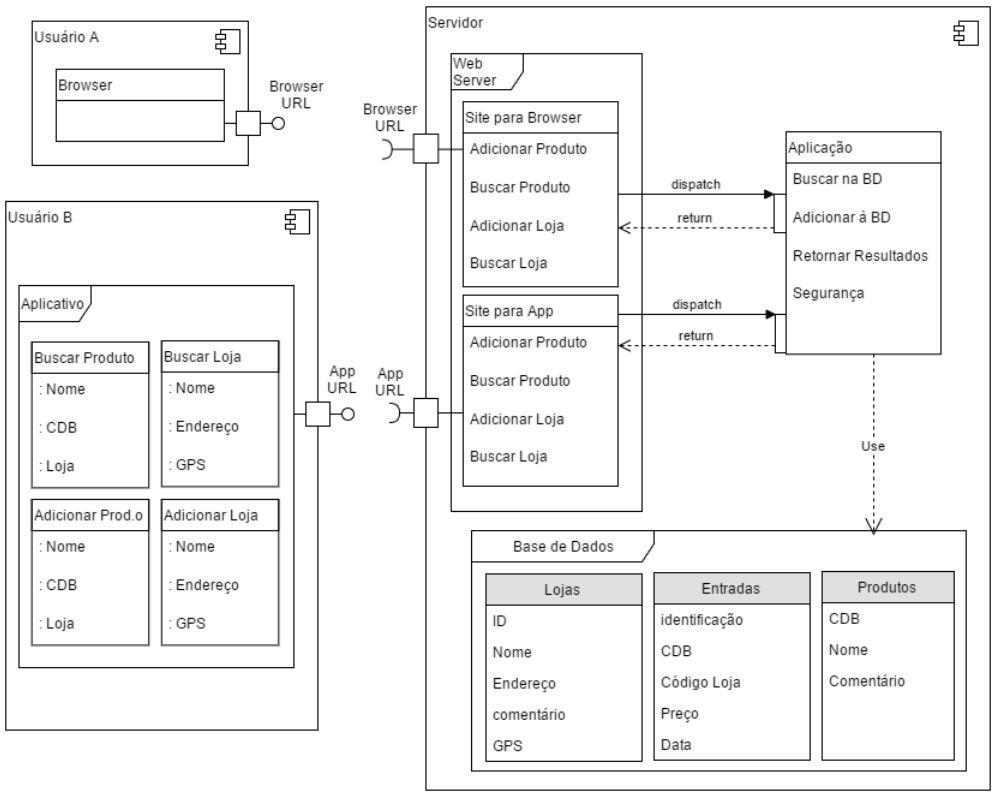


Figura 4.1: Estrutura do sistema

4.1.2 Planejamento do Aplicativo

Conforme descrito na seção de materiais, o aplicativo foi desenvolvido usando o ambiente de desenvolvimento da Google, o Android Studio.

O design escolhido foi um tema padrão disponibilizado pela Android. Essa prática é recomendada pois quando uma nova versão do sistema operacional é lançada, o aplicativo se atualiza automaticamente e fica sincronizado com as demais aplicações do aparalho.

Funções

O aplicativo foi planejado como sendo constituído de um menú principal que possibilita a seleção das quatro principais ações que o usuário deve realizar, como mostrado na figura :

- Adicionar produto: O usuário pode adicionar todas as informações sobre um produto, desde que preencha seu código de barras. Ele pode adicionar nome, tipo, comentário, preço e loja onde efetuou a compra. Caso um ou mais desses campos não sejam preenchidos, eles são ignorados e os demais são usados para atualizar o banco de dados.

- Adicionar Loja: Análogo à função de adicionar produto, esse opção serve para que

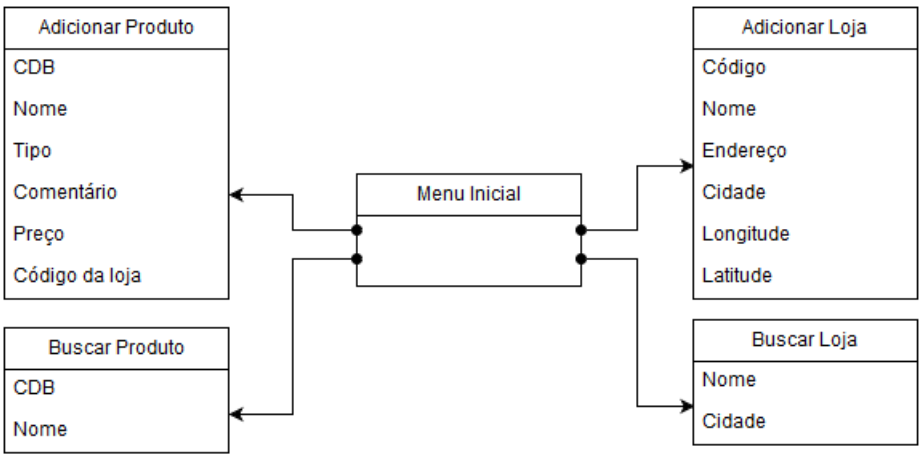


Figura 4.2: Estrutura das atividades do Aplicativo

o usuário entre com os detalhes da loja onde comprou o produto caso ela não esteja cadastrada ou tenha informações incompletas ou desatualizadas. Ele pode entrar com o nome, endereço, cidade, longitude e latidude da loja. As coordenadas geográficas devem ser adicionadas usando o sensor de posicionamento global do aparelho, quando o usuário se encontra deforonte à loja.

- Buscar produto: Quando se deseja buscar os preços cadastrados para um produto nas diferentes lojas, essa é a função utilizada. Uma opção para busca do produto é se entrar com o código de barras do mesmo, e para fazê-lo, o aplicativo conta com o apoio de outro aplicativo para escaneá-lo usando a câmera do aparelho. A segunda opção é buscar o produto pelo nome.

- Buscar Loja: É semelhante à busca por um produto, mas agora a busca pode ser feita entrando-se com o nome da loja ou com a cidade, e o aplicativo retornará as entradas correspondentes.

4.1.3 Planejamento do servidor

Inicialmente foi planejado que o servidor seria desenvolvido a partir de um computador com o sistema operacional Linux porque é gratuito, muito fácil de usar para essa aplicação e um sistema desse tipo já havia sido implementado com êxito pelo desenvolvedor.

Nesse computador é instalado um *webserver*, que gerencia as requisições HTTP. O programa escolhido para essa função foi o Lighttpd, como descrito na sua própria subseção.

Dentro desse *webserver* é criado um *website* composto por diferentes páginas HTML.

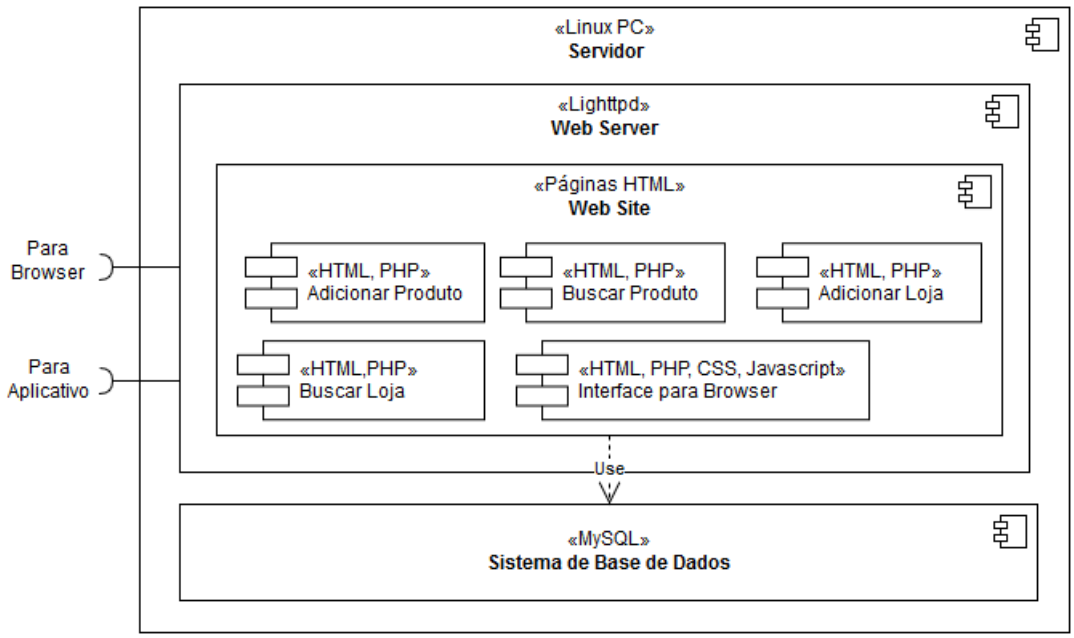


Figura 4.3: Servidor web e seus componentes

Cada página tem sua própria função, como adicionar ou buscar um item na base de dados ou criar a interface para o usuário que usa um navegador web. No caso do uso do aplicativo, as páginas corretas seriam diretamente acessadas pois a interface com o usuário ocorre localmente. Toda essa estrutura pode ser visto na figura 4.3.

Para execução de suas tarefas cada página usa a linguagem PHP e os chamados "PHP PDOs", que fazem com que a resposta à requisição HTTP ocorra de maneira dinâmica, ou seja, possa conter informações que não constam na página HTML em si. Essa interface pode é usada para acessar um sistema de base de dados (DBMS), e o sistema escolhido para a aplicação foi o difundido MySQL detalhadamente descrito em sua sub-seção.

Dessa forma o servidor cumpre a sua função de responder às requisições dos clientes adicionando ou buscando dados na base de dados. Além dessa estrutura, é necessário configurar a forma com que os clientes encontram o servidor na internet, como DDNS.

Contudo, conforme descrito na seção "Execução"4.2, o servidor Web usado foi alterado e com isso houve uma pequena mudança na estrutura do servidor como um todo.

Durante a busca por um servidor web para o site, foi encontrado um servidor web que já possui o banco de dados integrado e disponibiliza uma interface para comunicação com aplicativos android, como mostra a figura 4.4. Essas vantagens fizeram valer a pena a mudança do sistema já parcialmente implementado. A interface com o navegador web (o site composto pelas páginas web) pode ser portado para a nova tecnologia com mínimas adaptações.

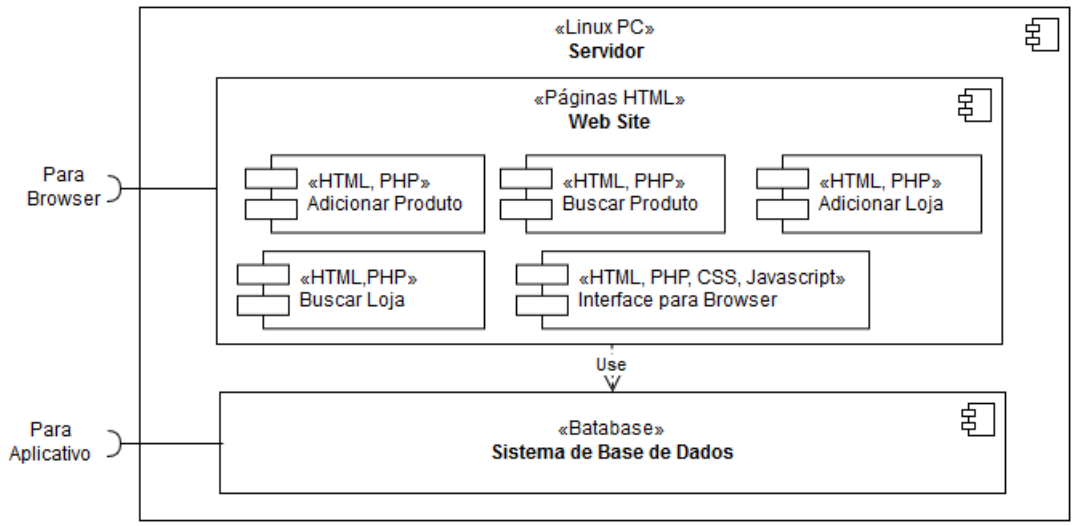


Figura 4.4: Servidor web Google e seus componentes

Base de dados

Para armazenar os dados dos produtos, lojas e preços foi planejada uma estrutura contendo três tabelas como mostra a figura 4.5:

- Entradas: Essa tabela armazena as entradas dos usuários, ou seja, cada linha representa a adição de dados feita por um usuário. Como o sistema deve comportar um número grande de entradas, essa tabela guarda apenas um mínimo de informações por entrada: código do produto, código da loja, preço, data e um índice da entrada. Esse índice é necessário para caso hajam duas entradas do mesmo produto na mesma loja, nenhuma informação seja perdida.

- Produtos: essa tabela contém as informações detalhadas sobre um produto. No sistema desenvolvido só contém código de barras, nome e produto, por se tratar de uma etapa intermediária no desenvolvimento.

- Lojas: De maneira semelhante à tabela produtos, essa contém os detalhes das lojas na base de dados: seu código, nome, endereço, comentário e coordenadas geográficas para uso do GPS.

A escolha dessa organização se deu com o intuito de minimizar o uso de memória necessário para cada entrada. Na tabela principal das entradas, não são armazenadas nenhuma das propriedades dos produtos ou lojas: somente seus códigos. As outras duas tabelas, por sua vez, guardam esses necessários para "traduzir"os códigos em informações relevantes e compreensíveis ao usuário: nomes, endereço, comentários, etc.

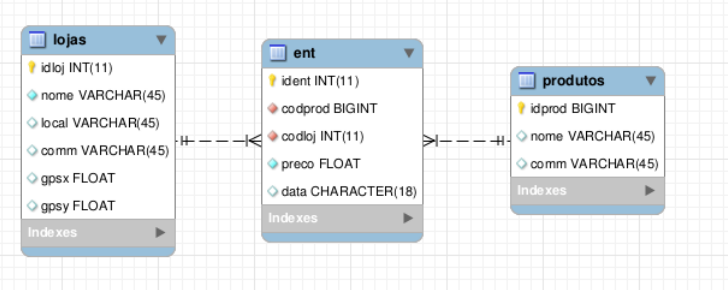


Figura 4.5: Base de dados relacional planejada

Assim, sempre que um usuário adiciona uma entrada no sistema, uma nova linha nessa tabela das entradas é criada com o código do produto, da loja, preço, data e hora da entrada. O programa no servidor fica então encarregado de guardar as demais informações relevantes (nome, endereço, etc. que tenham sido adicionadaos ou alterados) nas tabelas adequadas. Dessa maneira, nenhuma informação é perdida pois para toda e qualquer entrada de dados é criada uma entrada. Existiria no programa da aplicação do servidor um processo separado para verificar se as entradas são válidas ou relevantes, para "limpar"a base de dados de entradas incompletas.

Conforme descrito nas seções anteriores, quando houve a mudança de servidor web houve também uma alteração na estrutura do servidor porque o banco de dados agora é integrado com o servidor web. Contudo, o novo banco de dados não utiliza a estrutura relacional de armazenadmento de dados e sim o sistema de organização hierárquico.

Dessa maneira, o banco de dados teve que ser adaptado. A grande mudança é que neste não é necessário o uso de uma tabela específica para as entradas pois elas são salvas como ramos nos nós das árvores produtos e lojas. Em contrapartida são necessários índices para que se possa localizar os itens buscados sem ter de carregar a base de dados toda.

O novo modelo de armazenamento de dados é expresso na figura 4.6. Nesse novo sistema, a base de dados se dá usando o modelo hierárquico. O acesso a seus elementos é feito de maneira mais rápida, mas informações redundantes (criação de índices) são necessárias para que se possa realizar as buscas adequadamente como pode ser visto na figura 4.6.

Toda a informação referente às lojas, produtos e preços está contida nos nós "produtos"e "Lojas"e seus filhos. Contudo, os índices são necessários para que se possa realizar buscas sem que seja necessário buscar todos os filhos de um nó por uma prorpiedade específica.

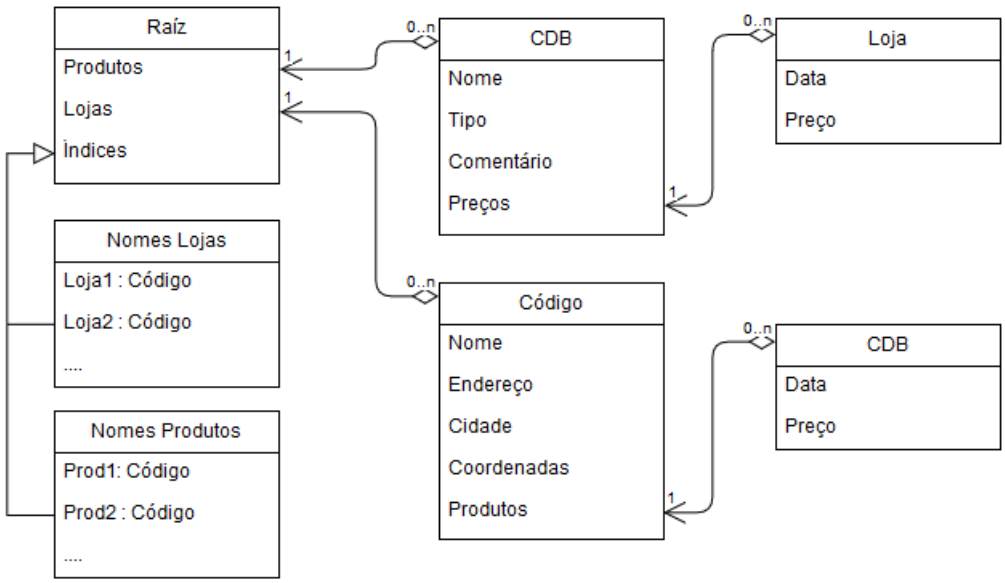


Figura 4.6: Base de dados hierárquica

4.2 Execução e funcionamento

O primeiro passo para desenvolvimento do projeto foi a criação do servidor usando um computador com sistema operacional baseado em Linux. Isso ocorreu sem demais prolemas, pois essa estrutura já era conhecida pelo desenvolvedor.

Para a criação da base dados, instalou-se MySQL e usando os comandos SQL se criou a base de dados composta por três tabelas.

Foram criadas então as diferentes páginas que executam as diferentes funções da aplicação, usando HTML e PHP (com PDO's).

Ao final dessas etapas um esqueleto do servidor estava pronto, e a seguir iniciou-se a programação do aplicativo Android com a principal função de comunicar-se com o servidor.

Tendo concluído a programação de um aplicativo que cumprisse essa tarefa de maneira simples, o próximo passo era colocar o servidor online. Foi então que houve uma mudança. Ao procurar por diferentes servidores, foi encontrada uma alternativa muito compatível com desenvolvimento de aplicações Android que compreende uma base de dados. A grande diferença é que agora não se utilizaria mais uma base de dados relacional (composta por tabelas), mas sim uma estrutura hierárquica. Dessa maneira a lógica do aplicativo teve de ser alterada.

4.2.1 Sobre o aplicativo

As seções a seguir descrevem o funcionamento e uso do aplicativo.

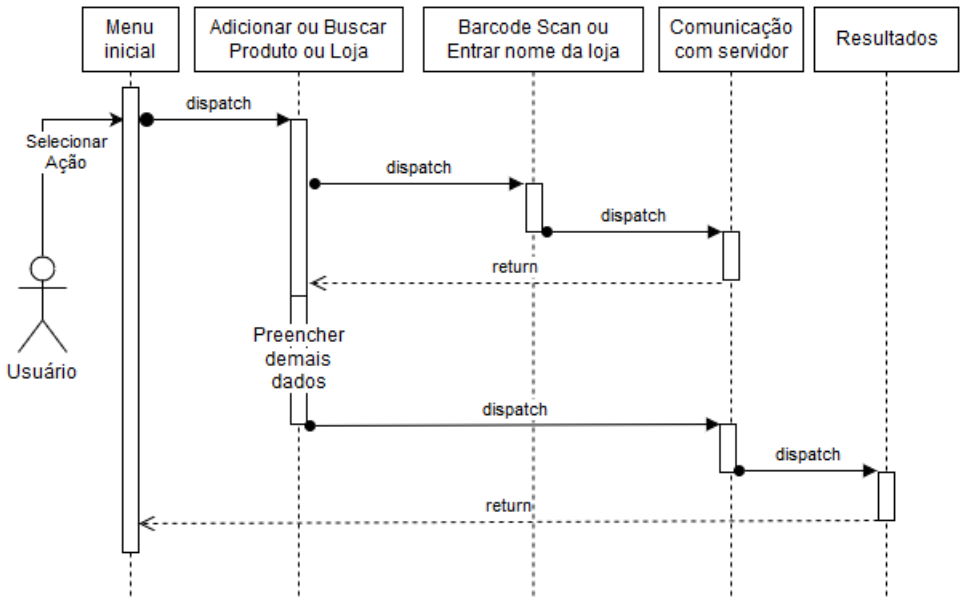


Figura 4.7: Sequência de execução de uma função do aplicativo

Sequência de funcionamento

Para o cumprimento de qualquer uma das tarefas descritas na seção anterior, foi planejada uma sequência específica de execução de suas partes como mostrado na figura 4.7.

Primeiramente, a partir do menu inicial, o usuário seleciona uma das funções entre adicionar produto, buscar produto, adicionar loja ou buscar loja. Seja qual for essa função, na etapa seguinte é necessário que o usuário preencha alguns dados como código de barras e preço no caso de uma nova entrada, ou nome do produto no caso de uma busca. Quando há a necessidade de preencher o código de barras, o usuário tem a opção de usar o recurso da câmera do celular para escaneá-lo. Assim, ao pressionar um botão ele lança uma nova aplicação que, assim que identifica um código válido, retorna à aplicação anterior.

Contudo, antes de retornar para que o usuário verifique ou preencha demais dados, o aplicativo realiza uma busca na base de dados para verificar se há informações que já estão contidas e preenche automaticamente os campos correspondentes.

Depois o usuário tem a chance de alterar algum desses dados antes de pressionar o botão para enviá-los. Asssim, os dados são enviados ao servidor e uma tela de resultados aparece. Ela contém informações sobre os produtos ou lojas no caso de ter sido realizado uma busca, ou uma mensagem dizendo se os dados foram inseridos com sucesso no caso de uma nova entrada.

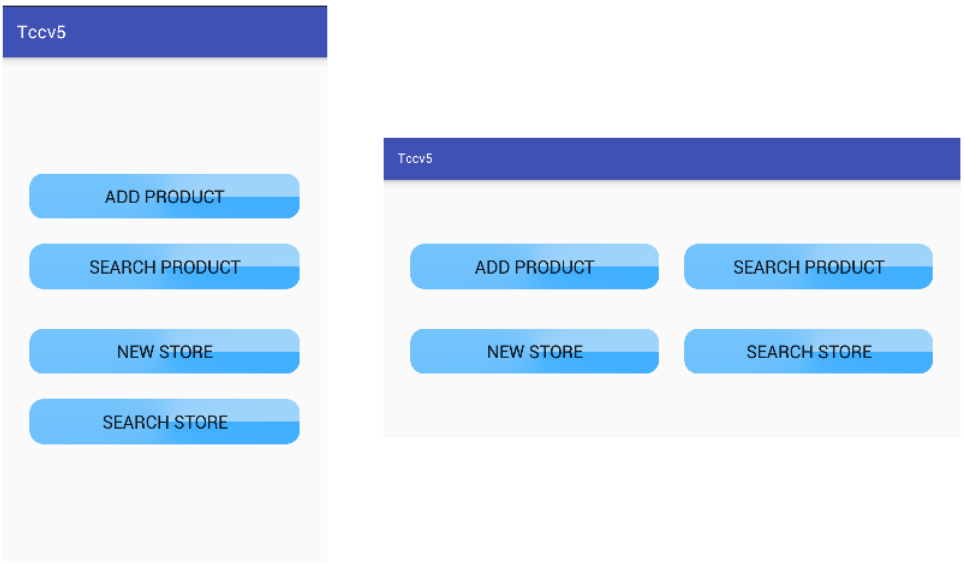


Figura 4.8: Diferentes *layouts* para o mesmo menu em orientações diferentes

Recursos

Uma das principais funcionalidades do aplicativo é o uso da câmera para escanear os códigos de barras dos produtos. Para isso foram adicionados ao projeto arquivos contendo código aberto que fazem interfaceamento com outro aplicativo especializado na tarefa de escanear códigos de barras. Caso esse aplicativo não esteja instalado no aparelho, ao se tentar usar o recurso uma mensagem pop-up aparece e requisita a sua instalação.

Outro recurso são diferentes *layouts* para diferentes orientações de tela. O menu principal, por exemplo, é composto de quatro botões apenas. Com o aparelho na posição vertical, eles são dispostos um sobre o outro verticalmente para aproveitar a dimensão da tela. Contudo, se o usuário usa o modo paisagem, essa disposição é extremamente inapropriada pois os botões ficam excepcionalmente largos na horizontal e finos na vertical. Assim, são criados outros *layouts* para quando esse tipo de orientação é usado, como mostrado na figura 4.8.

4.2.2 Segurança

Nessa etapa do desenvolvimento, aspectos para a segurança do sistema contra ataques externos ou entradas maliciosas não foram abordados. Portanto, o sistema não encontra-se em estado pronto de publicação (às vezes chamado de estado de "produção").

O principal risco é o de um ataque que tenha o intuito de apagar ou desestruturar a base de dados. Essa é uma grande vantagem de se utilizar um sistema terceirizado para o servidor, pois este já conta com algumas medidas e recomendações para evitar acessos indesejáveis.

Outro problema é a entrada de dados maliciosos pelos usuários. Algoritmos específicos devem ser utilizados para se verificar que os valores entrados são ao menos possíveis e aceitáveis. Um preço muito baixo, por exemplo, poderia indicar um erro de digitação ou propaganda enganosa.

Também deve-se garantir que a entrada dos dados não é feita por um computador ("*bot*"). Esses programas são gerados para se criar uma quantidade grande de entradas e sobrecarregar o sistema.

4.2.3 Programação

A programação do sistema também se divide em duas grandes frentes: a programação do aplicativo Android e a programção do website. As seções seguintes descrevem esses procedimentos.

Programação do Aplicativo

O ambiente de desonvolvimento utilizado para todo a programação do aplicativo foi o Android Studio, conforme planejado desde o início do projeto.

No sistema desenvolvido foi usada a organização padrão para os arquivos e recursos do software, descrita na seção "Android Studio"do capítulo "Materiais".

Os conhecimentos e materiais necessários para programação do aplicativo foram obtidos do site do *Android Open Source Project* [3]. Ele contém tutoriais simples e também descrições detalhadas dos recursos disponíveis para programação de um aplicativo.

Quase todo o aplicativo é programado na liguagem Java. Estes arquivos contém a lógica sequencial de execução, uso de recursos do aparelho e deifinição dos *layouts*. Conforme o modelo convencional de aplicações Android, foi usado um arquivo para cada atividade (ou ação) do usuário.

No modelo mais básico, cria-se um respectivo arquivo de *layout* para cada uma dessas atividades. O aplicativo, contudo, conta com o recurso de multiplos *layouts* para a mesma aplicação. Isso é útil, por exemplo, para criar interfaces com o usuário específicas para cada orientação de tela, conforme descrito na seção "Sobre o aplicativo".

Programação do site

No planejamento do site, descrito na seção "Planejamneto do servidor", estão as páginas previstas que o compõe. Elas foram programadas usando-se um editor de texto comum no ambiente Debian-Linux, pois esse foi o primeiro servidor utilizado.

Nesse primeiro desenvolvimento, a estrutra básica das páginas contendo código PHP é mostrada na figura 4.9 em forma de fluxograma. Essa é a aplicação do lado do servidor, que está dentro das páginas Web. Essas páginas web são acessadas por usuários usando navegador Web bem como os que usam o aplicativo, a diferença está somente nas etapas de interação com o usuário mas a sequência de execução se mantém.

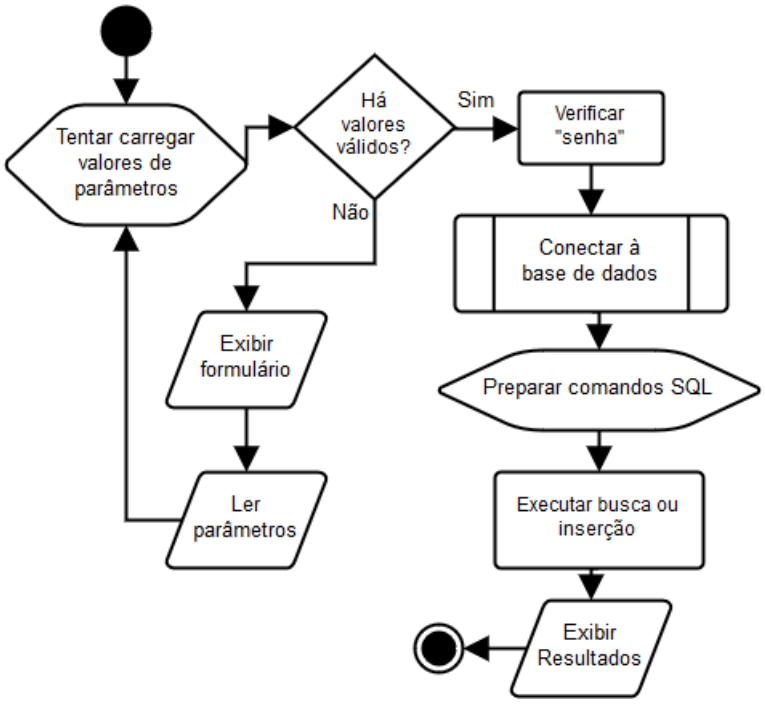


Figura 4.9: Diagrama sequencial da aplicação Web usando PHP

No início de cada arquivo das páginas há um cabeçalho HTML, que contém o tipo da página, tipo de codificação dos caracteres texto, e outros pequenos detelhes gerais sobre o código da página. Há também configurações de *design* e estilos usando a linguagem CSS. Estes compõe o conteúdo estático das páginas.

No corpo da página consta o código PHP que corresponde à aplicação do servidor, ou seja, o responsável pelo conteúdo dinâmico. Cada página tem duas tarefas: receber as entradas do usuário e exibir os resultados correspondentes. Quando uma página é carregada, se verifica se já foram entrados os parâmetros para busca ou inserção de dados; em caso afirmativo,

se executa a tarefa e exibe os resultados; em caso negativo, se exibe um formulário para preenchimento de dados pelo usuário. Junto com a entrada de parâmetros, se envia uma pequena "senha"com o intuito de verificar que as entradas foram geradas usando o formulário apropriado do próprio site e não de algum aplicativo externo.

A única exceção dessa estrutura é a página do menu principal, que exibe apenas os botões que levam às diferentes tarefas disponíveis no site.

Programação *Cloud Computing*

Como já mencionado várias vezes, no decorrer do desenvolvimento o servidor escolhido foi alterado. No novo modelo as páginas tiveram que sofrer umas adaptações, pois a forma de acesso bem como a lógica para armazenar os dados na base de dados é diferente no novo servidor. Isso ocorre porque ele armazena os dados usando o modelo hierárquico, enquanto que o site com PHP e MySQL usa o paradigma relacional.

O conteúdo estático das páginas coninua o mesmo: título, tipo de codificação do texto, estilos, etc.

O conteúdo dinâmico, entretanto, teve de ser adaptado. A primeira mudança ocorre porque a linguagem usada para acessar a base de dados no modelo *Cloud Computing* é Javascript. A sequência lógica de funcionamento, entretanto, é a mesma: primeiro usa-se um *script* para inicializar a conexão com a base de dados, depois prepara-se as instruções para execução e por último essas são executadas. Semelhante ao modelo usado nas páginas PHP, cada página desse modelo também é responsável por receber os dados do usuário e depois exibi-los adequadamente.

4.3 Considerações finais

Ficou claro como durante a execução do projeto planejado se tem uma visão mais ampla das possibilidades de implementação do que quando se planeja o sistema inicialmente. Isso se torna evidente pela alteração do servidor Web simples por um modelo usando *Cloud Computing*, pois as facilidades e recursos disponíveis se tornaram evidentes e atrativas.

Mesmo assim o desenvolvimento na tecnologia deixada de lado se mostrou muito útil. Primeiramente, quando se desenvolveu o sistema nela o conhecimento sobre o sistema em um nível de abstração relativamente elevado, algo entre planejamento e programação, aumentou automaticamente. Este foi muito útil durante a segunda implementação pois como

mencionado nesse capítulo, a sequência de execução (ou lógica de funcionamento) não se altera ao se substituir o tipo de servidor. Além disso, parte do código (conteúdo estático) das páginas foi aproveitado sem alteração, facilitando e acelerando o desenvolvimento do servidor *Cloud Computing*.

Em contrapartida o desenvolvimento na segunda grande frente do projeto, o aplicativo Android, ocorreu usando as ferramentas e métodos planejados. Apesar de durante a execução o entendimento sobre essa programação crescer muito, o uso do Android Studio e da linguagem Java conforme concebidos mostrou-se eficaz e não sofreu alterações.

Capítulo 5

Resultados e Discussões

5.1 Servidor

A principal função do servidor nesse sistema é gerenciar a base de dados e responder às requisições dos usuários. É prevista uma interface para navegadores web (Browser) e uma para usuários do aplicativo android.

No primeiro formato do desenvolvimento do sistema, como detalhadamente explicado na seção de materiais e métodos, foi previsto que houvessem páginas web para responder tanto às requisições feitas através do aplicativo como páginas que seriam adequadas ao acesso usando navegador web. Esse formato foi alterado no caso do acesso usando aplicativo, pois devido a escolha de um servidor adaptado para desenvolvimento de aplicações android as páginas se tornaram desnecessárias nesse contexto.

5.1.1 Páginas Web

Para acesso usando-se um navegador web, o sistema é apresentado através de páginas web.

Como elas compreendem as mesmas funções do aplicativo, foi criado uma página web equivalente para cada atividade da aplicação android. Elas possuem os mesmos campos e a mesma estrutura para se navegar que o aplicativo, pois o objetivo é que o usuário perceba que está acessando o mesmo sistema. Com esse intuito detalhes no design também foram adequados, como pode ser visto na figura 5.1.

Figura 5.1: Website

5.1.2 Base de dados

A base de bados foi projetada contendo três tabelas: entradas, produtos e lojas. Esse sistema foi desenvolvido e provou-se eficiente, mas devido a possibilidade do uso de um servidor adequado a aplicações android foi abandonado.

O novo servidor permite um acesso direto para aplicações android, "hosting"das páginas web e escalabilidade com integração à "Google Cloud". Esses fatores causaram a alteração do modelo planejado.

5.2 Aplicativo

A programação da aplicação Android ocorreu muito parecida com o planejado. Trata-se de um menu principal (figura 5.2) a partir do qual se seleciona uma entre as quatro tarefas possíveis.

Essas atividades, chamadas de atividades "menu"no capítulo anterior, recebem as entradas do usuário. Os campos necessários dependem da função a ser executada, como detalhado a seguir, e contam com a ajuda das atividades "auxiliares"para o preenchimento dos campos que não é esperado a entrada manual pelo usuário (código de barras e código da loja).

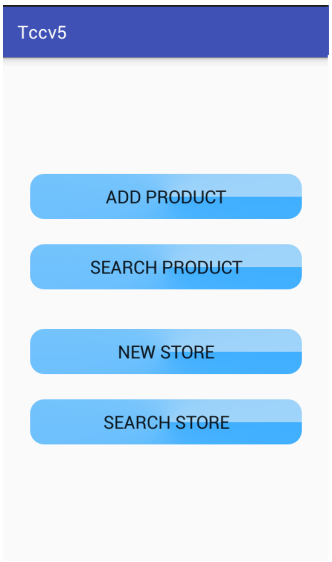


Figura 5.2: Menu inicial

5.2.1 Adicionar Produto

Essa interface é onde o usuário entra com os dados para adicionar uma entrada a base de dados. Para fazê-lo, no mínimo os campos código do produto e código da loja tem que ser preenchidos. Para preencher o código do produto, há um botão com o nome de "scan"que inicia uma outra interface para escanear o código do produto. Quando um número é lido com sucesso, essa outra atividade retorna o valor para a primeira automaticamente, como detalhado na seção sobre o scan de CDB. De maneira semelhante ocorre o preenchimento do código da loja, pois não é esperado do usuário seu preenchimento de maneira manual. Na versão final esse código não será visível ao usuário, sendo usado somente como referência para o funcionamento do aplicativo. Em contrapartida, os campos "nome do produto"e "comentário"são preenchidos pelo usuário. A princípio eles são opcionais, pois se o produto já constar na base de dados, são automaticamente preenchidos. Caso não sejam preenchidos e as informações não estarem contidas na bse de dados, um erro é retornado ao usuário pois a entrada se torna necessária. A figura 5.3 mostra a atividade e seus campos, antes do preenchimento pelo usuário.

5.2.2 Busca produto e resultado

Essas duas atividades têm, juntas, a função de permitir ao usuário buscar um produto na base de dados. A primeira, "Busca produto"é muito semelhante à "adicionar produto"no sentido de que é onde o usuário entra com os dados a serem buscados: nome ou código de

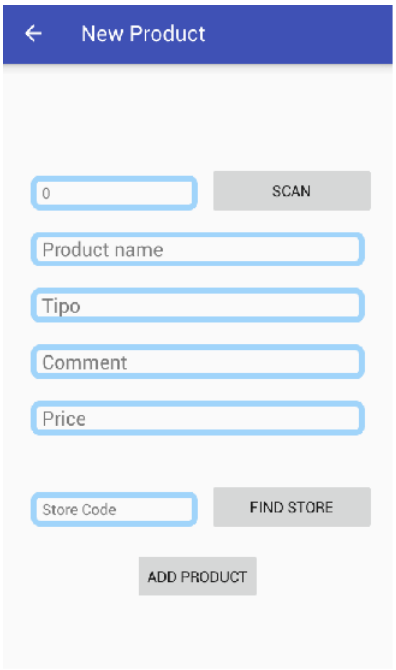


Figura 5.3: Atividade para se adicionar produto vazia

barras. Da mesma maneira que na atividade para adicionar o produto, para preenchimento do CDB o botão "scan"lança o recurso da câmera para preenchimento do campo. Depois de preenchidos os dados o usuário pressiona o botão "Buscar", disparando uma segunda atividade que se comunica com o servidor e exibe os resultados na tela. A seguir são expostos os dois estados, a esquerda antes de buscar um produto e a direita os resultados obtidos.

5.2.3 CDB Scan

Esse estado ou atividade do aplicativo é classificada como sendo "auxiliar", como explicado anteriormente. Ela é responsável por escanear o código de barras dos produtos e retornar automaticamente assim que conseguir registrar um valor corretamente. O código para execução dessa tarefa foi obtido de um repositório github, que foi então incorporado à aplicação [28] . Na primeira figura se demonstra como a atividade busca um código de barras completo e na segunda vê-se essa identificação, pouco antes de retornar o valor lido.

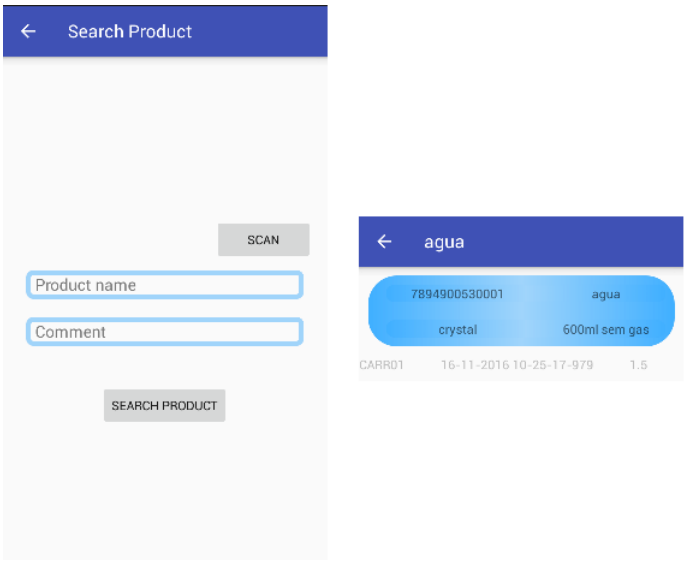


Figura 5.4: Interface para se buscar um produto: campos de busca (esquerda) e resultados (direita)

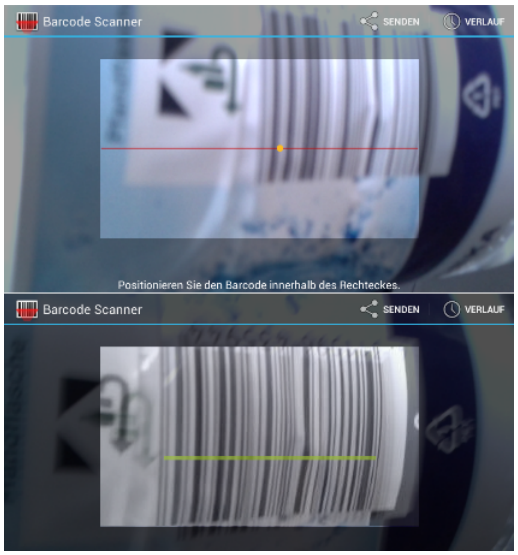


Figura 5.5: Escanear código de barras

5.2.4 Adicionar Loja

Outra possível entrada de dados é quando o usuário deseja cadastrar ou editar os dados de uma loja na base de dados. Selecionando essa opção deve-se preencher o nome, endereço, cidade e coordenadas geográficas da loja. Essa atividade é mostrada na figura 5.6:

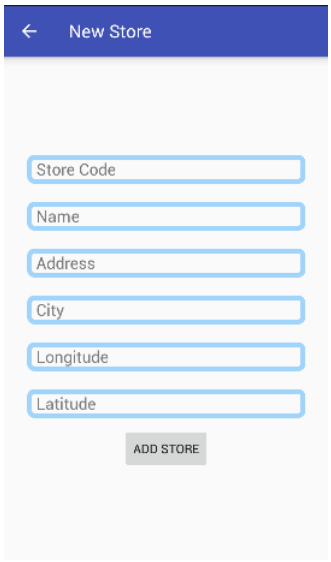


Figura 5.6: Atividade para adicionar loja

5.2.5 Buscar Loja e resultados-lojas

Quando é necessário entrar com o código da loja, há sempre o botão "lojas"para que a atividade "Busca loja"seja iniciada. Ela também é "auxiliar"e tem como função ser a entrada dos dados para que o usuário possa localizar uma loja cadastrada através de seu nome ou endereço, como exemplificado na parte esquerda da figura a seguir. Quando esses dados são entrados e se pressiona o botão "buscar", inicia-se a atividade "resultado lojas"que busca na base de dados as lojas correspondentes com a busca do usuário e retorna uma lista, como na parte direita da figura a seguir. O usuário então seleciona nessa lista a loja correta, e seu valor é retornado para a atividade onde o código da loja é requerido.

5.3 Teste: Tamanho da base de dados

Nessa seção é descrito o teste que foi realizado com o intuito de medir e prever o tamanho que a base de dados ocupa para diferentes quantidades de produtos e lojas.

5.3.1 Teste

O teste realizado se resume a adicionar novoas entradas a base de dados e verificar-se o quanto ela cresceu no espaço de memória ocupado em bytes.

Para medir esse crescimento é necessário levar-se em consideração cada diferente tipo de nova entrada. Há quatro possibilidades: o cadastro de uma nova loja; o cadastro de um novo

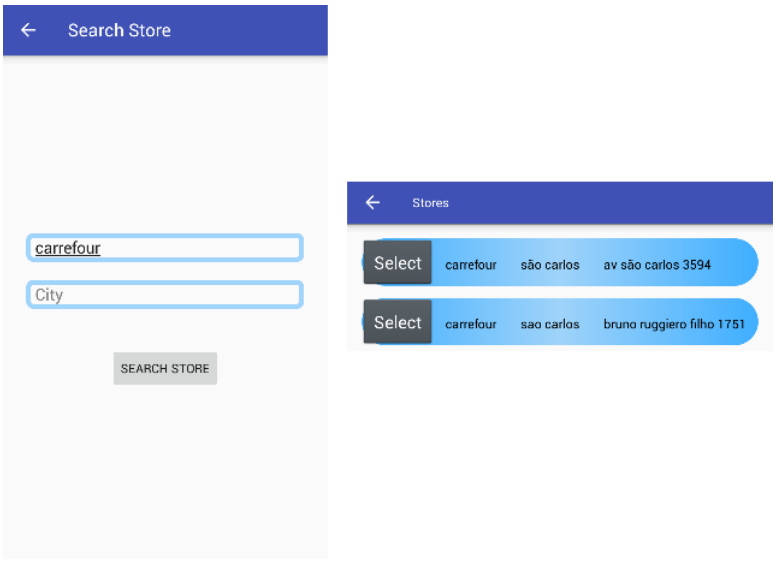
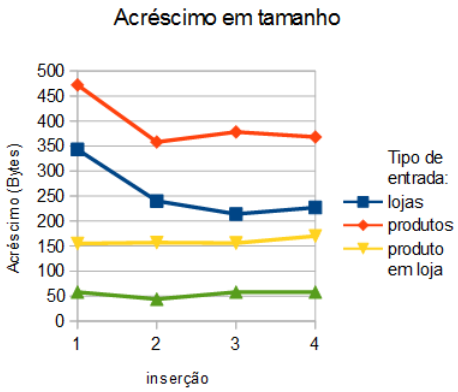


Figura 5.7: Atividade de busca de lojas e seus resultados



Acréscimo em bytes

entradas	lojas	produtos	produto em loja	Preço
1	343	472	155	58
2	240	358	157	44
3	214	378	156	58
4	227	368	170	58

Figura 5.8: Aumento do uso de memória pela base de dados

produto em uma loja; o cadastro de um produto existente mas em uma loja onde este ainda não estava cadastrado; uma nova entrada de preço para um produto loja já cadastrado em uma loja.

Assim, foram efetuadas quatro adições de cada tipo e o padrão de crescimento ficou claro, como pode ser visto na figura 5.8.

Se nota claramente que, além do caso da inserção da primeira loja e do primeiro produto, a cada nova entrada a base de dados cresce numa taxa relativamente constante levando-se en consideração o tipo de entrada.

5.3.2 Previsão

Usando os dados dos testes é possível se calcular qual seria o espaço em disco utilizado pela base de dados.

Como as taxas de crescimento são constantes para cada tipo de inserção, uma boa aproximação é usar as médias como incremento a cada nova entrada. Os valores aproximados são: 260 bytes a cada nova loja; 390 bytes a cada novo produto; 160 bytes a cada produto existente em nova loja; 50 bytes para cada nova entrada de preço.

Assim, considerando uma estrutura mostrada na figura 5.9 de L lojas, cada uma com uma média de P produtos cadastrados e cada produto com uma média de E entradas de preços salvos na base de dados, é possível se deduzir uma fórmula genérica para o cálculo do tamanho da base de dados, mostrada em 5.10.

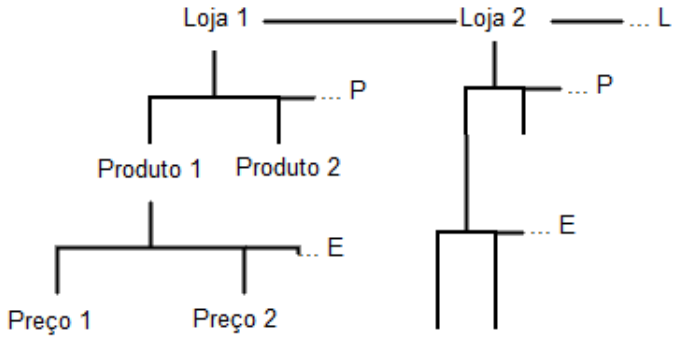


Figura 5.9: Árvore mostrando lojas, produtos e entradas arbitrárias cadastradas

$$Tamanho = L * l + (L - 1)p' + P * p + L * P * E * e$$

Figura 5.10: Equação para cálculo do tamanho da base de dados

Com L sendo o número de lojas, P de produtos, E de entradas, l o tamanho ocupado pela inserção de uma nova loja, p o tamanho da inserção de um novo produto, p' o tamanho ocupado pela inserção de um produto em uma nova loja e e o espaço ocupado por uma nova entrada de preço.

Usa-se então essa equação para prever o tamanho da base de dados necessária para o uso do sistema numa cidade como São Carlos. O sistema poderia ser composto, por exemplo, pelo cadastro de 20 lojas, cada uma com em média 100 produtos cadastrados e cada produto com três entradas diferentes de preços. Assim, o sistema ocuparia aproximadamente 340kB

em disco, resultando em uma média de 58 bytes por entrada.

Capítulo 6

Conclusões

O objetivo proposto foi alcançado, ou seja, foi desenvolvido um sistema para busca dos melhores preços dos produtos do varejo usando o código de barras, sendo acessível via aplicativo Android e navegador Web. A maior dificuldade no desenvolvimento foi o interfaceamento entre os sistemas: entre o aplicativo e o servidor, entre o *website* e a base de dados, etc. No desenvolvimento desses componentes individuais o desafio foi o uso das diferentes linguagens de programação, pois o desenvolvedor não possuía conhecimento prévio na maioria delas.

Como se trata de desenvolvimento de software, é difícil se considerar a solução como "pronta". Sempre surgem novas tecnologias que podem ser incorporadas, ou novas ideias de funcionalidades. É por isso que se o sistema vir a ser disponibilizado ao público é necessário um serviço de manutenção, sempre ligado a novas tendências e possibilidades.

Há também algumas limitações intrínsecas da solução desenvolvida que poderiam ser repensadas. Uma delas é que só é possível cadastrar um produto se ele possuir um código de barras. Outra limitação semelhante é a forma de busca dos produtos, que ocorre somente através do código de barras. Além disso, a busca só é efetuada para um produto de cada vez, fazendo com que o sistema seja útil apenas para verificação de alguns produtos por compra, pois é impraticável escanear-se os códigos de barras de todos itens a serem comprados.

Há também funcionalidades que devem ser incluídas antes do lançamento do sistema, antes da chamada "etapa de produção". Uma necessidade é criar um incentivo para que os usuários enviem registros à base de dados como por exemplo o acúmulo de pontos e categorias de usuários com diferentes privilégios. Também se incluiría penalidades para a entrada de dados errôneos, com o intuito de manter as informações da base corretas. Manter a conformidade da base de dados é tão importante que mais métodos devem ser implementados

nesse sentido: pode-se usar o código mac do aparelho para bloquear um usuário mal intencionado ou fazer a leitura do preço a partir de uma foto da etiqueta e algoritmos de visão computacional, etc.

Antes da produção seria ainda usual uma etapa de validação. Ela pode consistir da criação de um plano de testes para as diferentes funcionalidades do sistema e a realização desses testes. Nela, devem ser testados diferentes modelos de *Smartphones* e resoluções de câmeras.

A ideia desse sistema é coletar uma enorme quantidade de dados (*Big Data*). A estrutura utilizada permite várias análises dos dados coletados, como fluxo, crescimento, etc. Permite também a inclusão no sistema na chamada *Google Cloud*, que disponibiliza uma infinidade de recursos nesse sentido. Assim, apesar do sistema ainda não estar preparado para fornecer análises complexas dos dados obtidos, têm alta escalabilidade e é planejado de tal sorte que seja possível fazer várias análises no futuro. A união desses dois conceitos, *Big Data* e *Cloud Computing*, mostra-se uma tendência muito forte na atualidade.

O sistema contribui, assim, para a chamada "cidade inteligente". Trata-se do uso eficiente dos recursos de energia, serviços e materiais para melhoria da qualidade de vida dos cidadãos. Esse projeto contribuiria para o consumo eficiente e consciente, fazendo a livre concorrência mais justa no contexto dos produtos do varejo.

Além disso, numa etapa mais avançada, o monitoramento dos preços em função do tempo poderia identificar práticas proibidas no mercado, como por exemplo o chamado *dumping* onde uma loja já estabelecida vende produtos abaixo do preço de custo para eliminar concorrentes mais novos no mercado.

Trabalhos futuros

Com a quantidade de dados prevista que esse sistema coletará, é muito fácil imaginar diversas funcionalidades. Essa generalização do ramo de atuação é exatamente um ponto forte do projeto.

A principal limitação é que os produtos tenham cóigo de barras. Assim, um próximo desenvolvimento seria incluir de alguma maneira os produtos do varejo que não têm CDB, como por exemplo produtos de padarias, produtos artesenais, etc.

Outra grande limitação é que o aplicativo só pode ser instalado em aparelhos que usem o sistema operacional Android. Sendo assim, um próximo passo seria o desenvolvimento de um aplicativo igual para iOS e windows phone.

Uma outra funcionalidade interessante seria a de identificar os produtos buscados pelos usuários e oferecer produtos semelhantes. Esse processo poderia até ser influenciado por algoritmos que levam em conta patrocínio de empresas.

Outra possibilidade é a criação de uma lista de compras que geraria uma sugestão de local de compra, verificando onde há todos os produtos e onde estes têm o melhor preço. Essa lista poderia fazer uma comparação das difrentes lojas da região, considerar inclusive a distância dessas até o usuário e se elas estão abertas no horário da busca.

Em suma, são diversas as possibilidades. Esse é precisamente o intuito de aglomerar-se grandes quantidades de dados: analisá-los das mais diversas formas possíveis.

Referências

[1] Douglas C. Schmidt James C. HU, Irfan Pyaralii. Measuring the impact of event dispatching and concurrency models on web server performance over high-speed networks. IEEE, 1997.

[2] Vicenc. Beltran. Jordi Torres. Eduard Ayguade. Understanding tuning complexity in multithreaded and hybrid web servers. IEEE, 2008.

[3] Android Open Source Project. Android activities. https://developer.android.com/reference/android/app/Activity.html.

[4] Android Open Source Project. App structure. https://developer.android.com/design/patterns/app-structure.html.

[5] Android Open Source Project. Up navigation. https://developer.android.com/design/patterns/navigation.html.

[6] Android Open Source Project. Descedant and lateral navigation. https://developer.android.com/training/design-navigation/descendant-lateral.html.

[7] Android Open Source Project. Layouts. https://developer.android.com/guide/topics/ui/declaring-layout.html.

[8] Android Open Source Project. Screen support. https://developer.android.com/guide/practices/screens_support.html.

[9] Android Open Source Project. Acesso a internet eficiente. https://developer.android.com/training/efficient-downloads/efficient-network-access.html.

[10] Comparative of linux distribuitions. http://distrowatch.com/dwres.php?resource=popularity.

[11] Web server performance comparison. https://help.dreamhost.com/hc/en-us/articles/215945987-Web-server-performance-comparison.

[12] w3schools. Php connect to mysql. http://www.w3schools.com/php/php_mysql_connect.asp.

[13] firebase. App success made simple. https://firebase.google.com/.

[14] Redacao. Buscape vai incorporar 13 sites de comparacao de produtos da europa e africa. Veja, 2014.

[15] Roberto Minerva. Abyi Biru. Domenico Rotondi. Towards a definition of the internet of things (iot). IEEE, 2015.

[16] Alberto Messias da costa souza. Uma nova arquitetura para internet das coisas com analise e reconhecimento de padroes e processamento com big data. USP Escola Politecnica, 2015.

[17] Nicolas Bettenburg. Ahmed E Hassan. Bram Adams. Daniel M. German. Management of community contributions: A case study on the android and linux software ecosystems. Springer Science, 2013.

[18] Sergio Ilarri Carla Katarina M. Marques and Giovanni C. Barroso Jose Merseguer. Performance analysis of a dynamic architecture for reconfiguration of web servers clusters. IEEE, 2010.

[19] Martin Pollakowski. Grundkurs mysql und php. 2 ed., 2005.

[20] Alfons Kemper e AndrÃ© Eickler. Datenbanksysteme. 7 ed, 2009.

[21] Alfred Moos. Datenbank engineering. 3 ed., 2004.

[22] Linda Dailey Paulson. Developers shift to dynamic programming languages. IEEE, 2017.

[23] Mozilla Developer JSX. About javascript. https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript.

[24] Kai-Uwe Sattler e Andreas Heuer Gunter Saake. Datenbanken konzepte und sprachen. 3 ed., 2008.

[25] Android Open Source Project. About google play. https://developer.android.com/distribute/googleplay/about.html.

[26] Android Open Source Project. Distribuicao. https://developer.android.com/distribute/essentials/quality/billions.html.

[27] About manifest. https://developer.android.com/guide/topics/manifest/manifest-intro.html.

[28] Zxing android embedded, journey apps. https://github.com/journeyapps/zxing-android-embedded.

Capítulo 7

Apêndice: Fluxogramas do código do aplicativo

Aqui são expressos os fluxogramas que representam as diferentes atividades explicitadas no texto desse trabalho. São eles:

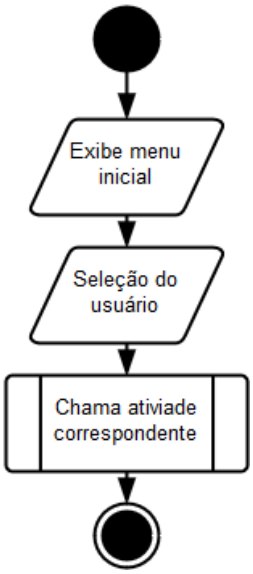


Figura 7.1: Atividade do menu inical

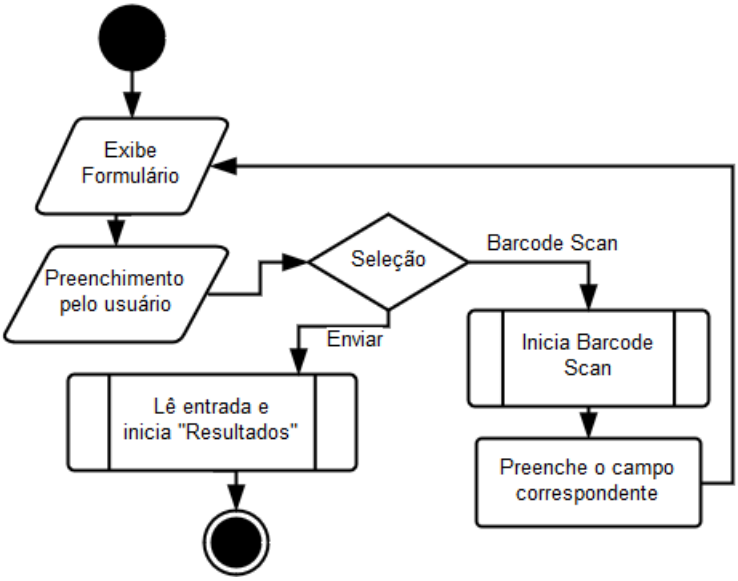


Figura 7.2: Atividade para adicionar novos produtos

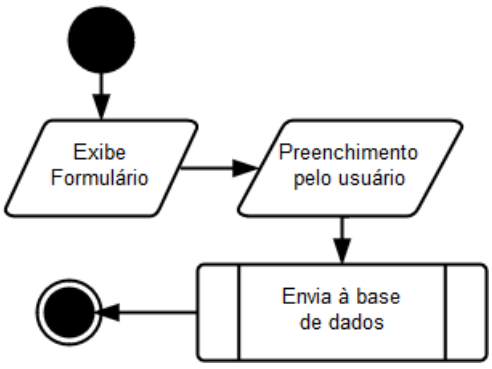


Figura 7.3: Atividade para adicionar novas lojas

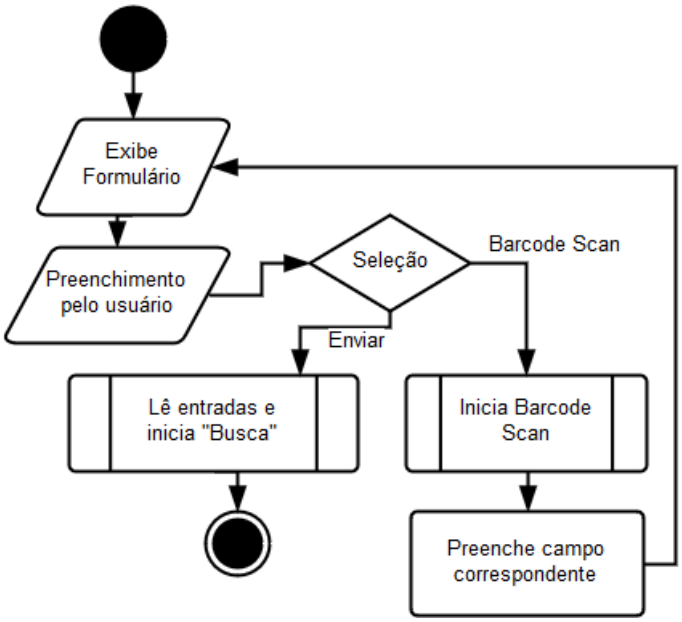


Figura 7.4: Atividade para buscar produtos

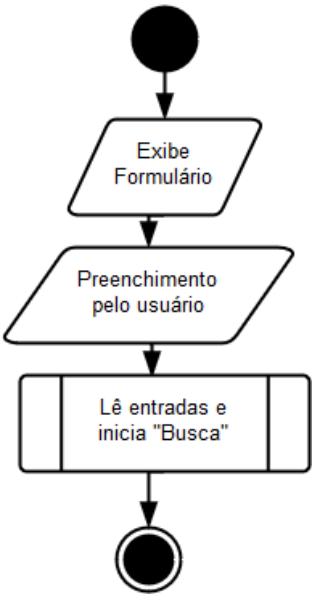


Figura 7.5: Atividade para buscar lojas

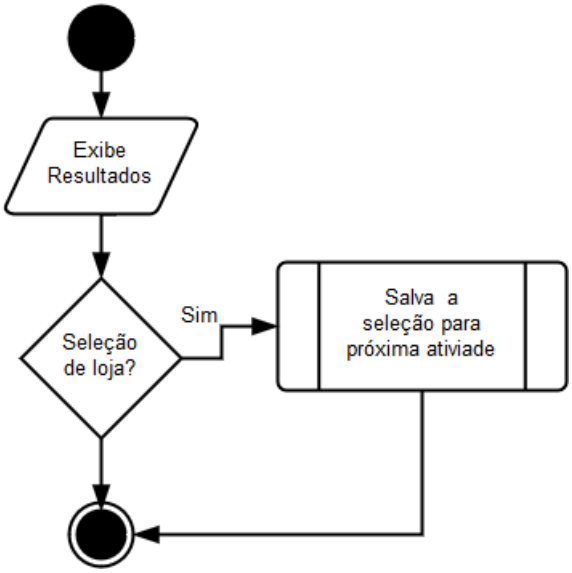


Figura 7.6: Atividade para exibir o resultado da busca de lojas

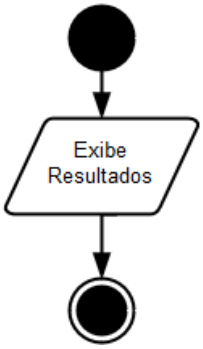


Figura 7.7: Atividade para exibir o resultado da busca de produtos