Curso de Ciência da Computação
Universidade Estadual de Mato Grosso do Sul

Aplicação Android para divulgar produtos em promoção

Alyson Alexandre Boone
Klaus Korte

Dr. Nilton Cézar de Paula (Orientador)

Curso de Ciência da Computação
Universidade Estadual de Mato Grosso do Sul

Dourados - MS
2013

Aplicação Android para divulgar produtos em promoção

Alyson Alexandre Boone
Klaus Korte

Monografia da disciplina Projeto Final de Curso, apresentada ao Curso de Ciência da Computação da Universidade Estadual de Mato Grosso do Sul, como parte dos requisitos para obtenção do título de Bacharel em Ciência da Computação.

Dr. Nilton Cézar de Paula (Orientador)

Dourados - MS
2013

Resumo

Com o avanço dos celulares nas últimas décadas, e com o aumento dos estabelecimentos comerciais, este trabalho apresenta a concepção e o desenvolvimento de uma aplicação Android para promover a divulgação de produtos em oferta. Visto que há a necessidade da sociedade ter aplicativos simples que visem gerar uma economia para o próprio usuário e também que estimule a concorrência entre os estabelecimentos comerciais.

Palavras-chave: *Android, ofertas comerciais, celular.*

Abstract

With the advancement of mobile phones in recent decades, and the rise of the shops, this paper presents the design and development of an Android application to promote the dissemination of products on offer. Since there is the need for society to have simple applications that aim to generate savings for the user himself and also stimulate competition between merchants.

Keywords: *Android, offers commercial cell.*

Sumário

Lista de Siglas

ADB – Android Debug Bridge
API – Application Programming Interface
APK – Android Package
ARM – Advanced RISC Machine
AVD – Android Virtual Device
CPU – Central Processing Unit
GPS – Global Positioning System
IPC – Inter-process Communication
RAM – Random Access Memory
SDK – Software Development Kit
SQL – Structured Query Language
TCP – Transmission Control Protocol
UDP – User Datagram Protocol

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

A divulgação de produtos sempre foi algo muito importante na venda de produtos no comercio e o grande alvo das divulgações são sempre os clientes.

Hoje em dia com o avanço da tecnologia as propagandas ganharam novos meio de divulgação como: e-mails, sites de propaganda, mensagens de texto, programas para divulgação, entre outros.

Com esse aumento das propagandas os comércios aumentaram mais as suas vendas e o cliente ganhou mais um meio para economizar o seu dinheiro e comparar quais são os estabelecimentos comerciais que tem os melhores produtos.

A proposta deste trabalho é apresentar um protótipo para que os estabelecimentos comerciais façam a divulgação dos seus produtos.

1.1 Objetivos

Este trabalho tem como objetivo geral desenvolver um protótipo para a divulgação de produtos que estão em promoção usando a plataforma Android.

1.1.1 Objetivos específicos

Pretende-se desenvolver um aplicativo que informe ao cliente os pontos comerciais que terão promoções e a localização destes estabelecimentos.

O cliente acessa o aplicativo e ao informar uma determinada região o aplicativo mostra os produtos que estão em oferta naquela região.

Para isso devemos realizar as seguintes atividades:

- Estudo da plataforma Android para suporte ao desenvolvimento do protótipo.
- Modelagem lógico e físico do prototipo usando diagramas.
- Explicação dos passos para o uso do protótipo.
- Implementação de uma aplicação móvel para validar a proposta.

1.2 Justificativa e motivação

Com o aumento do uso de celulares e facilidades de acesso à internet as pessoas necessitam de novos aplicativos que as ajudam a economizar tempo e dinheiro. Assim, houve a ideia de desenvolver um aplicativo na plataforma Android para ajudar as pessoas a economizar dinheiro e poupar tempo. Outro motivo foi a grande oportunidade dos estabelecimentos fazerem a sua propaganda e com isto aumentar a visibilidade da loja e também aumentar a concorrência.

1.3 Metodologia

Uma das propostas do trabalho é realizar o estudo da plataforma Android, bem como a sua arquitetura, funcionamento e desenvolvimento de aplicações.

Para o funcionamento do aplicativo é utilizado socket, um mecanismo que permite a troca de informação pela internet entre o aplicativo do comerciante e o aplicativo do cliente.

1.4 Organização do texto

Capitulo 1 - Introdução

Neste capitulo foram apresentados a introdução deste trabalho bem como os seu objetivos gerais e específicos, a justificativa e como será o metodologia do trabalho.

Capitulo 2 - Tecnologias abordadas

Neste capítulo iremos abordar e explicar quais serão as tecnologias que serão necessárias para o desenvolvimento do programa, tais como: Android, programa de posicionamento pessoal GPS, banco de dados nativo do Android, o SQLite, e o detalhamento dos componentes necessários para fazer o teste do aplicativo.

Capitulo 3 – Projeto do protótipo de divulgação de produtos em promoção

Neste capítulo iremos mostrar os resultados, os diagrama de caso de uso dos aplicativos, as telas dos aplicativos, os requisitos mínimos no qual os aparelhos portáteis tem que ter para rodar o aplicativo. Será abordado como tema principal o sistema de promoções com detalhes das principais classes da proposta..

Capitulo 4 - Considerações finais

No capitulo 4, iremos abordar quais serão as dificuldades que poderão ocorrer devido a falta de preparo dos usuários.

Apêndice A

No apêndice A, iremos mostrar o manual do usuário, como serão os passos para manusear sendo bem demostrado com telas autodidáticas.

Apêndice B

No apêndice B, iremos mostrar os programas utilizados para o desenvolvimento do aplicativo, bem como os passos da instalação dos programas.

Apêndice C

No apêndice C, iremos mostrar o código do aplicativo, mostrar os códigos das classes que pertencem ao aplicativo do comerciante e os códigos das classes que pertencem ao aplicativo do cliente.

Capítulo 2

Tecnologias abordadas

Neste capítulo iremos abordar e explicar quais são as tecnologias abordadas utilizadas para desenvolver o programa de ofertas. Vamos abordar temas como: Android, os seus componentes que são necessários para implementar uma aplicação e testá-la, outros complementos que nos ajudarão a coletar dados para o bom funcionamento do programa.

2.1 Android

O Android é uma plataforma para tecnologia móvel completa, envolvendo um pacote com programas para celulares, já com um sistema operacional, middleware, aplicativos e interface do usuário. [LECHETA, 2013].

O Android foi criado para que os desenvolvedores possam fazer aplicações móveis que tirem total proveito que o aparelho móvel possa oferece. Foi construído para ser verdadeiramente aberto, sendo assim qualquer pode adaptá-lo a fim de incorporar novas tecnologias. [LECHETA, 2013].

A plataforma Android foi desenvolvida com base no sistema operacional Linux e é composta por um conjunto de ferramentas que atua em todas as fases do desenvolvimento do projeto, desde a execução até a criação de *softwares* específicos. [LECHETA, 2013].

O Android tem a capacidade de segurança muito forte, onde não há conflito entre os dados de dois aplicativos diferentes. Como é executado em um *kernel* do Linux, toda a vez que um aplicativo é instalado no Android, é criado um novo usuário Linux

para aquele programa, com diretórios que serão usados pelo aplicativo, mas somente para aquele usuário. [LECHETA, 2013].

Se uma aplicação precisa de alguma recurso do sistema a mesma deve estar declarada no AndroidManifest.xml. Cada processo só pode executar uma determinada função se a mesma está declarada a permissão declarada, isto funciona assim pois, as permissões já possa ser conhecido desde o momento da instalação, e nada poderá ser modificado após isso. [LECHETA, 2013].

2.2 Arquitetura do Android

A arquitetura do Android é dividida em 4 camadas como na **Figura 2.1,** onde o nível mais baixo é encontrado o *kernel* e os *drives* para os recursos do dispositivo móvel, mais acima vem o *framework* que tem como objetivo de ser um link com a camada de bibliotecas do sistema operacional e das APIs contidas no framework. Uma breve descrição será mostrada abaixo:

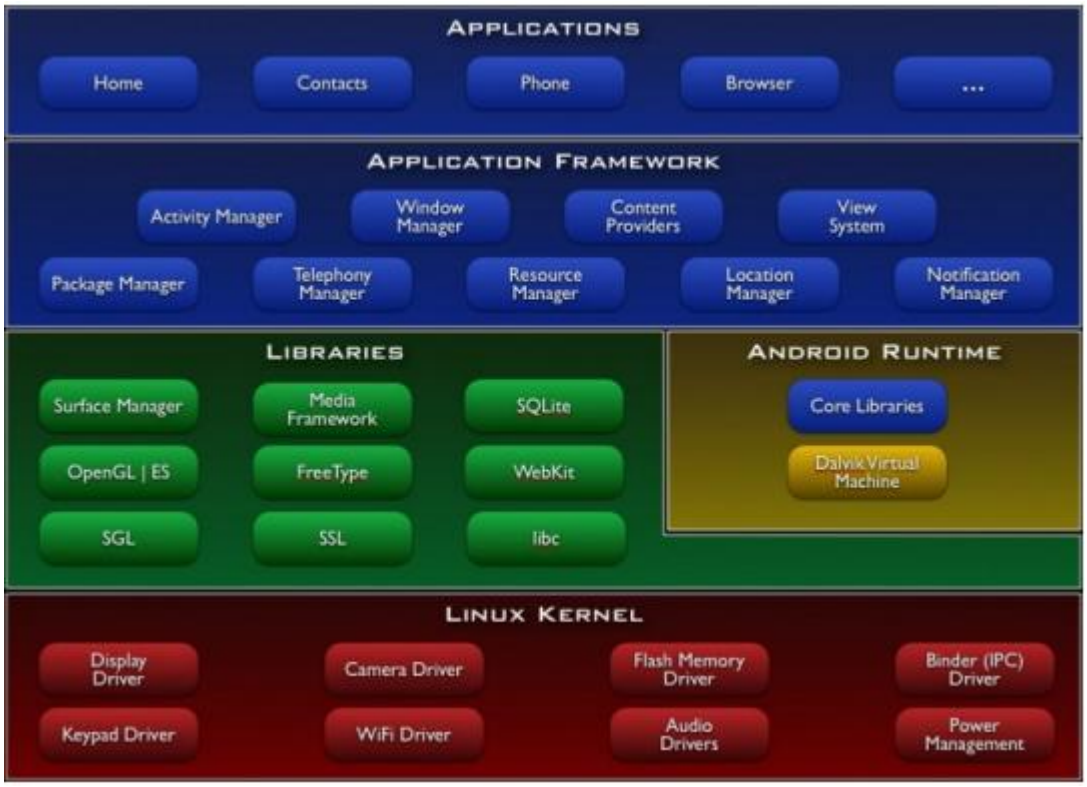


Figura 2.1. Arquitetura Android (SILVA. 2010, p.9).

Aplicações: acima de todas as camadas está a camada de aplicativos onde se encontra todos os aplicativos fundamentais do Android.

Framework: nesta camada encontramos todas as APIs e os recursos utilizados pelos aplicativos, como classes visuais, que incluem listas, grades, caixas de texto, botões e até um navegador embutido *View system*(componentes utilizados na construção de aplicativos) e provedor de conteúdo, que possibilita uma aplicação acessar dados de outras aplicações, ou até compartilharem informações.[LECHETA, 2013].

Entre os principais elementos desta camada temos:

- *Activity Manager*: gerencia as atividade de todas as *activities*, quando iniciar, quando terminar, quando chamar a próxima e assim por diante;

- *Package Manager*: é utilizada pelo *activity manager* para ler as informações do APKs. Ele se comunica com o resto do sistema e diz quais os pacotes que estão sendo usados e quais as capacidades desses pacotes;

- Window Manager: gerencia as apresentações das janelas, qual a janela que está ativa e assim por diante;

- *Content Porviders*: possibilita a troca dados entre os aparelhos e o compartilhamento de dados entre os aplicativos;

- *View System*: disponibiliza todo o tratamento gráfico para os aplicativos;

Esta camada foi criada com a função de simplificar a reutilização dos componentes.

Bibliotecas: esta camada carrega um conjunto de bibliotecas C/C++ utilizadas pelo sistema, incluindo ainda aquelas bibliotecas que servem como apoio as áreas de multimídia, funções para navegadores *web*, funções gráficas, funções de aceleração de *hardware* e funções de acesso ao banco SQLite. [LECHETA, 2013].

Android Runtime: a pequena camada do **ambiente de execução** é uma instância da máquina virtual *Dalvik.* A *Dalvik* é uma máquina virtual com melhor desempenho, maior integração com a nova geração de *hardware* e projetada para executar varias maquinas virtuais pararelamente. Foi criada para funcionar em sistemas com baixa CPU, pouca memória RAM onde o sistema operacional não disponibiliza *swap*.

O Dalvik são classes de Java convertidas para máquina virtual através da ferramenta DX, distribuída juntamente com a ferramenta SDK do Android. [LECHETA, 2013].

2.2.1 Linux Kernel

O Android utiliza a versão 2.6 do kernel do Linux para os serviços centrais do sistema, tais como segurança, gestão de memória, gestão de processos, pilha de protocolos de rede e modelo de drives. O *kernel* também serve como uma abstração entre o *hardware* e o resto da pilha de *software*. Um acessório é o *Binder* (IPC) é responsável por obter e enviar para a aplicação requerente a interface de serviço da aplicação requerida, possibilitando assim a comunicação entre os processos [LECHETA, 2013].

Nesta camada também tem um poderoso sistema próprio de energia, que verifica qual aplicativo que está ocupando mais energia e qual está ocupando menos energia assim, desligando o mesmo [LECHETA, 2013].

2.2.2 Elementos da aplicação

O Android possui um grupo de componentes essências que o sistema pode instanciar e chamar qualquer momento quando for necessário, formados pelos seguintes itens:

- Activities;
- Services;
- Broadcast Receivers;

- Content Proveiders;

Todos esse componentes devem ser declarados dentro de um arquivo XML, chamado AndroidManifest.xml, caso não sejam declaradas os componentes não podem ser usado na aplicação.

Dentre esses componentes o principal é o Activity e será apresentado mais detalhes sobre ele a seguir.

Activity ou atividade

É o mais utilizado de todos os componentes, pois é representado por uma tela na aplicação. Possui interface com o usuário por meio de Views, consistindo de varias telas as quais são ligados ao evento que são programados.

Assim que uma atividade é chamada ela é colocada no topo da pilha de atividades e se torna a atividade em execução, se existir uma atividade antes dela, permanecerá antes dessa que foi colocada no topo e só será executada quando a atividade que esta na frente dela sair do topo da pilha de atividades.

São quatro os estados em que uma atividade pode se encontrar:

- **Executando:** é a atividade que está ativa no visor da aplicação;

- **Parada:** uma atividade que perdeu o foco para outra atividade e não está interagindo com o usuário.

- **Interrompida:** caso uma atividade não esteja sendo utilizada pelo usuário, ela mantem as suas informações de estado, porém são muitas vezes terminadas e a sua recuperação de estado tenha se perdido, assim perdendo informações.

- **Finalizada:** o sistema pode finalizar uma atividade se ela esta parada por um determinado tempo ou se interrompida.

2.2.3 AndroidManifest.xml

Toda aplicação deve ter o arquivo AndroidManifest.xml no seu diretório. É um arquivo de configuração que descreve os elementos da aplicação e as classes utilizadas na aplicação.

2.3 Banco de dados do Android

O Android tem um banco de dados nativo chamado SQLite que é leve, isso foi uma grande questão que pesou na escolha desse banco de dados para estar no Android, outra questão é que ele é bem fácil de manusear a sua sintaxe. Na sequência iremos explicar melhor a sua história e a sua arquitetura.

2.3.1 O que é o SQLite.

SQLite é um banco de dados Open Source. SQLite suporta recursos de banco de dados relacionais padrão, como a sintaxe SQL, transações e instruções preparadas, além disso o banco de dados necessita de uma memória limitada em tempo de execução, o que torna um banco que se adaptou muito bem no ambiente Android. Sua estrutura está contida em apenas em um único arquivo no sistema e o acesso aos dados é implementado por uma biblioteca de funções escritas em C. [ANDROID SQLITE, 2013].

Tipos de Campos

No SQLite o tipo de dado está relacionado com o valor propriamente dito, diferente do que os outros bancos de dados, tornando um sistema dinâmico. Um campo da tabela do banco de dados pode receber qualquer tipo de dados ignorando o que está escrito no CREATE TABLE. Veja a seguir quais são os possíveis tipos de dados que o SQLite suporta. [SQLITE NO ANDROID, 2012].

NULL: como em todos os bancos os valores não podem ser Nulos;

INTEGER: inteiro com sinal, armazenado em 1, 2, 3, 4, 6 ou em 8 bytes dependendo da grandeza do valor;

REAL: valor com ponto flutuante com 8 bytes;

TEXT: uma *string* garante ter espaços entre as *strings*.

As funções usadas pelo SQLite para a manipulação dos dados são as seguintes: Select, Insert, Update e Delete.

2.4 Android SDK

Para rodar as nossas aplicações é necessário instalar a API de desenvolvimento e o componente Android SDK. [ANDROID SDK, 2013].

O Android SDK fornece as bibliotecas da API e ferramentas necessárias para construir, rodar e testar um aplicativo Android e também inclui debbuger, bibliotecas e um emulador baseado no QEMU. O Android SDK também suporta versões antigas do Android para possibilitar ao desenvolvedor criar aplicativos para versões mais antigas do Android.

O QEMU é um monitor de máquina virtual hospedada para emular unidades centrais de processamento através da dinâmica de tradução binária e fornece uma variedade de modelos de dispositivos que permite que cada dispositivo emulado execute sem que modifique algum arquivo do sistema operacional nativo.

2.4.1 Android Debug Bridge(ADB)

O Android Debug Bridge (ADB) é uma ferramenta incluída no pacote SDK e é constituída de dois programas, um cliente e um servidor, que comunicam um com o outro que inclui três componentes:

Um cliente, que é uma máquina de desenvolvimento.

Um servidor, que é executado em segundo plano na maquina de desenvolvimento como um processo. O servidor faz a comunicação do cliente entre o *daemon* adb executando na maquina.

O *daemon* que é executado como um processo no emulador.

Quando você inicia um cliente adb, ele primeiro verifica se existe um processo servidor em execução, senão esta em execução, ele inicia o servidor. Quando o servidor é iniciado o cliente se conecta a porta TCP 5037 e utiliza essa porta para se comunicar com o adb.

O servidor logo em seguia estabelece conexão com todas as instancias do emulador em funcionamento. O intervalo de portas utilizadas pelos emuladores para a conexão são as ímpares no intervalo de 5555 a 5585. Segue um exemplo de como será feita o intervalo de portas pelo emulador:

Emulador 1, console: 5554
Emulador 1, adb: 5555
Emulador 2, console: 5556
Emulador 2, adb: 5557
e assim por diante ...

Você pode utilizar o adb para instalar um aplicativo que está sendo desenvolvido no sistema operacional nativo no emulador para que possa ser verificado o seu funcionamento e realizar testes no aplicativo. No caso do Android o adb é responsável em pegar o programa executável (.apk) que a API gerou depois de ter compilado o programa e instala-lo no emulador. [ADB, 2013].

2.5. Android Virtual Device (AVD)

O Android SDK inclui um emulador de dispositivo móvel (um dispositivo móvel virtual no seu computador) para fazer testes sem o uso de dispositivos físicos, e é capaz de emular um dispositivo com CPU ARM (ARM é uma arquitetura de processador de 32 bits e é usado em sistemas embarcados) executando um sistema operacional Android. [ARM, 2013].

Um dispositivo virtual Android (Android Virtual Device ou AVD) é um conjunto de parâmetros que servem para configurar o emulador de modo a utilizar uma imagem específica do sistema Android, definir parâmetros como tamanho da tela, tamanho de memória, e outras características de *hardware* emulado.

2.6 GPS(Global Positioning System)

Para obter uma localização precisa de certo ponto na terra, os aparelhos celulares utilizam dois métodos de localização: o próprio GPS contido no celular e também o sinal de Internet que o mesmo recebe através de antenas WiFi ou o próprio plano de pacote de dados disponibilizado por sua operadora.

2.6.1 Multiplicidade de fontes de localização

A captação da localização pode ocorrer de três maneiras, através do próprio GPS contido no aparelho celular, através de antenas WiFi e também através do plano de pacotes de dados.

O uso do GPS contido no aparelho para se obter a localização é o método mais seguro de se obter uma localização com o máximo de precisão, porém seu sistema pode apresentar falhas ao ser exposto a certos ambientes como ambientes fechados, em dias de céu nublado e também sua resposta não é tão rápida como os usuários necessitam. Há também o impasse da demanda de energia, pois seu uso requer muita carga de energia da bateria.

Outra forma de se obter a localização de certo ponto geográfico é a través do uso de antenas WiFi ou plano de pacote de dados, o qual não possui restrições como ambientes fechados e dias de céu nublado, sendo sistemas que apresentam uma resposta com uma precisão menor mas que são mais eficientes por não consumirem tanta energia. [GPS, 2012].

2.6.2 Janelas de atualização de localização

Uma pequena janela na qual você ira ouvir atualizações de localização significa menos interação com o GPS e serviços de localização de rede, assim, preservando a vida da bateria.

Para aplicações que dependem fundamentalmente da localização do usuário não se pode interromper a busca por atualizações, pois compromete o funcionamento correto da aplicação. Ao contrario de aplicações que não dependem fundamentalmente da localização do usuário, sendo assim pode se pausar a interação com o GPS assim que as coordenadas forem obtidas com sucesso para que haja um melhor aproveitamento da carga da bateria.

2.6.3 Conjunto de provedores

Para obter uma melhora na precisão dos dados pode-se utilizar a mixagem de duas fontes de localização, sendo essa uma estratégia que pode ser aplicada a diversas aplicações que necessitam tanto de velocidade na obtenção de dados como precisão. Técnica na qual tem por característica um grande consumo de energia, por usar dois meios de obtenção de localização ao mesmo tempo [GPS, 2012].

2.7 Socket

Um recurso que está disponível na computação que iremos usar será o *socket*. Ele será útil para fazer a comunicação entre o aplicativo do comerciante e o aplicativo

do usuário final. Iremos abordar a sua definição e quais são as principais funções usadas para implementar o protótipo proposto neste trabalho.

2.7.1 Definição

Através dos sockets programas se comunicam com outros usando descritores de arquivos Unix. Um descritor de arquivo é um inteiro associado a um arquivo aberto e este arquivo pode ser uma conexão de rede, um FIFO, um pipe, um terminal, um arquivo no disco, ou qualquer outra coisa [SOCKET, 2012].

Um *socket* não é nada além de uma abstração conveniente. Ele representa um ponto de conexão para uma rede TCP/IP. Quando dois computadores querem manter uma conversação, cada um deles utiliza um *socket*. Um computador é chamado servidor, ele abre um *socket* e presta atenção às conexões. O outro computador denomina-se cliente, ele chama o *socket* servidor para iniciar a conexão. Para estabelecer uma conexão, é necessário apenas um endereço de destino e um número de porta. [HOPSON, 1997].

De uma forma geral, existe uma aplicação que cria um *socket* servidor e uma outra aplicação que implementa um cliente. Existem principalmente dois tipos de sockets: SOCKET_STREAM (TCP) ou SOCK_DGRAM (UDP). [SOCKET-JAVA, 2012].

A operação sem conexão utiliza o UDP (*User Datagrama Protocol*). Um datagrama é uma unidade autônoma que tem todas as informações necessárias para tentar fazer sua entrega. Similar a um envelope, o datagrama tem um endereço do destinatário e do remetente e contém em seu interior os dados a serem enviados. Um *socket* nesse modo de operação não precisa se conectar a um *socket* de destino; ele simplesmente envia o datagrama. O protocolo UDP só promete fazer o melhor esforço possível para tentar entregar o datagrama. A operação sem conexão é rápida e eficiente, mas não é garantida. [HOPSON, 1997].

A operação baseada em conexões emprega o TCP (*Transport Control Protocol*). Um *socket* nesse modo de operação precisa se conectar ao destino antes de transmitir os dados. Uma vez conectados, os sockets são acessados pelo uso de uma interface de fluxos: abertura-leitura-escrita-fechamento. Tudo que é enviado por um

socket é recebido pela outra extremidade da conexão, exatamente na mesma ordem em que foi transmitido. A operação baseada em conexões é menos eficiente do que a operação sem conexão, mas é garantida [HOPSON, 1997].

2.7.2 Socket no Android

Para começarmos a usar *socket* na nossa aplicação, devemos colocar as permissões no AndroidManifest.xml, pois devemos fazer a comunicação através da internet.

A seguir está o código da permissão para o acesso da internet. [PERMISSÃO, 2012].

```
<uses-permission android:name="android.permission.INTERNET"></uses-permission>
```

A seguir está o código da permissão para o acesso do *wifi*.

```
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"></uses-
permission>
```

O Android fornece uma parte dos pacotes *java.net* e *org.apache.http-client* para suportar comunicação de rede básica. Outros pacotes relacionados, como *android.net,* adicionam detalhes de comunicação interna e propriedades gerais de conectividade. [ABLESON, 2008].

Em termos de propriedades de conectividade, o uso de classe *ConectivityManager* para determinar quando a conexão de rede está ativa e que tipo de conexão temos: móvel ou Wi-Fi. A partir daí, iremos usar a rede de vários modos em aplicativos simples. [ABLESON, 2008].

Capítulo 3

Projeto do protótipo de divulgação de produtos em promoção

O aplicativo de consulta de ofertas tem como objetivo oferecer mais opções para os clientes escolherem os pontos de compras de mercadorias e divulgar vários estabelecimentos que o cliente desconhecia, e é uma oportunidade para o dono do estabelecimento ter mais um meio de divulgação que poderá abordar clientes de várias classes.

Com o avanço da tecnologia as pessoas querem comodidade para escolher os seu produtos, um exemplo disso seria a compra de produtos pela internet que o usuário pode escolher e pedir o produto que ele quiser com o conforto de fazer isso tudo sem sair de casa. No protótipo o usuário tem a comodidade de ter em seu telefone quais são as ofertas que estão perto dele, tendo assim uma variedade de ofertas que ele possa escolher.

As ofertas estarão organizadas por categorias, tais como: alimentos, bebidas, automotivo, bebês, beleza e saúde, brinquedos, eletrodomésticos, eletro portáteis, eletrônicos, esporte e lazer, ferramentas, games, informática, telefones e celulares, livros, moveis, utilidades domésticas, papelaria.

3.1 Levantamento de requisitos

Nesta seção serão apresentados a iteração dos usuários com o protótipo e são eles: o comerciante que cadastrará os produtos em promoção e o cliente que poderá realizar as consultas aos produtos em promoção usando o celular. Essa iteração será descrita usando casos de uso e o código do aplicativo.

3.1.1 Diagrama de caso de uso

No aplicativo de ofertas temos dois grupos de usuário que estão envolvidos: o cliente e o comerciante.

O cliente é o usuário que vai acompanhar as ofertas que estarão organizadas por categorias e poderá receber ofertas de várias localidades independente da sua localidade. Assim, o dono do estabelecimento poderá atingir clientes de várias localidades, não sendo apenas os clientes que estão perto do comercio. Segue o Diagrama de Caso de Uso do Cliente.

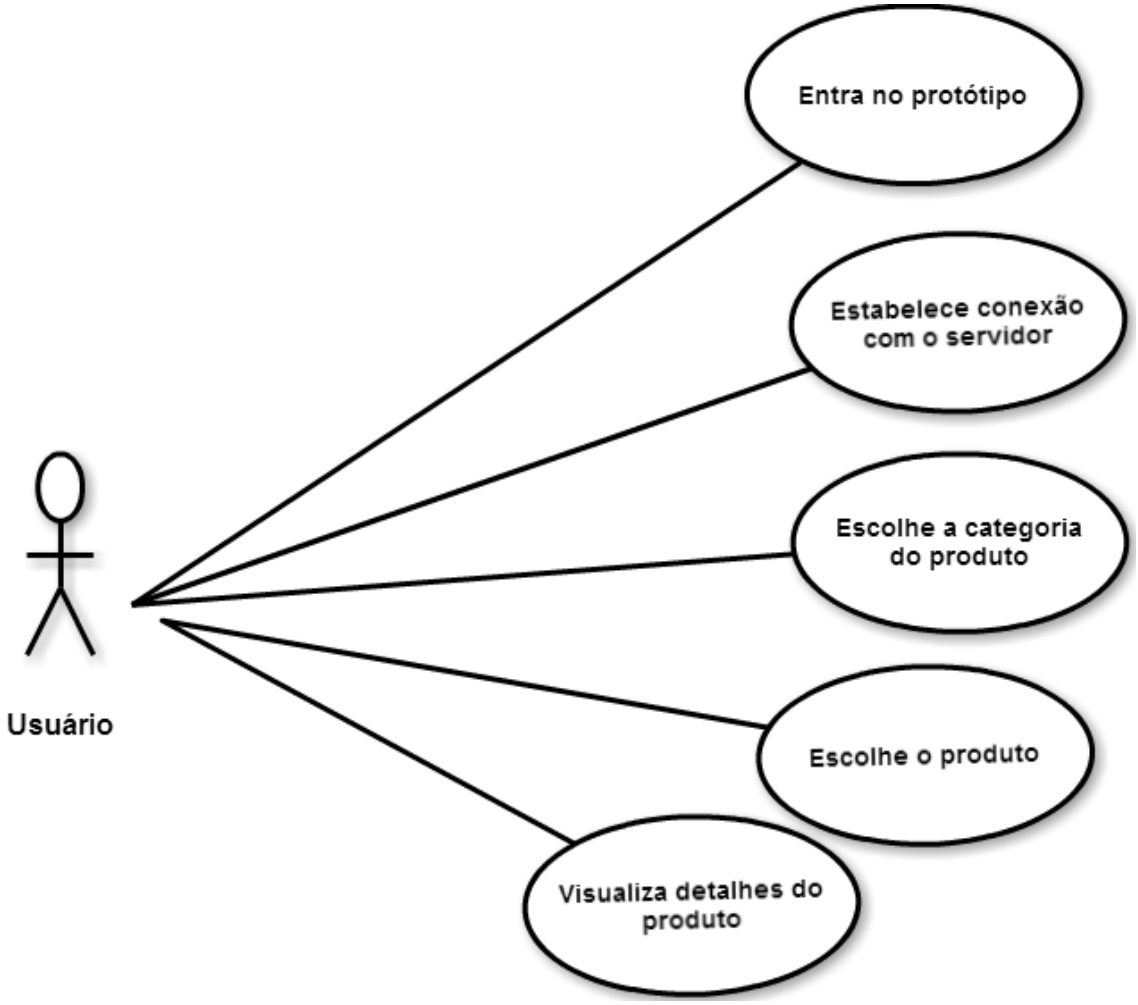


Figura 3.1: Diagrama de caso de uso do Usuário

- Entra no protótipo: o usuário entrará no protótipo, através de um ícone do aplicativo.
- Estabelece conexão com o servidor: O programa vai fazer a conexão com o servidor via socket.
- Escolhe a categoria do produto: O usuário deverá fazer a escolha das categorias dos produtos.

- Escolhe o produto: Depois do usuário escolher as categorias serão mostrados os produtos da categoria escolhida e então ele deverá escolher algum produto de sua escolha.
- Visualiza detalhes do produto: Escolhido o produto será mostrado detalhes do produto como: razão social, endereço e valor.

O comerciante será responsável por cadastrar os produtos em oferta. Os dados que deverão entrar no aplicativo serão: o nome do estabelecimento comercial, a categoria que o produto se enquadra, o nome do produto, o preço do produto, a sua localização o programa junto com o GPS do dispositivo móvel irão captar. Segue o diagrama de caso de uso do comerciante.

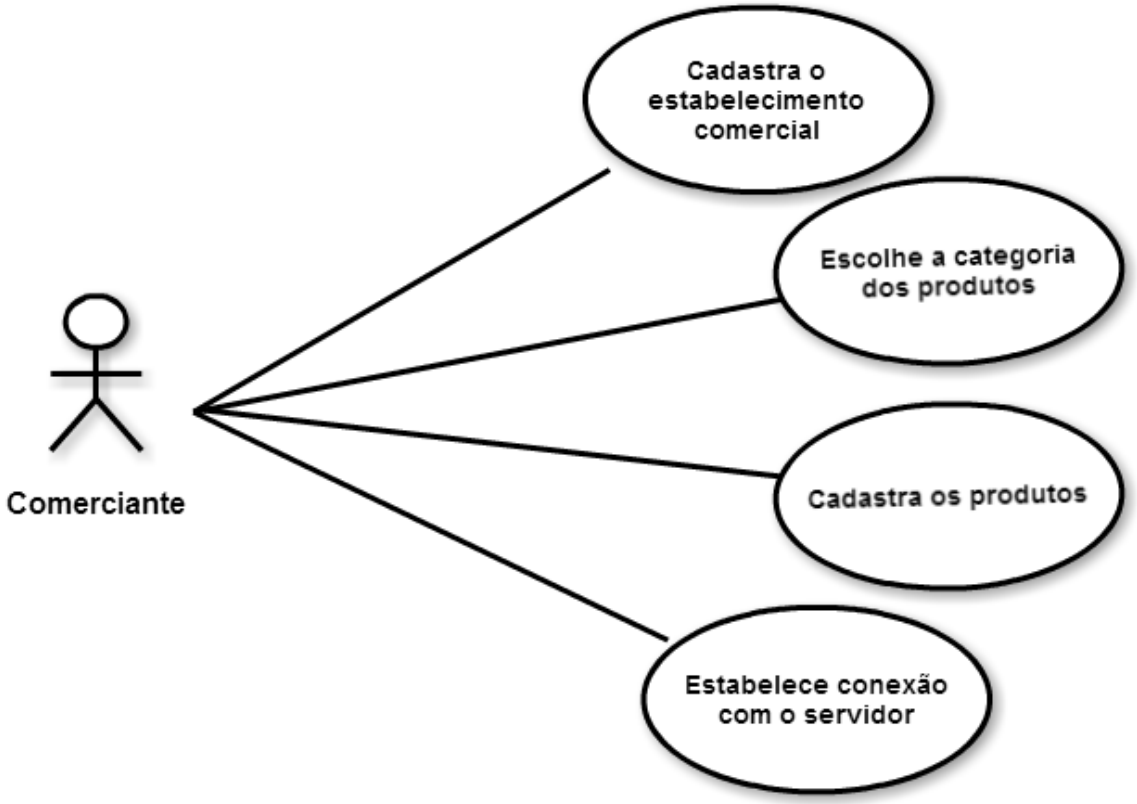


Figura 3.2: Diagrama de caso de uso do Comerciante.

- Cadastra o estabelecimento comercial: O comerciante deverá preencher os dados referentes ao estabelecimento tais como: razão social, endereço, telefone.
- Escolhe categoria do produto: O comerciante deverá escolher qual a categoria em que o produto a ser cadastrado será incluído.

- Cadastra produto: O comerciante deverá preencher um formulário com os dados do produto.
- Estabelece conexão com o servidor: Cadastrado os devidos produtos o protótipo vai gravar os dados no servidor.

3.1.2 Digrama de atividade

Nesta seção iremos detalhar o protótipo através do diagrama de atividade para mostrar os passos de funcionamento do protótipo.

Começaremos detalhando o protótipo do comerciante, logo em seguida vamos detalhar o protótipo do cliente.

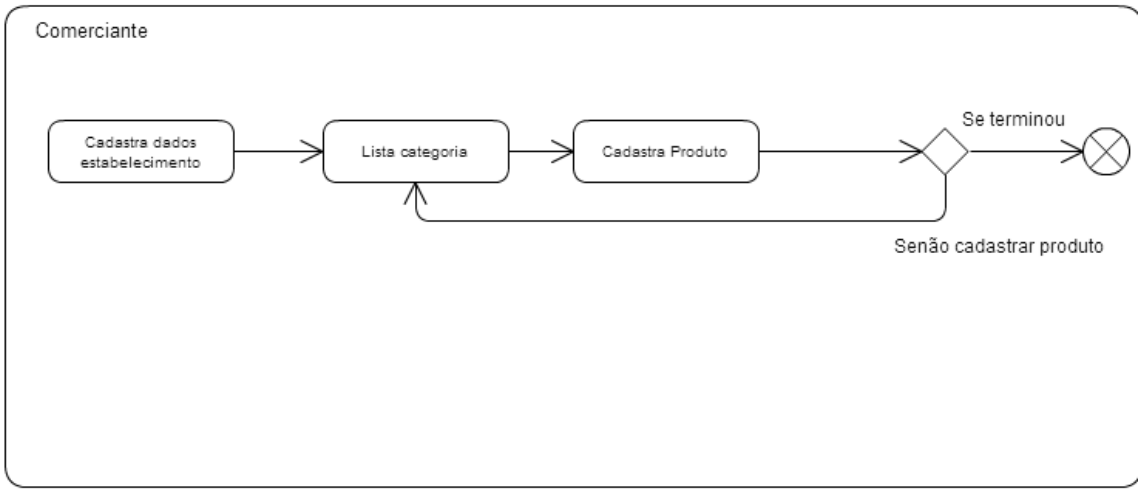


Figura 3.3: Diagrama de atividade do Comerciante

A Figura 3.3 detalha os passos que o comerciante deverá fazer para que cadastre um produto no protótipo.

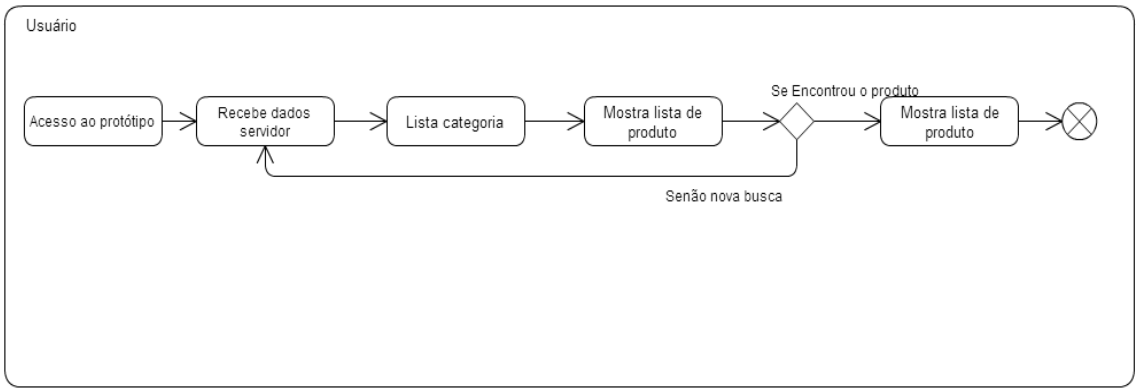


Figura 3.4: Diagrama de atividade do Cliente

A Figura 3.4 detalha os passos que o usuário deve seguir para que possa visualizar os produtos.

3.1.3 Diagrama de classes

Nesta seção vamos mostrar o diagrama das classes que detalhará quais serão os atributos e os métodos usados para desenvolver o protótipo.

A Figura 3.5 detalha as classes que foi usado para desenvolver o protótipo do comerciante, nela será mostrado os atributos e os métodos usados.

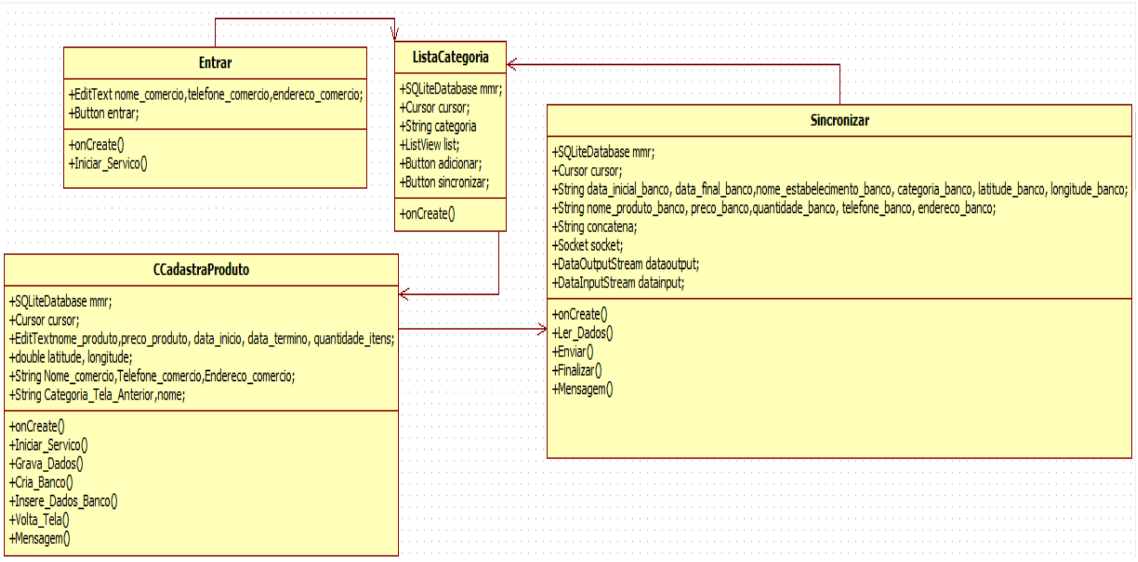


Figura 3.5: Diagrama de classe do Comerciante

A Figura 3.6 detalha as classes que foi usado para desenvolver o protótipo do usuário, nela será mostrado os atributos e os métodos usados.

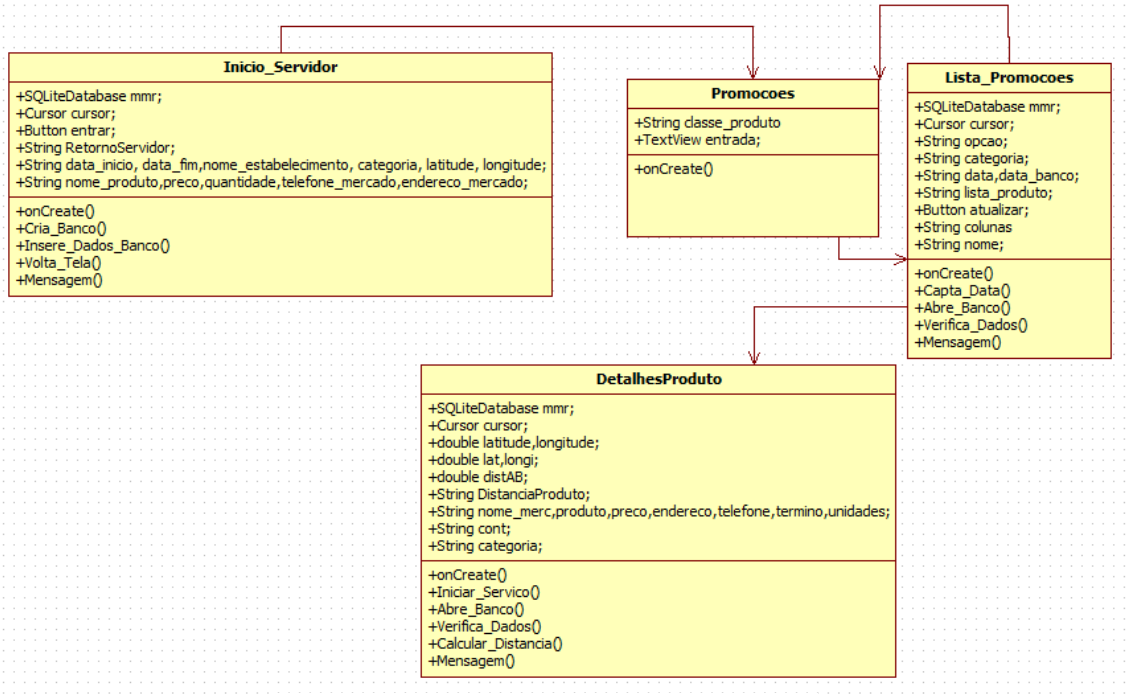


Figura 3.6: Diagrama de classe do Cliente

3.2 Requisitos mínimos do sistema

Para o funcionamento adequado do aplicativo de promoções é necessário um dispositivo móvel com acesso a internet, GPS e com sistema operacional Android 2.2 ou superior. A versão utilizada nos testes realizados e produção do manual do usuário está disponível no Apêndice A.

3.3 Principais classes da implementação do sistema.

No sistema temos dois aplicativos, o módulo comerciante para o comerciante cadastrar os produtos em oferta e o móduclo cliente para os clientes consultarem os produtos em oferta.

A seguir são apresentados detalhes das principais classes do módulo comerciante e cliente. Essas classes estão nos apêndice C e D.

3.3.1 Módulo comerciante

As principais classes do módulo comerciante são apresentadas na Tabela 3.1.

Classes	Objetivos
Entrar	Classe responsável por iniciar a aplicação e realizar a captação dos dados iniciais do comerciante
ListaCategoria	Classe responsável por listar as categorias de classificação dos produtos
CCadastraProduto	Classe que realiza o cadastro do produto no banco de dados juntamente com os dados iniciais do estabelecimento comercial
Sincronizar	Classe que realiza o tratamento e envio dos dados via Socket para o servidor

Tabela 3.1: Detalhes das classes do comerciante

CLASSE Entrar

MÉTODO onCreate (): método responsável por captar as informações iniciais deste medulo como nome, endereço e telefone do comércio.
MÉTODO Iniciar_Servico (): realiza o serviço de iniciar o GPS para que se possa realiza a captação das coordenadas de latitude e longitude.

CLASSE ListaCategoria

MÉTODO onCreate (): método responsável por receber os dados informados pelo usuário na tela anterior e listar as categorias em que os produtos serão classificados e enviar para a próxima tela a categoria informada pelo usuário juntamente com os demais valores recebidos.

CLASSE CCadastraProduto

MÉTODO onCreate (): realiza a captação dos dados enviados pela tela anterior e realiza tambem a captação dos dados informados pelo usuário referente ao produto cadastrado.
MÉTODO Iniciar_Servico (): realiza o serviço de captar as coordenadas de latitude e longitude.
MÉTODO Grava_Dados (): realiza o tratamento de espaços em branco nos itens como nome do estabelecimento, endereço e nome do produto.
MÉTODO Cria_Banco (): método responsável por criar o banco de dados local.
MÉTODO Insere_Dados_Banco (): método responsável por gravar os dados referente ao produto em promoção e ao estabelecimento comercial onde este produto se encontra na base de dados local.
MÉTODO Volta_Tela (): realiza a chamada da próxima tela de navegação.
MÉTODO Mensagem (): método chamado caso haja algum problema na execução de algum método acima.

CLASSE Sincronizar

MÉTODO onCreate (): método responsável por permitir que haja a conexão via rede de internet.
MÉTODO Ler_Dados (): método responsável por abrir e ler os dados inseridos na base de dados local e realizar o tratamento para que se possa ser enviado ao servidor.
MÉTODO Enviar (): realiza o envio dos dados tratados via Socket para o servidor.
MÉTODO Finalizar (): realiza o encerramento de forma adequada do Socket e do banco de dados.
MÉTODO Mensagem (): método chamado caso haja algum problema na execução de algum método acima.

3.3.2 Módulo cliente

As principais classes do módulo cliente são apresentadas na Tabela 3.2.

Classes	Objetivos
Inicio_Servidor	Classe responsável por receber os dados vindos do servidor e armazenar no banco de dados local do dispositivo móvel
Promocoes	Classe responsável por listar as categorias de classificação dos produtos
ListaPromocoes	Classe que realiza a listagem dos produtos que foram armazenados no banco de dados e classificá-los de acordo com a categoria informada pelo usuário
DetalhesProduto	Classe responsável por listar os detalhes do produto que o usuário deseja visualizar

Tabela 3.2: Detalhes das classes do Cliente

CLASSE Inicio_Servidor

MÉTODO onCreate (): método responsável por receber os dados q foram enviados pelo servidor
MÉTODO Cria_Banco (): realiza a criação ou a abertura do banco de dados com as colunas data_inicio_promocao String , data_fim_promocao String , nome_estabelecimento String, categoria String, latitude double, longitude double,

nome_produto String, preco float, quantidade String, telefone_mercado String, endereco_mercado String

MÉTODO Insere_Dados_Banco (): realiza a gravação no banco de dados dos valores recebidos pelo servidor

MÉTODO Volta_Tela (): realiza a chamada da próxima tela de navegação

MÉTODO Mensagem (): método chamado caso haja algum problema na execução de algum método acima

CLASSE Promocoes

MÉTODO onCreate (): método responsável por listar as categorias em que os produtos serão classificados e enviar para a próxima tela a categoria informada pelo usuário

CLASSE Lista_Promocoes

MÉTODO onCreate (): método responsável por receber os dados enviados pela tela anterior e listar os devidos produtos que estão em promoção em certa categoria e com o período de validade da promoção válido e também realizar a chamada da nova tela caso clique em algum item da lista de produtos em promoção juntamente com a ação de voltar para a tela em que se tem a lista de categorias para que se possa realizar uma nova busca.

MÉTODO Capta_Data (): método responsável por realizar a captação da data atual.

MÉTODO Abre_Banco (): método responsável por realizar a abertura do banco de dados para que se possa ler os valores recebidos pelo servidor.

MÉTODO Verifica_Dados (): método responsável por ler os dados gravados no banco para que se possa realizar as devidas comparações entre os dados recebidos da tela anterior e os dados armazenados no banco.

MÉTODO Mensagem (): método chamado caso haja algum problema na execução de algum método acima.

CLASSE DetalhesProduto

MÉTODO onCreate (): método responsável por receber os dados enviados pela tela anterior.
MÉTODO Iniciar_Servico (): realiza o serviço de captar as coordenadas de latitude e longitude.
MÉTODO Abre_Banco (): método responsável por realizar a abertura do banco de dados para que se possa ler os valores recebidos pelo servidor.
MÉTODO Verifica_Dados (): método responsável por ler os dados gravados no banco para que se possa realizar as devidas comparações entre os dados recebidos da tela anterior e os dados armazenados no banco.
MÉTODO Calcular_Distancia (): método responsável por calcular a distancia entre as coordenadas que foram obtidas pelo método Iniciar_Servico() e os valores lidos do banco de dados,método que realiza a exibição do conteúdo desejado pelo cliente como nome do produto,preço,distancia, nome do comércio, endereço, telefone, data de término da promoção e a quantidade de itens em promoção.
MÉTODO Mensagem (): método chamado caso haja algum problema na execução de algum método acima.

3.4 Detalhes dos aplicativos

Para a realização dos testes foi utilizado o smartphone Samsung GALAXY S4 modelo GT-I9500 com processador de 1.6GHz Quad Core + 1.2 GHz Quad Core, com 16 GB de memória com sistema operacional móvel Android versão 4.3.

Agora serão apresentadas as telas dos módulos Comerciante e Cliente com detalhes sobre a funcionalidade de cada tela.

3.4.1 Aplicativo do comerciante

Figura 3.7 – Inicio do módulo Comerciante

Na Figura 3.7 o comerciante deve preencher com os dados referentes ao seu estabelecimento comercial.

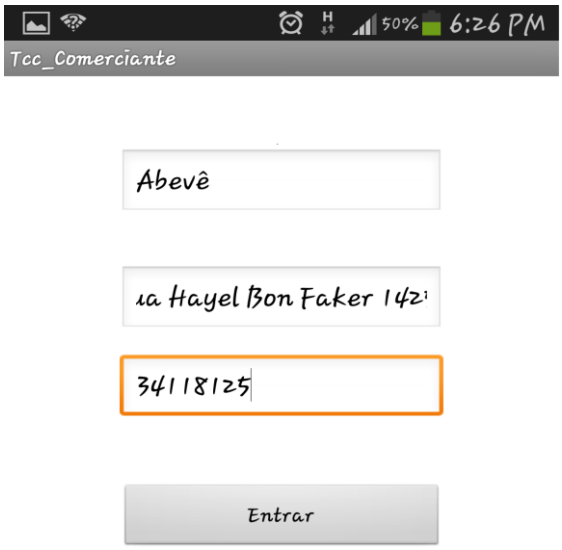


Figura 3.8 – Inserçao dos dados comerciais

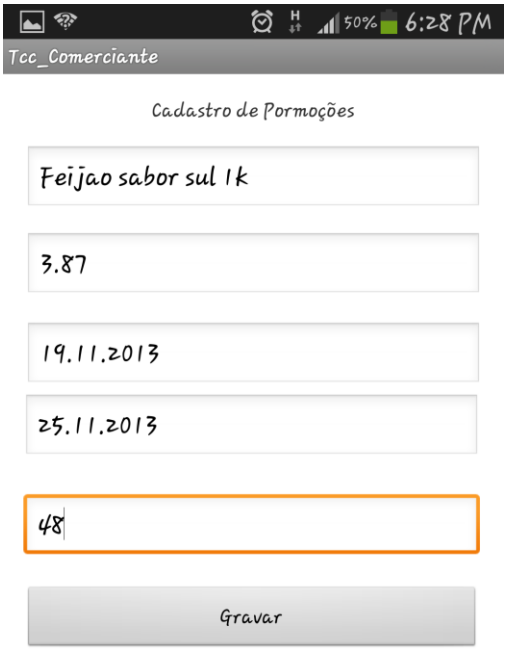
A tela da Figura 3.8 tem a finalidade de colher os dados de localização do comércio, para que o usuário possa ter uma referencia de localização, juntamente com um telefone para contato. Além disto, nesta tela o dono do estabelecimento comercial cadastra de alguns produtos que estão em promoção.

Figura 3.9 – Lista de categorias

A tela de lista de categorias (Figura 3.9) mostra as categorias que os produtos em promoção devem se encaixar. O comerciante escolhe uma delas e será redirecionado a outra tela, a qual fará o cadastro dos dados do produto que estará em promoção.

Figura 3.10 – Cadastro de produtos

A tela de cadastro dos produtos (Figura 3.10), possui 5 campos, os quais possuem funcionalidade diferente, porém ficam pré vizualizados para que o usuário do módulo comerciante tenha uma precisão exata de quais valores devem ser inseridos nos respectivos campos.



3.11 – Inserçao de dados referente ao produto e promoção

A tela da Figura 3.11 possui os principais atributos para registro de uma promoção. Esses atributos são: detalhes do produto, preço, data de inicio da promoção, data de término da promoção juntamente com a quantidade de itens em estoque oferecidos para tal promoção. Após a inserção dos dados referentes ao produto em promoção seleciona-se a opção de gravar, então os dados são armazenados em uma base de dados local e enviados para o servidor via socket. Ao término da gravação, a tela de categorias é mostrada ao usuário.

3.4.2 Aplicativo do cliente

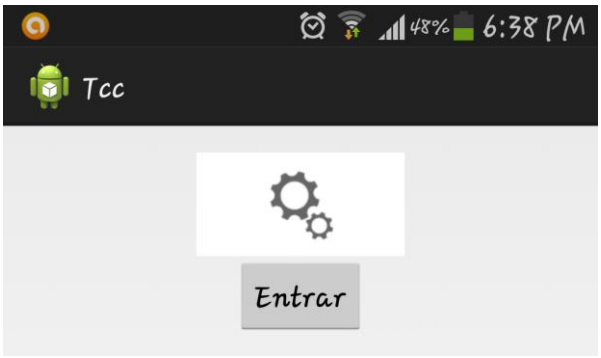
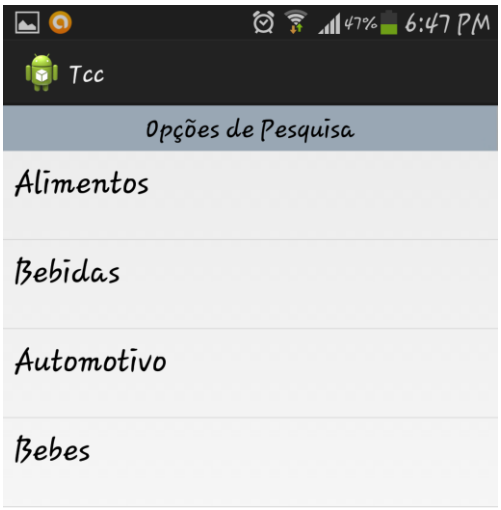


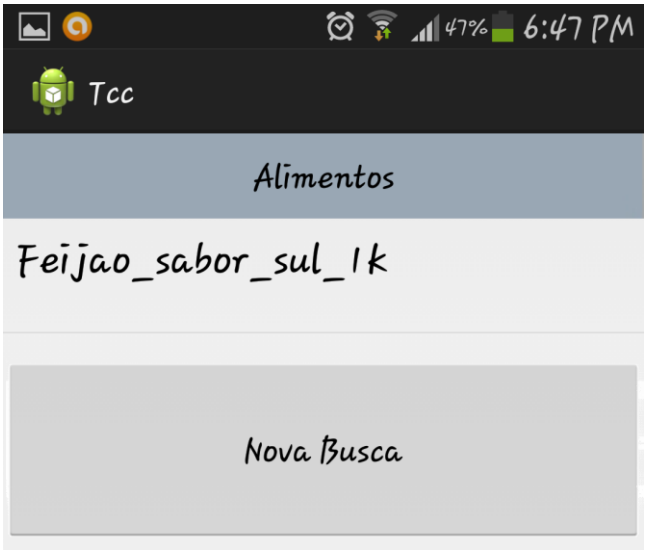
Figura 3.12 – Inicio do módulo cliente

A Figura 3.12 ilustra a primeira tela do módulo Cliente, nela contém apenas um botão, o qual é responsável por realizar a busca de dados no servidor, dados os quais foram enviados ao servidor pelo módulo Comerciante.



3.13 – Categorias

A tela de categorias (Figura 3.13) mostra a lista de categorias dos produtos que estão em promoção.



3.14 – Lista de produtos em promoção

Feita a escolha da categoria a tela (Figura 3.14) que é chamada é a tela de produtos que estão dentro da categoria escolhida pelo usuário. Se o cliente desejar escolher outra categoria de promoções ele deve escolher a opção de “Nova Busca”, escolhendo essa opção ele será encaminhado para a tela anterior que será a tela de

categorias. Caso o usuário queira ver detalhes de algum produto que ele queira basta selecionar com um toque o produto que ele será encaminhado para a tela seguinte.

3.15 – Detalhes produto

Na tela detalhes produto (Figura 3.15) serão mostrados os detalhes do produto escolhido pelo cliente em promoção, tendo como informação: Preço do produto, Distancia (está que é calculada levando a consideração do ponto em que o cliente está e a localização do estabelecimento comercial), Razão Social que é o nome da empresa que está oferecendo o produto em promoção, juntamente com o endereço do estabelecimento comercial e o telefone para contato, é mostrado também a data de término da promoção do produto escolhido, juntamente com o estoque inicial de itens que iniciaram a promoção.

Capítulo 4

Considerações finais

Neste trabalho foi realizado o estudo da plataforma Android para a construção de um protótipo para informar aos usuários produtos em promoção. Para alcançar o objetivo foram estudados os sistemas operacionais móveis, o acesso ao banco de dados SQLite e o acesso à localização do usuário através do GPS. Para tanto, foram consultados livros e a internet a fim adquirir um embasamento teórico das tecnologias para que pudesse ser implementado o protótipo proposto.

Podemos concluir na perspectiva de utilização de um aplicativo Android por parte do usuário que a maioria dos sistemas operacionais móveis se preocupam em oferecer uma interface intuitiva. Sendo assim, estão disponíveis de acordo com as tecnologias de cada dispositivo móvel novas funcionalidades que podem ser obtidas em lojas virtuais. Dentre essas extensões podemos citar o acelerômetro, GPS, alto falantes, multi-touch e muitos outros.

4.1 Dificuldades encontradas

Durante a realização deste estudo, foi possível notar que as ferramenta para o desenvolvimento de aplicações para a plataforma Android consomem muito processamento, e também há casos em que a aplicação desenvolvida não funciona corretamente em ambiente simulado, pelo fato de não haver uma igualdade entre as arquiteturas.

4.2 Trabalhos futuros

Como a plataforma Android possui um amplo ramo de pesquisa e desenvolvimento alguns trabalhos futuros podem ser complementados nos módulos Cliente e Comerciante, sendo alguns deles:

- Interface de mapas no módulo Cliente: permitir vizualizar o ponto em que um produto esta e mostrar o caminho exato entre o usuário da aplicação e o produto divulgado;
- Inserção de um campo de pesquisa direta: possibilitar obter um produto em promoção com mais rapidez, por exemplo usando o nome do produto que será inserido pelo usuário;
- Inserção de produtos em promoção via leitor de código de barras: possibilitar agilidade no cadastro dos produtos em promoção pelo comerciante.

Referências

ABLESON. Frank; KING,Chris; SEN,Robi, Android em Ação, 3ª Ed Curitiba: Elsevier Brasil, 2008.

ADB. Android Debug Bridge. Disponível em: http://developer.android.com/tools/help/adb.html, 2013. Ultimo acesso em: 19 novembro. 2013.

ANDROID COM SQLITE. Introdução ao Android com SQLite. Disponível em: http://www.theclub.com.br/Restrito/Revistas/201112/intr1211.aspx, 2012. Ultimo acesso em: 19 novembro. 2013.

ANDROID SDK. Obter o SDK Android. Disponível em: http://developer.android.com/sdk/index.html, 2012. Ultimo acesso em: 19 novembro. 2013.

ANDROID SQLITE. Android SQLite. Disponível em: http://www.vogella.com/articles/AndroidSQLite/article.html, 2013. Ultimo acesso em: 19 novembro. 2013.

ARM. Arqutetura ARM. Disponível em: http://pt.wikipedia.org/wiki/Arquitetura_ARM, 2013. Ultimo acesso em: 19 novembro. 2013.

COMUNICAÇÃO. Comunicação de Computadores utilizando Sockets. Disponível em: http://www2.unoeste.br/~chico/comunicacao_socket/, 2012. Ultimo acesso em: 19 novembro. 2013.

GPS. Location Strategies. Disponível em: http://developer.android.com/guide/topics/location/strategies.html, 2012. Ultimo acesso em: 19 novembro. 2013.

HOPSON. Hopson, K. C e Ingram, Stephen E. *Desenvolvendo Applets com Java*. Editora Campus, 1997. Págs. 335 – 351.

LECHETA. Ricardo R, Google Android: Aprenda a Criar Aplicações para Dispositivos Móveis com o Android SDK, 3ª Ed Novatec, 2013.

PERMISSÃO. Trabalhando com Socket no Android. Disponível em: http://www.portalandroid.org/comunidade/viewtopic.php?f=7&t=11077, 2012. Ultimo acesso em: 19 novembro. 2013.

SDK. Android Software Development. Disponível em: http://em.wikipedia.org/wiki/Android_software_development, 2013. Ultimo acesso em: 19 novembro. 2013.

SQLITE NO ANDROID. SQLite no Android – Artigo Web Mobile 34. Disponível em: http://www.devmedia.com.br/sqlite-no-android-artigo-web-mobile-34/19201, 2012. Ultimo acesso em: 19 novembro. 2013.

SILVA. L. A (2010). Programando passo a passo 4ª Edição. Rio de Janeiro.

SOCKET-JAVA. Redes – Sabe o que são sockets de comunicação?. Disponível em: http://pplware.sapo.pt/networking/redes-sabe-o-que-sao-sockets-de-comunicacao-parte-i/http://blog.globalcode.com.br/2011/02/java-e-redes-consultando-um-servico-com.html#.UnACVPmkpWY, 2012. Ultimo acesso em: 19 novembro. 2013.

SOCKET. Tutorial de Sockets. Disponível em: http://olinux.uol.com.br/artigos/370/1.html, 2012. Ultimo acesso em: 19 novembro. 2013.

THOMAS. Thomas, Michael D., Patel, Pratik R., Hudson, Alan D e Ball, Donald A Jr. *Programando em Java para a Internet*. Makron Books, 1997. Págs. 467 – 489

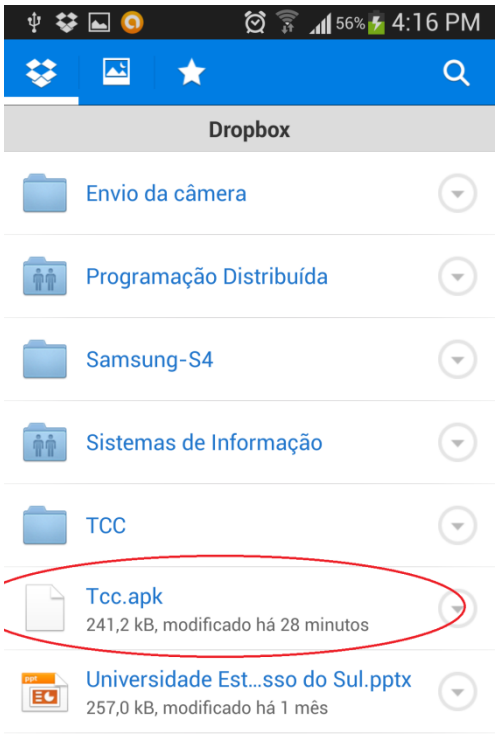
Anexo A

Manual do usuário

O programa que mostra as ofertas de estabelecimento comercial é dividido em dois programas: Um para o comerciante que servirá para cadastrar os produtos que serão divulgados para os clientes, e o outro programa será o do cliente que ira usá-lo para procuras os produtos de sua preferencia.

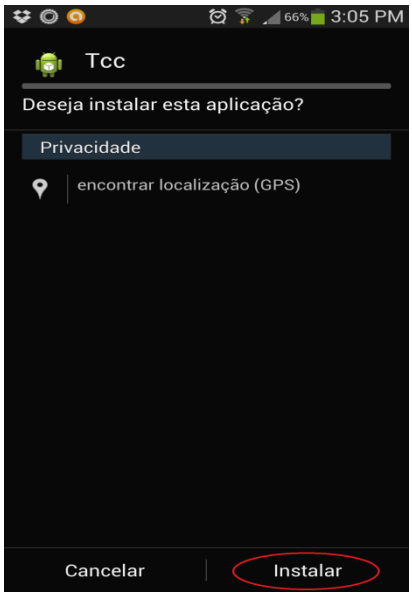
A.1 Programa do Usuário

Para realizar a instalação de um aplicativo Android deve adquirir o arquivo Tcc.apk que aqui esta disponível via Dropbox, para iniciar a instalação deve-se clicar sobre o ícone que indica o aplicativo.



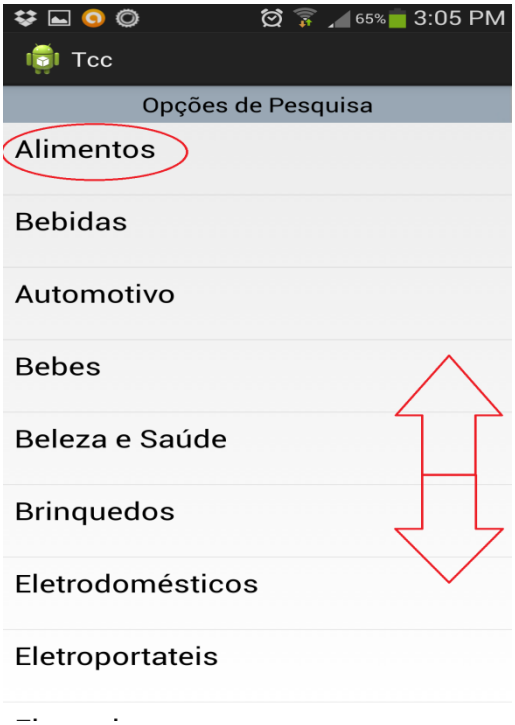
Ao clicar sobre o ícone Tcc.apk é iniciado o processo de instalação do aplicativo em seu aparelho celular, o qual exibe uma tela onde é constatado todas as informações de acesso que o aplicativo realizara sobre seu dispositivo móvel, como este aplicativo

tem a capacidade de calcular a distancia entre o aparelho celular e o produto em oferta, as permissões necessárias são as permissões de localização, de modo com que o usuário do sistema tenha total conhecimento das permissões que a aplicação realizara e também a oportunidade de cancelar a instalação se descordar de alguma permissão requerida pela aplicação.

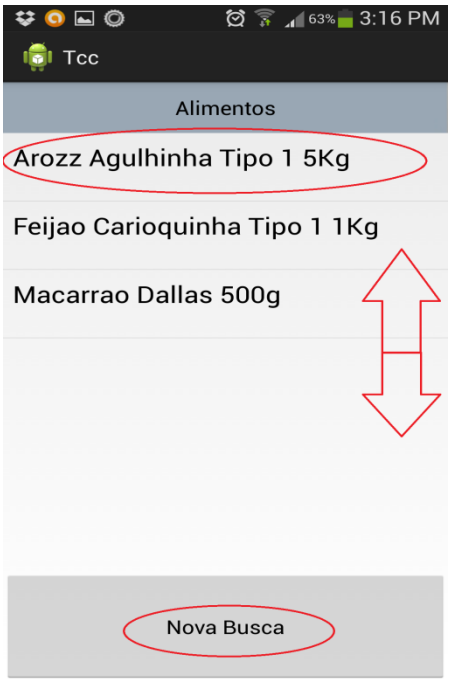


Ao terminar o processo de instalação da aplicação é exibida uma tela onde se tem a opção de iniciar o aplicativo imediatamente ou apenas instalar pra uma inicialização futura, nesta tela também contem o status da instalação, caso seja bem sucedida a mensagem “aplicação instalada” aparecera na tela do seu dispositivo, caso contrario a mensagem “erro durante a instalação” aparecera no visor do seu dispositivo.

Ao clicar em “Concluído” o aplicativo foi instalado com sucesso porem sua inicialização não será iniciada, ao clicar em “Abrir” será iniciada a execução do aplicativo imediatamente.

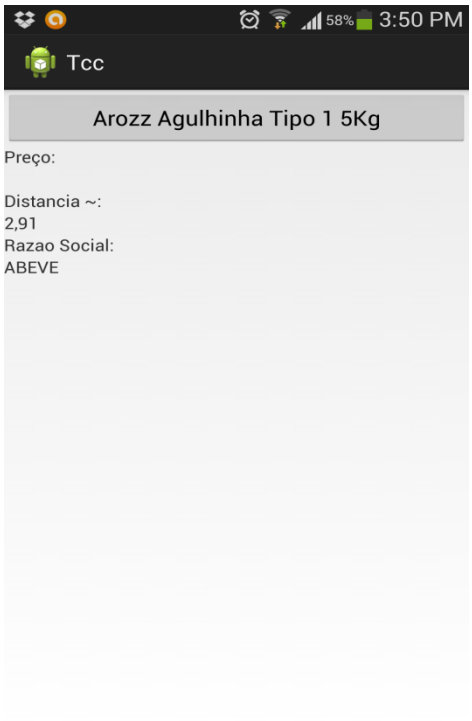


A classificação dos produtos que são de seu interesse, como essas classes são muitas, há a opção de deslizamento da lista através da tela do dispositivo, ao se clicar em uma opção a próxima tela é chamada.primeira tela do aplicativo referencia as classes de produtos para que o usuário possa obter uma



Esta nova tela possui varias funcionalidades uma delas é retornar a tela anterior para que se possa realizar uma nova busca de produtos que estão em outra classificação, esta tela possui também o sistema de deslizamento da lista de produtos, a qual é determinada apenas pelo movimento do seu dedo em relação à tela, caso haja algum

item de interesse pode-se clicar em algum item da lista, chamando a nova tela, onde se obtém os detalhes do produto.



Esta tela possui os itens principais e mais importantes itens do aplicativo, tela na qual é mostrada a distancia em linha reta da localização do produto em relação a sua localização, preço do produto que esta em promoção, e também o nome do comércio para que o usuário tenha conhecimento da localização do produto.

Anexo B

Manual de desenvolvimento

Nesta seção vamos mostrar como é feita a instalação dos componentes que são necessários para desenvolver aplicativos para a plataforma Android e seus complementos.

O Android SDK proporciona ao desenvolvedor bibliotecas API e ferramentas de desenvolvimento necessárias para construir, testar e depurar aplicativos através de um emulador para a plataforma Android.

Após baixar o arquivo executável compactado "android-sdk-_r22.3-windows.zip" no site[1], e executando-o, instala-se as ferramentas do Android SDK conforme a Figura B.1.

---

[1] http://developer.android.com/sdk/index.html

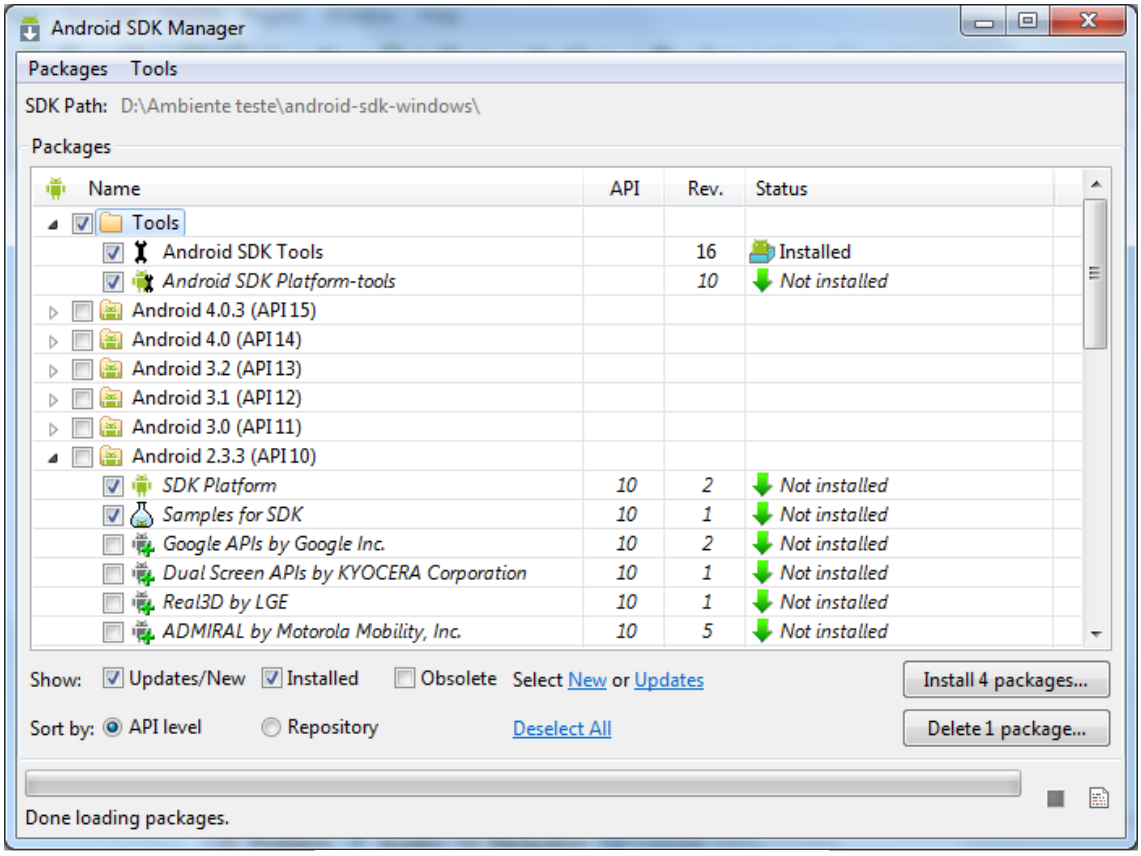


Figura B.1 – Gerenciador Android SDK

Após a conclusão, executar o SDK Manager.exe que é encontrado no diretório principal da instalação. Aqui é onde se encontra quais ferramentas adicionais, atualizações, plataformas e outros pacotes de componentes o desenvolvedor escolhe para baixar, apenas selecionando as opções disponíveis e clicando em Install packages. É necessário baixar ao menos uma plataforma para começar a desenvolver os aplicativos.

Instalar o ambiente de desenvolvimento Eclipse, disponível na página oficial do Eclipse[2] recomenda-se instalar o Eclipse 3.6.2 (Helios) ou posterior.

A próxima etapa consiste em instalar o ADT plugin para Eclipse.
1. Inicie o Eclipse, selecione “Help”, selecione “Install New Software....”;
2. Clique “Add”, no canto superior direito;
3. No diálogo Add Repository, entre com “ADT Plugin” para Name e a seguinte URL para Location: https://dl-ssl.google.com/android/eclipse/;
4. Clique “OK”
5. Na tela seguinte, Figura B.2 “Avaliable Software”, selecione a opção “Developer Tools” e clique em “Next”

[2] http://www.eclipse.org/downloads/

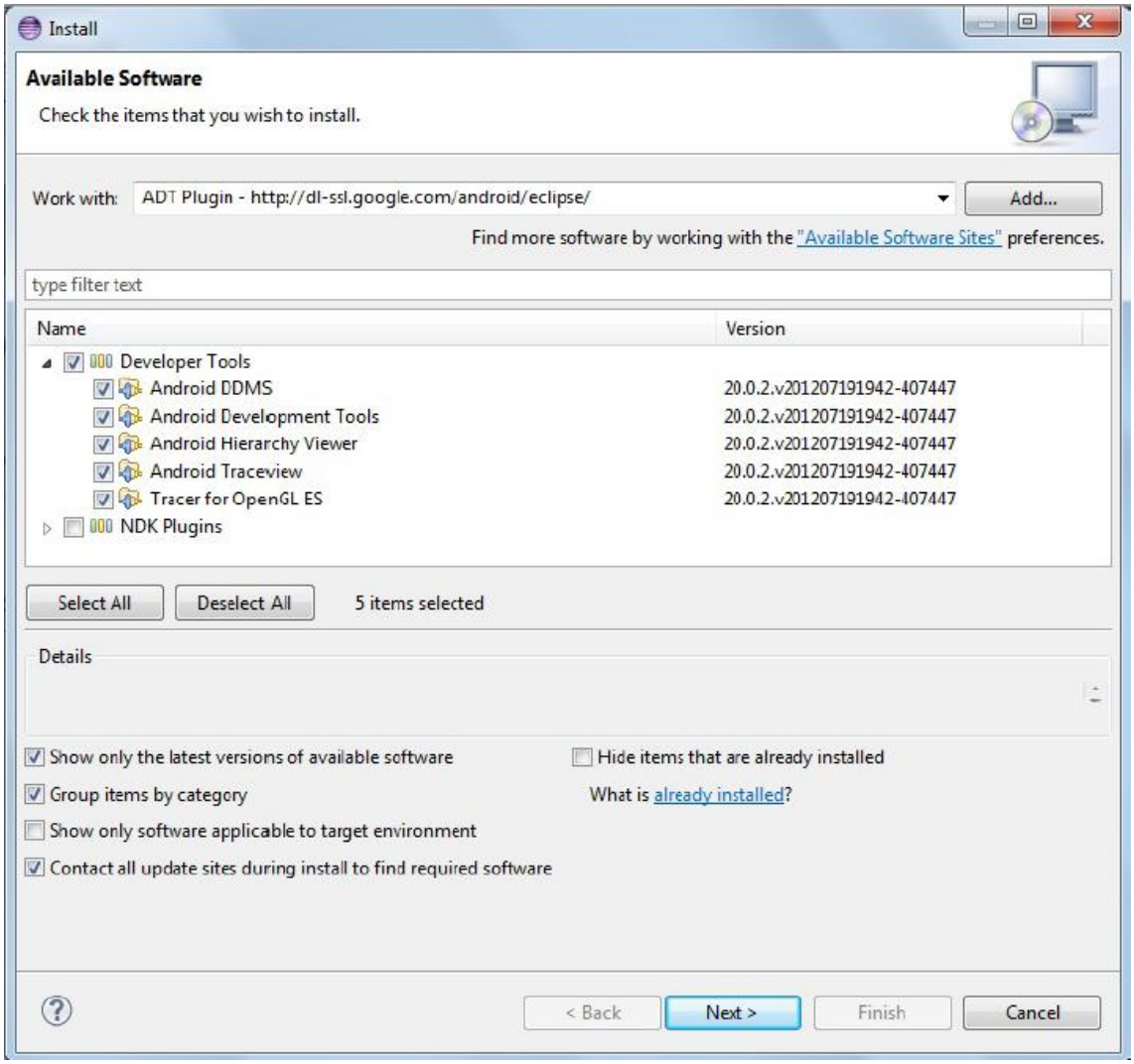


Figura B.2 – Instalaçao do Android SDK

6. Na próxima janela, aparecerá uma lista de ferramentas a serem baixadas. Clique "Next";
7. Leia e aceite as licenças, e clique "Finish";
8. Quando completar, reinicie o Eclipse;

Para a criação do emulador Android (Android Device Emulator (AVD)) deve-se seguir os próximos passos:

1. Inicie o Eclipse, selecione "Window", selecione "AVD Manager";
2. Na tela "Android Virtual Device Manager", escolha "New. . . ";
3. Escolha um nome qualquer para seu novo dispositivo virtual;
4. Em "target", escolha a API level para qual irá criar uma aplicação;
5. Em "SD Card Size", escolha o tamanho do cartão de memória do emulador;
6. Em "Skin", escolha o tamanho da tela do emulador;
7. Para finalizar, clique em "Create AVD";

Anexo C

Classes do aplicativo do Comerciante

C.1 Classe entrar

```
1.  package com.example.tcc_comerciante;
2.
3.  import java.io.DataInputStream;
4.  import java.io.DataOutputStream;
5.  import java.io.IOException;
6.  import java.net.Socket;
7.
8.  import android.app.Activity;
9.  import android.app.AlertDialog;
10. import android.content.Intent;
11. import android.database.Cursor;
12. import android.database.sqlite.SQLiteDatabase;
13. import android.os.Bundle;
14. import android.os.StrictMode;
15. import android.widget.Button;
16. import android.widget.ListView;
17. import android.widget.Toast;
18.
19. public class Sincronizar extends Activity
20. {
21.         SQLiteDatabase mmr = null;
22.         Cursor cursor;
23.
24.         String
    data_inicial_banco,data_final_banco,nome_estabelecimento_banco,categoria_banco,la
    titude_banco,longitude_banco;
25.         String
    nome_produto_banco,preco_banco,quantidade_banco,telefone_banco,endereco_banco;
26.
27.         String concatena;
28.         Socket socket= null;
29.
30.         DataOutputStream dataoutput = null;
31.         DataInputStream datainput = null;
32.
33.         public void onCreate(Bundle savedInstanceState)
34.         {
35.                 super.onCreate(savedInstanceState);
36.
37.                 StrictMode.ThreadPolicy policy = new
    StrictMode.ThreadPolicy.Builder().permitNetwork().build();
38.                 StrictMode.setThreadPolicy(policy);
39.
40.                 Ler_Dados();
41.
42.          };
43.          public void Ler_Dados()
44.          {
45.                 mmr = openOrCreateDatabase("mmr",MODE_WORLD_READABLE, null);
46.                 String colunas[] =
    {"data_inicio_promocao","data_fim_promocao","nome_estabelecimento","categoria","l
    atitude","longitude",
47.
           "nome_produto","preco","quantidade","telefone_mercado","endereco_mercado"
    };
48.                 cursor = mmr.query("oferta", colunas, null, null, null, null,
    null);
49.                 try
50.                 {
51.                         int numeroRegistro = cursor.getCount();
52.                         if (numeroRegistro > 0)
53.                         {
54.                                 cursor.moveToPosition(numeroRegistro-1);
55.
           data_inicial_banco=cursor.getString(cursor.getColumnIndex("data_inicio_pr
    omocao"));
```

```
56.
        data_final_banco=cursor.getString(cursor.getColumnIndex("data_fim_promoca
    o"));
57.
        nome_estabelecimento_banco=cursor.getString(cursor.getColumnIndex("nome_e
    stabelecimento"));
58.
        categoria_banco=cursor.getString(cursor.getColumnIndex("categoria"));
59.
        latitude_banco=cursor.getString(cursor.getColumnIndex("latitude"));
60.
        longitude_banco=cursor.getString(cursor.getColumnIndex("longitude"));
61.
        nome_produto_banco=cursor.getString(cursor.getColumnIndex("nome_produto")
    );
62.
        preco_banco=cursor.getString(cursor.getColumnIndex("preco"));
63.
        quantidade_banco=cursor.getString(cursor.getColumnIndex("quantidade"));
64.     telefone_banco=cursor.getString(cursor.getColumnIndex("telefone_mercado")
    );
65.
        endereco_banco=cursor.getString(cursor.getColumnIndex("endereco_mercado")
    );
66.
67.                             concatena=data_inicial_banco;
68.                             concatena=concatena.concat(" ");
69.                             concatena=concatena.concat(data_final_banco);
70.                             concatena=concatena.concat(" ");
71.
        concatena=concatena.concat(nome_estabelecimento_banco);
72.                             concatena=concatena.concat(" ");
73.                             concatena=concatena.concat(categoria_banco);
74.                             concatena=concatena.concat(" ");
75.                             concatena=concatena.concat(latitude_banco);
76.                             concatena=concatena.concat(" ");
77.                             concatena=concatena.concat(longitude_banco);
78.                             concatena=concatena.concat(" ");
79.                             concatena=concatena.concat(nome_produto_banco);
80.                             concatena=concatena.concat(" ");
81.                             concatena=concatena.concat(preco_banco);
82.                             concatena=concatena.concat(" ");
83.                             concatena=concatena.concat(quantidade_banco);
84.                             concatena=concatena.concat(" ");
85.                             concatena=concatena.concat(telefone_banco);
86.                             concatena=concatena.concat(" ");
87.                             concatena=concatena.concat(endereco_banco);
88.
89.                             Enviar();
90.                             }
91.                         }catch(Exception erro)
92.                         {
93.                                 Mensagem("Erro ao inserir: "+erro.getMessage(),
    "ERRO");
94.                         }
95.         }
96.         public void Enviar()
97.         {
98.                         try
99.                         {
100.                            socket = new Socket("192.168.1.102",4001);
101.
102.                            dataoutput= new
    DataOutputStream(socket.getOutputStream());
103.                            datainput= new
    DataInputStream(socket.getInputStream());
104.                            dataoutput.writeUTF(concatena .toString());
105.                            Finalizar();
106.                        }catch(Exception e)
107.                            {
108.                                    e.printStackTrace();
109.                            }
110.        };
111.        public void Finalizar() throws IOException
112.        {
113.                        mmr.close();
114.                        socket.close();
```

```
115.                     dataoutput.close();
116.                     datainput.close();
117.                     Intent i = new Intent(Sincronizar.this
    ,ListaCategoria.class);
118.                     i.putExtra("Nome_comercio",
    nome_estabelecimento_banco.toString());//envia o nome do comercio para a tela que
    lista as categorias
119.                     i.putExtra("Endereco_comercio", endereco_banco.toString());
120.                     i.putExtra("Telefone_comercio", telefone_banco.toString());
121.                     startActivity(i);
122.         };
123.         public void Mensagem (String msg, String titulo)
124.         {
125.                 AlertDialog.Builder msg1 = new
    AlertDialog.Builder(Sincronizar.this);
126.                 msg1.setTitle(titulo);
127.                 msg1.setMessage(msg);
128.                 msg1.setNeutralButton("OK", null);
129.                 msg1.show();
130.         };
131.}
```

C.2 Classe ListaCategoria

```
1.  package com.example.tcc_comerciante;
2.
3.
4.  import android.app.Activity;
5.  import android.app.AlertDialog;
6.  import android.content.Intent;
7.  import android.database.Cursor;
8.  import android.database.sqlite.SQLiteDatabase;
9.  import android.os.Bundle;
10. import android.view.View;
11. import android.widget.AdapterView;
12. import android.widget.AdapterView.OnItemClickListener;
13. import android.widget.ArrayAdapter;
14. import android.widget.Button;
15. import android.widget.ListView;
16. import android.widget.Toast;
17.
18. public class ListaCategoria extends Activity
19. {
20.         //declaração das variaveis
21.     SQLiteDatabase mmr = null;
22.     Cursor cursor;
23.         String[] categoria= new
    String[]{"Alimentos","Bebidas","Automotivo","Bebes","Beleza","Saúde","Brinquedos"
    ,"Eletrodomésticos",
24.
         "Eletroportateis","Eletronicos","Esporte","Lazer","Ferramentas","Games","
    Informatica","Telefones","Celulares","Livros","Moveis",
25.                         "Papelaria"};//essas são as categorias ja definidas para
    ser mostrado para o usuario
26.
27.         String nome_comercio,telefone_comercio,endereco_comercio;
28.
29.     @Override
30.     public void onCreate(Bundle savedInstanceState)
31.     {
32.         ListView list;
33.         Button adicionar;
34.         Button sincronizar;
35.
36.
37.                 Intent i = getIntent();
38.                 nome_comercio = i.getStringExtra("Nome_comercio");//traz o id do
    usuario da outra tela
39.                 endereco_comercio=i.getStringExtra("Endereco_comercio");
40.             telefone_comercio=i.getStringExtra("Telefone_comercio");
41.
42.          super.onCreate(savedInstanceState);
43.          setContentView(R.layout.mostra_categoria);
44.
```

```
45.               ArrayAdapter<String> adapter = new
    ArrayAdapter<String>(this,android.R.layout.simple_list_item_1, categoria);
46.         list = (ListView) findViewById(R.id.listCategoria);
47.         list.setAdapter(adapter);
48.         //caso o usuario clique em algum item da lista para escolher uma
    categoria
49.         //essa funçao é chamada
50.         list.setOnItemClickListener(new OnItemClickListener()
51.         {
52.             public void onItemClick(AdapterView parent, View view,int position,
    long id)
53.             {
54.                Intent j = new Intent(ListaCategoria.this
    ,CCadastraProduto.class);
55.                j.putExtra("Nome_comercio", nome_comercio);//envia o nome do
    comercio que vem da primeira tela
56.                j.putExtra("Endereco_comercio", endereco_comercio);
57.                j.putExtra("Telefone_comercio", telefone_comercio);
58.                j.putExtra("Categoria", categoria[position]);//envia a categoria
    para ser salvo no banco de dados
59.                startActivity(j);
60.             }
61.         });
62.     };
63. }
```

C.3 Classe CCadastraProduto

```
1.  package com.example.tcc_comerciante;
2.
3.  import java.text.SimpleDateFormat;
4.  import java.util.Date;
5.  import android.app.Activity;
6.  import android.app.AlertDialog;
7.  import android.content.Context;
8.  import android.content.Intent;
9.  import android.database.Cursor;
10. import android.database.sqlite.SQLiteDatabase;
11. import android.location.Location;
12. import android.location.LocationListener;
13. import android.location.LocationManager;
14. import android.os.Bundle;
15. import android.view.View;
16. import android.widget.Button;
17. import android.widget.EditText;
18. import android.widget.Toast;
19.
20. public class CCadastraProduto extends  Activity
21. {
22.        SQLiteDatabase mmr = null;
23.        Cursor cursor;
24.        EditText
    nome_produto,preco_produto,data_inicio,data_termino,quantidade_itens;
25.        double latitude, longitude;
26.        String Nome_comercio,Telefone_comercio,Endereco_comercio;
27.        String Categoria_Tela_Anterior,nome;
28.
29.        @Override
30.        public void onCreate(Bundle savedInstanceState)
31.        {
32.        Intent i = getIntent();
33.        Nome_comercio = i.getStringExtra("Nome_comercio");//traz o id do usuario
    da outra tela
34.               Endereco_comercio=i.getStringExtra("Endereco_comercio");
35.            Telefone_comercio=i.getStringExtra("Telefone_comercio");
36.            Categoria_Tela_Anterior = i.getStringExtra("Categoria");//traz o id
    do usuario da outra tela
37.
38.
39.            super.onCreate(savedInstanceState);
40.               setContentView(R.layout.cadastra_dados_produto);
41.
42.               nome_produto=(EditText)findViewById(R.id.produto);//pega o nome
    que o usuario digita no campo produto
```

```
43.               preco_produto=(EditText)findViewById(R.id.preco);//pega o preco
   que o usuario digita no campo preço
44.               data_inicio=(EditText)findViewById(R.id.data_inicio_promocao);
45.               data_termino=(EditText)findViewById(R.id.data_fim_promocao);
46.
          quantidade_itens=(EditText)findViewById(R.id.quantidade_de_itens_em_promo
   cao);
47.
48.               //Grava_Dados();
49.
50.               IniciarServico();
51.       }
52.
53.       public void IniciarServico()
54.    {
55.               LocationManager locationManager =
   (LocationManager)getSystemService(Context.LOCATION_SERVICE);
56.        LocationListener locationListener = new LocationListener()
57.        {
58.            public void onLocationChanged(Location location)
59.            {
60.               latitude = (location.getLatitude()*-1);//grava a latitude na
   variavel
61.                longitude = (location.getLongitude()*-1);//grava a longitude na
   variavel
62.                       Grava_Dados();
63.            }
64.            public void onStatusChanged(String provider, int status, Bundle
   extras) {}
65.            public void onProviderEnabled(String provider) {}
66.            public void onProviderDisabled(String provider) {}
67.          };
68.        locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0,
   0, locationListener);
69.    }
70.
71.       public void Grava_Dados()
72.       {
73.               Button gravar_banco;
74.               gravar_banco = (Button)findViewById(R.id.Gravar);
75.               gravar_banco.setOnClickListener(new View.OnClickListener()
76.               {
77.                       public void onClick(View v)
78.                       {
79.                              Nome_comercio=Nome_comercio.replace(" ", "_");
80.                           Endereco_comercio=Endereco_comercio.replace(" ","_");
81.                              nome=nome_produto.getText().toString().replace(" ",
   "_");
82.                              Cria_Banco();
83.                       }
84.               });
85.       }
86.
87.       public void Cria_Banco()
88.       {
89.               try
90.               {
91.                       mmr = openOrCreateDatabase("mmr",MODE_WORLD_READABLE,
   null);
92.                       String sql = "CREATE TABLE IF NOT EXISTS oferta"
93.                                     +"(data_inicio_promocao String
   ,data_fim_promocao String ,nome_estabelecimento String,categoria String, latitude
   double, " +
94.                                     "longitude double,nome_produto String, preco
   float,quantidade String,telefone_mercado String,endereco_mercado String);";
95.                       mmr.execSQL(sql);
96.                       Insere_Dados_Banco();
97.               } catch (Exception e)
98.               {
99.                       Mensagem(e.getMessage(),"ERRO");
100.              }
101.      };
102.
103.      public void Insere_Dados_Banco()
104.      {
105.                      cursor = mmr.query("oferta", null, null, null,
   null, null, null);
```

```
106.              try
107.              {
108.                     String sql = "INSERT INTO
    oferta(data_inicio_promocao,data_fim_promocao,nome_estabelecimento,categoria,lati
    tude," +
109.
           "longitude,nome_produto,preco,quantidade,telefone_mercado,endereco_mercad
    o) values ("
110.                            +"'"+data_inicio.getText().toString()+"'"+","
111.                            +"'"+data_termino.getText().toString()+"'"+","
112.                            +"'"+Nome_comercio.toString()+"'"+","
113.                            +"'"+Categoria_Tela_Anterior.toString()+"'"+","
114.                            +latitude+","
115.                            +longitude+","
116.                            +"'"+nome.toString()+"',"
117.                            +"'"+preco_produto.getText().toString()+"',"
118.                            +"'"+quantidade_itens.getText().toString()+"',"
119.                            +"'"+Telefone_comercio.toString()+"',"
120.                            +"'"+Endereco_comercio.toString()+"')";
121.                     mmr.execSQL(sql);
122.
123.                     Toast.makeText(getApplicationContext(),"Dados salvos com
    Sucesso ", Toast.LENGTH_SHORT).show();
124.                     mmr.close();
125.                     Volta_Tela();
126.              } catch (Exception erro)
127.              {
128.                     Mensagem("Erro ao inserir  : "+erro.getMessage(), "ERRO");
129.              }
130.              mmr.close();
131.       };
132.
133.       public void Volta_Tela()
134.       {
135.              Intent k = new Intent(CCadastraProduto.this ,Sincronizar.class);
136.       startActivity(k);
137.       };
138.
139.       public void Mensagem (String msg, String titulo)
140.       {
141.              AlertDialog.Builder msg1 = new
    AlertDialog.Builder(CCadastraProduto.this);
142.              msg1.setTitle(titulo);
143.              msg1.setMessage(msg);
144.              msg1.setNeutralButton("OK", null);
145.              msg1.show();
146.       };
147.}
```

C.4 Classe Sincronizar

```
1.  package com.example.tcc_comerciante;
2.
3.  import java.io.DataInputStream;
4.  import java.io.DataOutputStream;
5.  import java.io.IOException;
6.  import java.net.Socket;
7.
8.  import android.app.Activity;
9.  import android.app.AlertDialog;
10. import android.content.Intent;
11. import android.database.Cursor;
12. import android.database.sqlite.SQLiteDatabase;
13. import android.os.Bundle;
14. import android.os.StrictMode;
15. import android.widget.Button;
16. import android.widget.ListView;
17. import android.widget.Toast;
18.
19. public class Sincronizar extends Activity
20. {
21.        SQLiteDatabase mmr = null;
22.        Cursor cursor;
23.
```

```
24.         String
    data_inicial_banco,data_final_banco,nome_estabelecimento_banco,categoria_banco,la
    titude_banco,longitude_banco;
25.         String nome_produto_banco, preco_banco, quantidade_banco, telefone_banco,
    endereco_banco;
26.
27.         String concatena;
28.         Socket socket= null;
29.
30.         DataOutputStream dataoutput = null;
31.         DataInputStream datainput = null;
32.
33.         public void onCreate(Bundle savedInstanceState)
34.         {
35.                 super.onCreate(savedInstanceState);
36.
37.                 StrictMode.ThreadPolicy policy = new
    StrictMode.ThreadPolicy.Builder().permitNetwork().build();
38.                 StrictMode.setThreadPolicy(policy);
39.
40.                 Ler_Dados();
41.
42.          };
43.          public void Ler_Dados()
44.          {
45.                 mmr = openOrCreateDatabase("mmr",MODE_WORLD_READABLE, null);
46.                 String colunas[] =
    {"data_inicio_promocao","data_fim_promocao","nome_estabelecimento","categoria","l
    atitude","longitude",
47.
          "nome_produto","preco","quantidade","telefone_mercado","endereco_mercado"
    };
48.                 cursor = mmr.query("oferta", colunas, null, null, null, null,
    null);
49.                 try
50.                 {
51.                         int numeroRegistro = cursor.getCount();
52.                         if (numeroRegistro > 0)
53.                         {
54.                                 cursor.moveToPosition(numeroRegistro-1);
55.
          data_inicial_banco=cursor.getString(cursor.getColumnIndex("data_inicio_pr
    omocao"));
56.
          data_final_banco=cursor.getString(cursor.getColumnIndex("data_fim_promoca
    o"));
57.
          nome_estabelecimento_banco=cursor.getString(cursor.getColumnIndex("nome_e
    stabelecimento"));
58.
          categoria_banco=cursor.getString(cursor.getColumnIndex("categoria"));
59.
          latitude_banco=cursor.getString(cursor.getColumnIndex("latitude"));
60.
          longitude_banco=cursor.getString(cursor.getColumnIndex("longitude"));
61.
          nome_produto_banco=cursor.getString(cursor.getColumnIndex("nome_produto")
    );
62.
          preco_banco=cursor.getString(cursor.getColumnIndex("preco"));
63.
          quantidade_banco=cursor.getString(cursor.getColumnIndex("quantidade"));
64.
          telefone_banco=cursor.getString(cursor.getColumnIndex("telefone_mercado")
    );
65.
          endereco_banco=cursor.getString(cursor.getColumnIndex("endereco_mercado")
    );
66.
67.                                 concatena=data_inicial_banco;
68.                                 concatena=concatena.concat(" ");
69.                                 concatena=concatena.concat(data_final_banco);
70.                                 concatena=concatena.concat(" ");
71.
          concatena=concatena.concat(nome_estabelecimento_banco);
72.                                 concatena=concatena.concat(" ");
73.                                 concatena=concatena.concat(categoria_banco);
```

```
74.                             concatena=concatena.concat(" ");
75.                             concatena=concatena.concat(latitude_banco);
76.                             concatena=concatena.concat(" ");
77.                             concatena=concatena.concat(longitude_banco);
78.                             concatena=concatena.concat(" ");
79.                             concatena=concatena.concat(nome_produto_banco);
80.                             concatena=concatena.concat(" ");
81.                             concatena=concatena.concat(preco_banco);
82.                             concatena=concatena.concat(" ");
83.                             concatena=concatena.concat(quantidade_banco);
84.                             concatena=concatena.concat(" ");
85.                             concatena=concatena.concat(telefone_banco);
86.                             concatena=concatena.concat(" ");
87.                             concatena=concatena.concat(endereco_banco);
88.
89.                             Enviar();
90.                             }
91.                         }catch(Exception erro)
92.                         {
93.                                 Mensagem("Erro ao inserir: "+erro.getMessage(),
    "ERRO");
94.                         }
95.         }
96.         public void Enviar()
97.         {
98.                         try
99.                         {
100.                            socket = new Socket("192.168.1.102",4001);
101.
102.                            dataoutput= new
    DataOutputStream(socket.getOutputStream());
103.                            datainput= new
    DataInputStream(socket.getInputStream());
104.                            dataoutput.writeUTF(concatena .toString());
105.                            Finalizar();
106.                        }catch(Exception e)
107.                            {
108.                                    e.printStackTrace();
109.                            }
110.        };
111.        public void Finalizar() throws IOException
112.        {
113.                        mmr.close();
114.                        socket.close();
115.                        dataoutput.close();
116.                        datainput.close();
117.                        Intent i = new Intent(Sincronizar.this
    ,ListaCategoria.class);
118.                        i.putExtra("Nome_comercio",
    nome_estabelecimento_banco.toString());//envia o nome do comercio para a tela que
    lista as categorias
119.                        i.putExtra("Endereco_comercio", endereco_banco.toString());
120.                        i.putExtra("Telefone_comercio", telefone_banco.toString());
121.                        startActivity(i);
122.        };
123.        public void Mensagem (String msg, String titulo)
124.        {
125.                AlertDialog.Builder msg1 = new
    AlertDialog.Builder(Sincronizar.this);
126.                msg1.setTitle(titulo);
127.                msg1.setMessage(msg);
128.                msg1.setNeutralButton("OK", null);
129.                msg1.show();
130.        };
131.}
```

Anexo D

Classes do aplicativo do Cliente

D.1 Classe Inicia_Servidor

```
1.  package com.example.tcc_comerciante;
2.
3.  import java.text.SimpleDateFormat;
4.  import java.util.Date;
5.  import android.app.Activity;
6.  import android.app.AlertDialog;
7.  import android.content.Context;
8.  import android.content.Intent;
9.  import android.database.Cursor;
10. import android.database.sqlite.SQLiteDatabase;
11. import android.location.Location;
12. import android.location.LocationListener;
13. import android.location.LocationManager;
14. import android.os.Bundle;
15. import android.view.View;
16. import android.widget.Button;
17. import android.widget.EditText;
18. import android.widget.Toast;
19.
20. public class CCadastraProduto extends  Activity
21. {
22.        SQLiteDatabase mmr = null;
23.        Cursor cursor;
24.        EditText
    nome_produto,preco_produto,data_inicio,data_termino,quantidade_itens;
25.        double latitude, longitude;
26.        String Nome_comercio,Telefone_comercio,Endereco_comercio;
27.        String Categoria_Tela_Anterior,nome;
28.
29.        @Override
30.        public void onCreate(Bundle savedInstanceState)
31.        {
32.        Intent i = getIntent();
33.        Nome_comercio = i.getStringExtra("Nome_comercio");//traz o id do usuario
    da outra tela
34.                Endereco_comercio=i.getStringExtra("Endereco_comercio");
35.            Telefone_comercio=i.getStringExtra("Telefone_comercio");
36.            Categoria_Tela_Anterior = i.getStringExtra("Categoria");//traz o id
    do usuario da outra tela
37.
38.
39.            super.onCreate(savedInstanceState);
40.                setContentView(R.layout.cadastra_dados_produto);
41.
42.                nome_produto=(EditText)findViewById(R.id.produto);//pega o nome
    que o usuario digita no campo produto
43.                preco_produto=(EditText)findViewById(R.id.preco);//pega o preco
    que o usuario digita no campo preço
44.                data_inicio=(EditText)findViewById(R.id.data_inicio_promocao);
45.                data_termino=(EditText)findViewById(R.id.data_fim_promocao);
46.
          quantidade_itens=(EditText)findViewById(R.id.quantidade_de_itens_em_promo
    cao);
47.
48.                //Grava_Dados();
49.
50.                IniciarServico();
51.        }
52.
53.        public void IniciarServico()
54.    {
55.                LocationManager locationManager =
    (LocationManager)getSystemService(Context.LOCATION_SERVICE);
56.         LocationListener locationListener = new LocationListener()
57.         {
```

```
58.             public void onLocationChanged(Location location)
59.             {
60.                latitude = (location.getLatitude()*-1);//grava a latitude na
   variavel
61.                 longitude = (location.getLongitude()*-1);//grava a longitude na
   variavel
62.                       Grava_Dados();
63.             }
64.             public void onStatusChanged(String provider, int status, Bundle
   extras) {}
65.             public void onProviderEnabled(String provider) {}
66.             public void onProviderDisabled(String provider) {}
67.           };
68.         locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0,
   0, locationListener);
69.     }
70.
71.        public void Grava_Dados()
72.        {
73.               Button gravar_banco;
74.               gravar_banco = (Button)findViewById(R.id.Gravar);
75.               gravar_banco.setOnClickListener(new View.OnClickListener()
76.               {
77.                      public void onClick(View v)
78.                      {
79.                             Nome_comercio=Nome_comercio.replace(" ", "_");
80.                          Endereco_comercio=Endereco_comercio.replace(" ","_");
81.                             nome=nome_produto.getText().toString().replace(" ",
   "_");
82.                             Cria_Banco();
83.                      }
84.               });
85.        }
86.
87.        public void Cria_Banco()
88.        {
89.               try
90.               {
91.                      mmr = openOrCreateDatabase("mmr",MODE_WORLD_READABLE,
   null);
92.                      String sql = "CREATE TABLE IF NOT EXISTS oferta"
93.                                    +"(data_inicio_promocao String
   ,data_fim_promocao String ,nome_estabelecimento String,categoria String, latitude
   double, " +
94.                                    "longitude double,nome_produto String, preco
   float,quantidade String,telefone_mercado String,endereco_mercado String);";
95.                      mmr.execSQL(sql);
96.                      Insere_Dados_Banco();
97.               } catch (Exception e)
98.               {
99.                      Mensagem(e.getMessage(),"ERRO");
100.              }
101.       };
102.
103.       public void Insere_Dados_Banco()
104.       {
105.                           cursor = mmr.query("oferta", null, null, null,
   null, null, null);
106.              try
107.              {
108.                     String sql = "INSERT INTO
   oferta(data_inicio_promocao,data_fim_promocao,nome_estabelecimento,categoria,lati
   tude," +
109.
          "longitude,nome_produto,preco,quantidade,telefone_mercado,endereco_mercad
   o) values ("
110.                           +"'"+data_inicio.getText().toString()+"'"+","
111.                           +"'"+data_termino.getText().toString()+"'"+","
112.                           +"'"+Nome_comercio.toString()+"'"+","
113.                           +"'"+Categoria_Tela_Anterior.toString()+"'"+","
114.                           +latitude+","
115.                           +longitude+","
116.                           +"'"+nome.toString()+"',"
117.                           +"'"+preco_produto.getText().toString()+"',"
118.                           +"'"+quantidade_itens.getText().toString()+"',"
119.                           +"'"+Telefone_comercio.toString()+"',"
120.                           +"'"+Endereco_comercio.toString()+"')";
```

```
121.                     mmr.execSQL(sql);
122.
123.                     Toast.makeText(getApplicationContext(),"Dados salvos com
    Sucesso ", Toast.LENGTH_SHORT).show();
124.                     mmr.close();
125.                     Volta_Tela();
126.             } catch (Exception erro)
127.             {
128.                     Mensagem("Erro ao inserir  : "+erro.getMessage(), "ERRO");
129.             }
130.             mmr.close();
131.       };
132.
133.       public void Volta_Tela()
134.       {
135.             Intent k = new Intent(CCadastraProduto.this ,Sincronizar.class);
136.       startActivity(k);
137.       };
138.
139.       public void Mensagem (String msg, String titulo)
140.       {
141.             AlertDialog.Builder msg1 = new
    AlertDialog.Builder(CCadastraProduto.this);
142.             msg1.setTitle(titulo);
143.             msg1.setMessage(msg);
144.             msg1.setNeutralButton("OK", null);
145.             msg1.show();
146.       };
147.}
```

D.2 Classe DetalhesProduto

```
1.  package com.example.tcc_comerciante;
2.
3.  import java.text.SimpleDateFormat;
4.  import java.util.Date;
5.  import android.app.Activity;
6.  import android.app.AlertDialog;
7.  import android.content.Context;
8.  import android.content.Intent;
9.  import android.database.Cursor;
10. import android.database.sqlite.SQLiteDatabase;
11. import android.location.Location;
12. import android.location.LocationListener;
13. import android.location.LocationManager;
14. import android.os.Bundle;
15. import android.view.View;
16. import android.widget.Button;
17. import android.widget.EditText;
18. import android.widget.Toast;
19.
20. public class CCadastraProduto extends  Activity
21. {
22.       SQLiteDatabase mmr = null;
23.       Cursor cursor;
24.       EditText
    nome_produto,preco_produto,data_inicio,data_termino,quantidade_itens;
25.       double latitude, longitude;
26.       String Nome_comercio,Telefone_comercio,Endereco_comercio;
27.       String Categoria_Tela_Anterior,nome;
28.
29.       @Override
30.       public void onCreate(Bundle savedInstanceState)
31.       {
32.       Intent i = getIntent();
33.       Nome_comercio = i.getStringExtra("Nome_comercio");//traz o id do usuario
    da outra tela
34.             Endereco_comercio=i.getStringExtra("Endereco_comercio");
35.          Telefone_comercio=i.getStringExtra("Telefone_comercio");
36.          Categoria_Tela_Anterior = i.getStringExtra("Categoria");//traz o id
    do usuario da outra tela
37.
38.
39.          super.onCreate(savedInstanceState);
40.             setContentView(R.layout.cadastra_dados_produto);
```

```
41.
42.                nome_produto=(EditText)findViewById(R.id.produto);//pega o nome
   que o usuario digita no campo produto
43.                preco_produto=(EditText)findViewById(R.id.preco);//pega o preco
   que o usuario digita no campo preço
44.                data_inicio=(EditText)findViewById(R.id.data_inicio_promocao);
45.                data_termino=(EditText)findViewById(R.id.data_fim_promocao);
46.
          quantidade_itens=(EditText)findViewById(R.id.quantidade_de_itens_em_promo
   cao);
47.
48.                //Grava_Dados();
49.
50.                IniciarServico();
51.        }
52.
53.        public void IniciarServico()
54.     {
55.                LocationManager locationManager =
   (LocationManager)getSystemService(Context.LOCATION_SERVICE);
56.         LocationListener locationListener = new LocationListener()
57.         {
58.             public void onLocationChanged(Location location)
59.             {
60.                latitude = (location.getLatitude()*-1);//grava a latitude na
   variavel
61.                 longitude = (location.getLongitude()*-1);//grava a longitude na
   variavel
62.                       Grava_Dados();
63.             }
64.             public void onStatusChanged(String provider, int status, Bundle
   extras) {}
65.             public void onProviderEnabled(String provider) {}
66.             public void onProviderDisabled(String provider) {}
67.           };
68.         locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0,
   0, locationListener);
69.     }
70.
71.        public void Grava_Dados()
72.        {
73.                Button gravar_banco;
74.                gravar_banco = (Button)findViewById(R.id.Gravar);
75.                gravar_banco.setOnClickListener(new View.OnClickListener()
76.                {
77.                       public void onClick(View v)
78.                       {
79.                              Nome_comercio=Nome_comercio.replace(" ", "_");
80.                           Endereco_comercio=Endereco_comercio.replace(" ","_");
81.                              nome=nome_produto.getText().toString().replace(" ",
   "_");
82.                              Cria_Banco();
83.                       }
84.                });
85.        }
86.
87.        public void Cria_Banco()
88.        {
89.                try
90.                {
91.                       mmr = openOrCreateDatabase("mmr",MODE_WORLD_READABLE,
   null);
92.                       String sql = "CREATE TABLE IF NOT EXISTS oferta"
93.                                     +"(data_inicio_promocao String
   ,data_fim_promocao String ,nome_estabelecimento String,categoria String, latitude
   double, " +
94.                                     "longitude double,nome_produto String, preco
   float,quantidade String,telefone_mercado String,endereco_mercado String);";
95.                       mmr.execSQL(sql);
96.                       Insere_Dados_Banco();
97.                } catch (Exception e)
98.                {
99.                       Mensagem(e.getMessage(),"ERRO");
100.               }
101.       };
102.
103.       public void Insere_Dados_Banco()
```

```
104.       {
105.                               cursor = mmr.query("oferta", null, null, null,
    null, null, null);
106.               try
107.               {
108.                       String sql = "INSERT INTO
    oferta(data_inicio_promocao,data_fim_promocao,nome_estabelecimento,categoria,lati
    tude," +
109.
           "longitude,nome_produto,preco,quantidade,telefone_mercado,endereco_mercad
    o) values ("
110.                               +"'"+data_inicio.getText().toString()+"'"+","
111.                               +"'"+data_termino.getText().toString()+"'"+","
112.                               +"'"+Nome_comercio.toString()+"'"+","
113.                               +"'"+Categoria_Tela_Anterior.toString()+"'"+","
114.                               +latitude+","
115.                               +longitude+","
116.                               +"'"+nome.toString()+"',"
117.                               +"'"+preco_produto.getText().toString()+"',"
118.                               +"'"+quantidade_itens.getText().toString()+"',"
119.                               +"'"+Telefone_comercio.toString()+"',"
120.                               +"'"+Endereco_comercio.toString()+"')";
121.                       mmr.execSQL(sql);
122.
123.                       Toast.makeText(getApplicationContext(),"Dados salvos com
    Sucesso ", Toast.LENGTH_SHORT).show();
124.                       mmr.close();
125.                       Volta_Tela();
126.               } catch (Exception erro)
127.               {
128.                       Mensagem("Erro ao inserir  : "+erro.getMessage(), "ERRO");
129.               }
130.               mmr.close();
131.       };
132.
133.       public void Volta_Tela()
134.       {
135.               Intent k = new Intent(CCadastraProduto.this ,Sincronizar.class);
136.       startActivity(k);
137.       };
138.
139.       public void Mensagem (String msg, String titulo)
140.       {
141.               AlertDialog.Builder msg1 = new
    AlertDialog.Builder(CCadastraProduto.this);
142.               msg1.setTitle(titulo);
143.               msg1.setMessage(msg);
144.               msg1.setNeutralButton("OK", null);
145.               msg1.show();
146.       };
147.}
```

D.3 Classe ListaPromocoes

```
1.  package com.example.tcc_comerciante;
2.
3.  import java.text.SimpleDateFormat;
4.  import java.util.Date;
5.  import android.app.Activity;
6.  import android.app.AlertDialog;
7.  import android.content.Context;
8.  import android.content.Intent;
9.  import android.database.Cursor;
10. import android.database.sqlite.SQLiteDatabase;
11. import android.location.Location;
12. import android.location.LocationListener;
13. import android.location.LocationManager;
14. import android.os.Bundle;
15. import android.view.View;
16. import android.widget.Button;
17. import android.widget.EditText;
18. import android.widget.Toast;
19.
20. public class CCadastraProduto extends  Activity
21. {
```

```
22.        SQLiteDatabase mmr = null;
23.        Cursor cursor;
24.        EditText
   nome_produto,preco_produto,data_inicio,data_termino,quantidade_itens;
25.        double latitude, longitude;
26.        String Nome_comercio,Telefone_comercio,Endereco_comercio;
27.        String Categoria_Tela_Anterior,nome;
28.
29.        @Override
30.        public void onCreate(Bundle savedInstanceState)
31.        {
32.        Intent i = getIntent();
33.        Nome_comercio = i.getStringExtra("Nome_comercio");//traz o id do usuario
   da outra tela
34.                Endereco_comercio=i.getStringExtra("Endereco_comercio");
35.            Telefone_comercio=i.getStringExtra("Telefone_comercio");
36.            Categoria_Tela_Anterior = i.getStringExtra("Categoria");//traz o id
   do usuario da outra tela
37.
38.
39.            super.onCreate(savedInstanceState);
40.                setContentView(R.layout.cadastra_dados_produto);
41.
42.                nome_produto=(EditText)findViewById(R.id.produto);//pega o nome
   que o usuario digita no campo produto
43.                preco_produto=(EditText)findViewById(R.id.preco);//pega o preco
   que o usuario digita no campo preço
44.                data_inicio=(EditText)findViewById(R.id.data_inicio_promocao);
45.                data_termino=(EditText)findViewById(R.id.data_fim_promocao);
46.
        quantidade_itens=(EditText)findViewById(R.id.quantidade_de_itens_em_promo
   cao);
47.
48.                //Grava_Dados();
49.
50.                IniciarServico();
51.        }
52.
53.        public void IniciarServico()
54.    {
55.                LocationManager locationManager =
   (LocationManager)getSystemService(Context.LOCATION_SERVICE);
56.         LocationListener locationListener = new LocationListener()
57.         {
58.             public void onLocationChanged(Location location)
59.             {
60.                latitude = (location.getLatitude()*-1);//grava a latitude na
   variavel
61.                 longitude = (location.getLongitude()*-1);//grava a longitude na
   variavel
62.                       Grava_Dados();
63.             }
64.             public void onStatusChanged(String provider, int status, Bundle
   extras) {}
65.             public void onProviderEnabled(String provider) {}
66.             public void onProviderDisabled(String provider) {}
67.           };
68.         locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0,
   0, locationListener);
69.    }
70.
71.        public void Grava_Dados()
72.        {
73.                Button gravar_banco;
74.                gravar_banco = (Button)findViewById(R.id.Gravar);
75.                gravar_banco.setOnClickListener(new View.OnClickListener()
76.                {
77.                        public void onClick(View v)
78.                        {
79.                                Nome_comercio=Nome_comercio.replace(" ", "_");
80.                            Endereco_comercio=Endereco_comercio.replace(" ","_");
81.                                nome=nome_produto.getText().toString().replace(" ",
   "_");
82.                                Cria_Banco();
83.                        }
84.                });
85.        }
```

```
86.
87.         public void Cria_Banco()
88.         {
89.                 try
90.                 {
91.                         mmr = openOrCreateDatabase("mmr",MODE_WORLD_READABLE,
    null);
92.                         String sql = "CREATE TABLE IF NOT EXISTS oferta"
93.                                         +"(data_inicio_promocao String
    ,data_fim_promocao String ,nome_estabelecimento String,categoria String, latitude
    double, " +
94.                                         "longitude double,nome_produto String, preco
    float,quantidade String,telefone_mercado String,endereco_mercado String);";
95.                         mmr.execSQL(sql);
96.                         Insere_Dados_Banco();
97.                 } catch (Exception e)
98.                 {
99.                         Mensagem(e.getMessage(),"ERRO");
100.                }
101.        };
102.
103.        public void Insere_Dados_Banco()
104.        {
105.                                cursor = mmr.query("oferta", null, null, null,
    null, null, null);
106.                try
107.                {
108.                        String sql = "INSERT INTO
    oferta(data_inicio_promocao,data_fim_promocao,nome_estabelecimento,categoria,lati
    tude," +
109.
           "longitude,nome_produto,preco,quantidade,telefone_mercado,endereco_mercad
    o) values ("
110.                                +"'"+data_inicio.getText().toString()+"'"+","
111.                                +"'"+data_termino.getText().toString()+"'"+","
112.                                +"'"+Nome_comercio.toString()+"'"+","
113.                                +"'"+Categoria_Tela_Anterior.toString()+"'"+","
114.                                +latitude+","
115.                                +longitude+","
116.                                +"'"+nome.toString()+"',"
117.                                +"'"+preco_produto.getText().toString()+"',"
118.                                +"'"+quantidade_itens.getText().toString()+"',"
119.                                +"'"+Telefone_comercio.toString()+"',"
120.                                +"'"+Endereco_comercio.toString()+"')";
121.                        mmr.execSQL(sql);
122.
123.                        Toast.makeText(getApplicationContext(),"Dados salvos com
    Sucesso ", Toast.LENGTH_SHORT).show();
124.                        mmr.close();
125.                        Volta_Tela();
126.                } catch (Exception erro)
127.                {
128.                        Mensagem("Erro ao inserir  : "+erro.getMessage(), "ERRO");
129.                }
130.                mmr.close();
131.        };
132.
133.        public void Volta_Tela()
134.        {
135.                Intent k = new Intent(CCadastraProduto.this ,Sincronizar.class);
136.        startActivity(k);
137.        };
138.
139.        public void Mensagem (String msg, String titulo)
140.        {
141.                AlertDialog.Builder msg1 = new
    AlertDialog.Builder(CCadastraProduto.this);
142.                msg1.setTitle(titulo);
143.                msg1.setMessage(msg);
144.                msg1.setNeutralButton("OK", null);
145.                msg1.show();
146.        };
147.}
```

D.4 Classe Promocoes

```
1.  package com.example.tcc_comerciante;
2.
3.  import java.text.SimpleDateFormat;
4.  import java.util.Date;
5.  import android.app.Activity;
6.  import android.app.AlertDialog;
7.  import android.content.Context;
8.  import android.content.Intent;
9.  import android.database.Cursor;
10. import android.database.sqlite.SQLiteDatabase;
11. import android.location.Location;
12. import android.location.LocationListener;
13. import android.location.LocationManager;
14. import android.os.Bundle;
15. import android.view.View;
16. import android.widget.Button;
17. import android.widget.EditText;
18. import android.widget.Toast;
19.
20. public class CCadastraProduto extends  Activity
21. {
22.        SQLiteDatabase mmr = null;
23.        Cursor cursor;
24.        EditText
    nome_produto,preco_produto,data_inicio,data_termino,quantidade_itens;
25.        double latitude, longitude;
26.        String Nome_comercio,Telefone_comercio,Endereco_comercio;
27.        String Categoria_Tela_Anterior,nome;
28.
29.        @Override
30.        public void onCreate(Bundle savedInstanceState)
31.        {
32.        Intent i = getIntent();
33.        Nome_comercio = i.getStringExtra("Nome_comercio");//traz o id do usuario
    da outra tela
34.                Endereco_comercio=i.getStringExtra("Endereco_comercio");
35.            Telefone_comercio=i.getStringExtra("Telefone_comercio");
36.            Categoria_Tela_Anterior = i.getStringExtra("Categoria");//traz o id
    do usuario da outra tela
37.
38.
39.            super.onCreate(savedInstanceState);
40.                setContentView(R.layout.cadastra_dados_produto);
41.
42.                nome_produto=(EditText)findViewById(R.id.produto);//pega o nome
    que o usuario digita no campo produto
43.                preco_produto=(EditText)findViewById(R.id.preco);//pega o preco
    que o usuario digita no campo preço
44.                data_inicio=(EditText)findViewById(R.id.data_inicio_promocao);
45.                data_termino=(EditText)findViewById(R.id.data_fim_promocao);
46.
         quantidade_itens=(EditText)findViewById(R.id.quantidade_de_itens_em_promo
    cao);
47.
48.                //Grava_Dados();
49.
50.                IniciarServico();
51.        }
52.
53.        public void IniciarServico()
54.     {
55.                LocationManager locationManager =
    (LocationManager)getSystemService(Context.LOCATION_SERVICE);
56.         LocationListener locationListener = new LocationListener()
57.         {
58.             public void onLocationChanged(Location location)
59.             {
60.                latitude = (location.getLatitude()*-1);//grava a latitude na
    variavel
61.                 longitude = (location.getLongitude()*-1);//grava a longitude na
    variavel
62.                       Grava_Dados();
63.             }
64.             public void onStatusChanged(String provider, int status, Bundle
    extras) {}
65.             public void onProviderEnabled(String provider) {}
66.             public void onProviderDisabled(String provider) {}
```

```
67.           };
68.          locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0,
    0, locationListener);
69.     }
70.
71.         public void Grava_Dados()
72.         {
73.                 Button gravar_banco;
74.                 gravar_banco = (Button)findViewById(R.id.Gravar);
75.                 gravar_banco.setOnClickListener(new View.OnClickListener()
76.                 {
77.                         public void onClick(View v)
78.                         {
79.                                 Nome_comercio=Nome_comercio.replace(" ", "_");
80.                             Endereco_comercio=Endereco_comercio.replace(" ","_");
81.                                 nome=nome_produto.getText().toString().replace(" ",
    "_");
82.                                 Cria_Banco();
83.                         }
84.                 });
85.         }
86.
87.         public void Cria_Banco()
88.         {
89.                 try
90.                 {
91.                         mmr = openOrCreateDatabase("mmr",MODE_WORLD_READABLE,
    null);
92.                         String sql = "CREATE TABLE IF NOT EXISTS oferta"
93.                                         +"(data_inicio_promocao String
    ,data_fim_promocao String ,nome_estabelecimento String,categoria String, latitude
    double, " +
94.                                         "longitude double,nome_produto String, preco
    float,quantidade String,telefone_mercado String,endereco_mercado String);";
95.                         mmr.execSQL(sql);
96.                         Insere_Dados_Banco();
97.                 } catch (Exception e)
98.                 {
99.                         Mensagem(e.getMessage(),"ERRO");
100.                }
101.        };
102.
103.        public void Insere_Dados_Banco()
104.        {
105.                            cursor = mmr.query("oferta", null, null, null,
    null, null, null);
106.                try
107.                {
108.                        String sql = "INSERT INTO
    oferta(data_inicio_promocao,data_fim_promocao,nome_estabelecimento,categoria,lati
    tude," +
109.
           "longitude,nome_produto,preco,quantidade,telefone_mercado,endereco_mercad
    o) values ("
110.                            +"'"+data_inicio.getText().toString()+"'"+","
111.                            +"'"+data_termino.getText().toString()+"'"+","
112.                            +"'"+Nome_comercio.toString()+"'"+","
113.                            +"'"+Categoria_Tela_Anterior.toString()+"'"+","
114.                            +latitude+","
115.                            +longitude+","
116.                            +"'"+nome.toString()+"',"
117.                            +"'"+preco_produto.getText().toString()+"',"
118.                            +"'"+quantidade_itens.getText().toString()+"',"
119.                            +"'"+Telefone_comercio.toString()+"',"
120.                            +"'"+Endereco_comercio.toString()+"')";
121.                        mmr.execSQL(sql);
122.
123.                        Toast.makeText(getApplicationContext(),"Dados salvos com
    Sucesso ", Toast.LENGTH_SHORT).show();
124.                        mmr.close();
125.                        Volta_Tela();
126.                } catch (Exception erro)
127.                {
128.                        Mensagem("Erro ao inserir  : "+erro.getMessage(), "ERRO");
129.                }
130.                mmr.close();
131.        };
```

```
132.
133.       public void Volta_Tela()
134.       {
135.              Intent k = new Intent(CCadastraProduto.this ,Sincronizar.class);
136.       startActivity(k);
137.       };
138.
139.       public void Mensagem (String msg, String titulo)
140.       {
141.              AlertDialog.Builder msg1 = new
    AlertDialog.Builder(CCadastraProduto.this);
142.              msg1.setTitle(titulo);
143.              msg1.setMessage(msg);
144.              msg1.setNeutralButton("OK", null);
145.              msg1.show();
146.       };
147.}
```

Anexo E

Classes do aplicativo Sevidor

E.1 Classe Server

```
1.  import java.io.DataInputStream;
2.  import java.io.DataOutputStream;
3.  import java.io.IOException;
4.  import java.net.ServerSocket;
5.  import java.net.Socket;
6.  //a classe server tem por finalidade realizar a operaçao de comunicaçao
7.  //entre os dois aplicativos, sendo assim, o que a classe realmente faz é
8.  // recebe os dados do mercado na variavel IN após isto é necessario enviar
9.  //os dados para a pessoa, isto é feito através da variavel OUT2
10. //as variaveis OUT e IN2 necessitam ser declaradas pois sempre que ha uma
11. //conexao entre cliente e servidor deve haver uma troca de mensagens entre eles
12. //com essa troca de mensagens é possivel determinar se houve ou nao a conexao.
13. //mesnsagens estas que sao interpretadas pela funçao accept();
14.
15. public class Server
16. {
17.         public static void main(String[] args) throws IOException
18.         {
19.                 ServerSocket serverSocket = new ServerSocket(4001);//porta de
    conexao com o mercado
20.                 ServerSocket serverSocket2 = new ServerSocket(4000);//porta de
    conexao com a pessoa
21.
22.                 Socket socket;//mercado
23.                 Socket socket2;//pessoa
24.
25.                 while(true)
26.                 {
27.                         System.out.println("esperando conexao comerciante");
28.                         socket= serverSocket.accept();//conexao com o mercado
29.
30.                         System.out.println("esperando conexao cliente");
31.                         socket2=serverSocket2.accept();//conexao com a pessoa
32.
33.                         System.out.println("conectado ao cliente e comerciante");
34.                         DataOutputStream out = new
    DataOutputStream(socket.getOutputStream()); //envia uma msg ao mercado
35.                         DataInputStream in = new
    DataInputStream(socket.getInputStream());//msg recebida do mercado
36.
37.                         DataOutputStream out2 = new
    DataOutputStream(socket2.getOutputStream()); //envia uma msg pessoa
38.                         DataInputStream in2 = new
    DataInputStream(socket2.getInputStream());//msg recebida da pessoa
39.
40.                         out2.writeBytes(in.readUTF());//leitura do IN que foi a
    mensagem recebida do mercado
41.                         out2.flush();//envia para o cliente
42.                         socket.close();
43.                         socket2.close();
44.                         in.close();
45.                         in2.close();
46.                         out.close();
47.                         out2.close();
48.                 }
49.         }
50. }
```