



**Universidade Federal de Sergipe  
Centro de Ciências Exatas e Tecnologia  
DEPARTAMENTO DE COMPUTAÇÃO - DCOMP  
CIÊNCIA DA COMPUTAÇÃO**

Documentos Desenvolvimento de Software

Prof. Dr. MICHEL DOS SANTOS SOARES

São Cristóvão, Sergipe  
Maio – 2016

**Componentes:**

ELTON MOREIRA CARVALHO – 201310004602

FERNNADO MELO NASCIMENTO – 201210009310

FERNNADO MESSIAS DOS SANTOS – 201320001408

RODRIGO BENEDITO OTONI – 201210009188

THALES FRANCISCO SOUZA SAMPAIO ALVES DOS SANTOS – 201210012648

# Conteúdo

<b>1 Levantamento de Requisitos</b>	<b>5</b>
1.1 Propósito do Documento . . . . .	5
1.2 Escopo do Produto . . . . .	5
1.3 Definições e Abreviações . . . . .	5
1.3.1 Definições . . . . .	5
1.3.2 Abreviações . . . . .	5
1.4 Referências . . . . .	6
1.5 Visão Geral do Restante do Documento . . . . .	6
1.6 Descrição Geral . . . . .	6
1.6.1 Perspectiva do Produto . . . . .	6
1.6.2 Funções do Produto . . . . .	6
1.6.3 Características do Usuário . . . . .	6
1.6.4 Restrições Gerais . . . . .	6
1.6.5 Suposições e Dependências . . . . .	7
1.7 Requisitos específicos . . . . .	7
1.7.1 Requisitos Funcionais . . . . .	7
1.7.2 Requisitos Não Funcionais . . . . .	10
<b>2 Plano de Projeto</b>	<b>11</b>
<b>3 Casos de Uso</b>	<b>12</b>
<b>4 Código do Projeto</b>	<b>15</b>
4.1 Modelos . . . . .	15
4.1.1 Model . . . . .	15
4.1.2 Comanda . . . . .	16
4.1.3 Compra . . . . .	17
4.1.4 Endereço . . . . .	17
4.1.5 Fornecedor . . . . .	18
4.1.6 Funcionário . . . . .	19
4.1.7 Garçom . . . . .	20
4.1.8 Gerente . . . . .	20
4.1.9 Ingrediente . . . . .	20
4.1.10 Item . . . . .	20
4.1.11 Mesa . . . . .	21
4.1.12 Pedido . . . . .	22
4.1.13 Produto . . . . .	23
4.2 Controles . . . . .	24
4.2.1 Controle de Atendimento . . . . .	24
4.2.2 Controle de Compra . . . . .	25
4.2.3 Controle de Fornecedor . . . . .	25
4.2.4 Controle de Funcionário . . . . .	25
4.2.5 Controle de Impressão . . . . .	25
4.2.6 Controle de Item . . . . .	25

4.2.7	Controle de Produto . . . . .	25
4.3	Fronteiras . . . . .	26
4.3.1	Main Activity . . . . .	26
4.3.2	Menu Principal . . . . .	26
4.3.3	Menu Fornecedores . . . . .	28
4.3.4	Menu Funcionários . . . . .	28
4.3.5	Menu Garçom . . . . .	29
4.3.6	Menu Itens . . . . .	29
4.3.7	Menu Produtos . . . . .	30
4.3.8	Menu Relatório . . . . .	30
4.3.9	Busca Fornecedor . . . . .	31
4.3.10	Busca Funcionário . . . . .	31
4.3.11	Busca Item . . . . .	31
4.3.12	Busca Produto . . . . .	31
4.3.13	Cadastro Endereço . . . . .	31
4.3.14	Cadastro Fornecedor . . . . .	32
4.3.15	Cadastro Funcionário . . . . .	32
4.3.16	Cadastro Item . . . . .	32
4.3.17	Cadastro Produto . . . . .	32
4.3.18	Criação Comanda . . . . .	32
4.3.19	Exibir Pedido . . . . .	33
4.3.20	Selecionar Ingredientes . . . . .	33
4.3.21	Tela Garçom . . . . .	33
4.3.22	Tela Relatórios . . . . .	34
4.4	Adapters . . . . .	34
4.4.1	Fornecedor Adapter . . . . .	34
4.4.2	Funcionario Adapter . . . . .	34
4.4.3	Item Adapter . . . . .	35
4.4.4	Produto Adapter . . . . .	36
4.5	Helpers . . . . .	37
4.5.1	Server Request . . . . .	37
4.5.2	Request Callback . . . . .	37
4.5.3	Utilitários . . . . .	37
4.6	Banco de Dados . . . . .	39
<b>5</b>	<b>Diagramas</b>	<b>42</b>
5.1	Diagrama de Classes . . . . .	42
5.2	Diagrama de Casos de Uso . . . . .	42

# 1 Levantamento de Requisitos

## 1.1 Propósito do Documento

Este documento contém a especificação de um sistema para controle e atendimento de um restaurante. Nas próximas seções, serão apresentadas de forma mais detalhada as características do sistema.

## 1.2 Escopo do Produto

O sistema destina-se à gerência de restaurantes, abrangendo os seguintes setores: financeiro, estoque, controle de funcionários e atendimento.

## 1.3 Definições e Abreviações

### 1.3.1 Definições

Item(ns): Cada mercadoria existente no estoque.

Produto(s): Mercadorias comercializadas pelo restaurante.

Pedidos: Requisição de produtos feitas pela comanda.

Comanda: Lista de pedidos vinculada a uma mesa.

Comanda Ativa: Comanda que permite a adesão de pedidos.

Comanda Fechada: Comanda que não permite a adesão de pedidos.

Restaurante: Estabelecimento comercial de venda de produtos.

Funcionário: Pessoa física com vínculo empregatício com o restaurante.

Relatório: Apresentação de um conjunto de informações específicas do restaurante.

Entrada de Itens: A aquisição de itens feita pelo restaurante, e inseridos no estoque.

Saída de Itens: A retirada de um item do estoque.

Arrecadação Líquida : Valor arrecadado subtraído dos gastos.

### 1.3.2 Abreviações

RF: Requisito Funcional.

RNF: Requisito Não Funcional.

Prioridade do requisito. A prioridade é medida em uma escala de 0 a 2, onde 0 indica a menor prioridade e 2 indica a maior prioridade.

## **1.4 Referências**

SOMMERVILLE, I. Engenharia de Software. Pearson/Prentice Hall. PRESSMAN, R. S. Engenharia de Software. McGraw Hill.

## **1.5 Visão Geral do Restante do Documento**

Nas próximas seções serão apresentadas as características gerais do sistema e suas funcionalidades. Segue ainda, a descrição de restrições e dependências que devem ser consideradas para o devido funcionamento. Atrelado às funcionalidades do sistema, é apresentada também uma lista de requisitos funcionais e não funcionais que servirão como guia para o desenvolvimento do sistema.

## **1.6 Descrição Geral**

### **1.6.1 Perspectiva do Produto**

Espera-se que o sistema seja desenvolvido, implementado e testado durante as disciplinas de Desenvolvimento de Software I, II e III da Universidade Federal de Sergipe.

### **1.6.2 Funções do Produto**

O produto permite a gestão de um restaurante, possibilitando o controle de estoque, controle do atendimento, gerenciando pedidos e comandas, permitindo também o controle financeiro e de funcionários.

### **1.6.3 Características do Usuário**

Segue abaixo a definição de cada tipo de usuário do sistema:

Garçom: Usuário responsável pelo atendimento, solicitação, alteração, exclusão e entrega de pedidos, bem como o fechamento de comandas.

Gerente: Usuário com acesso a todas as funcionalidades do sistema.

### **1.6.4 Restrições Gerais**

Os funcionários que utilizarão o sistema deverão ser treinados para que se tornem aptos para o uso do sistema.

O restaurante deverá possuir dispositivos móveis (Smartphone ou Tablet) com configuração mínima de: processador de 1GHz, memória RAM 1GB, espaço de armazenamento interno de 500 MB e sistema operacional Android 4.2.

O restaurante deverá possuir uma rede local WiFi (IEEE 802.11) disponível exclusivamente para os funcionários.

O restaurante deverá possuir um computador central onde ficarão todos os dados do sistema.

## **1.6.5 Suposições e Dependências**

Faz-se necessário para o funcionamento do software a existência de tablets ou smartphones para que os garçons utilizem no atendimento.

Rede WiFi para que os tablets ou smartphones se comuniquem com o sistema do restaurante, e um computador que funcione como servidor do sistema.

## **1.7 Requisitos específicos**

### **1.7.1 Requisitos Funcionais**

RF1 Inclusão de fornecedores. (Pr.: 2)

O sistema deve efetuar o cadastro dos fornecedores.

RF2 Alteração de fornecedores. (Pr.: 2)

O sistema deve efetuar a alteração dos dados cadastrais de fornecedores.

RF3 Exclusão de fornecedores. (Pr.: 2)

O sistema deve efetuar a exclusão de fornecedores.

RF4 Consulta de fornecedores. (Pr.: 2)

O sistema deve efetuar a consulta dos dados dos fornecedores.

RF5 Geração de relatório de fornecedores. (Pr.: 0)

O sistema deve gerar um relatório com os dados de todos os fornecedores.

RF6 Geração de relatórios de itens por fornecedor. (Pr.: 1)

O sistema deve efetuar a geração de relatórios de itens por fornecedor.

RF7 Inclusão dos itens. (Pr.: 2)

O sistema deve efetuar a inclusão dos dados dos itens fornecidos para o restaurante.

RF8 Alteração de itens. (Pr.: 2)

O sistema deve efetuar a alteração dos dados dos itens fornecidos para o restaurante.

RF9 Exclusão de itens. (Pr.: 2)

O sistema deve efetuar a exclusão dos itens fornecidos para o restaurante.

RF10 Consultar de itens. (Pr.: 2)

O sistema deve efetuar a consulta dos dados dos itens.

RF11 Geração de relatório de itens. (Pr.: 1)

O sistema deve gerar o relatório de todos os itens fornecidos por todos os fornecedores.

RF12 Aviso de quantidade de itens abaixo do limite delimitado. (Pr.: 1)

O sistema deve informar quando a quantidade de um item estiver abaixo do valor mínimo definido pelo gerente.

**RF13 Geração de relatório de Itens em falta.** (Pr.: 1)

O sistema deve efetuar geração de relatórios dos itens que estão em falta, ou seja, aqueles que a quantidade é igual a zero.

**RF14 Inclusão de comandas.** (Pr.: 2)

O sistema deve efetuar a inclusão de comandas, registrando a sua hora de abertura.

**RF15 Associar comanda a um funcionário.** (Pr.: 2)

O sistema deve associar uma comanda a um funcionário responsável.

**RF16 Encerramento de comandas.** (Pr.: 2)

O sistema deve efetuar o fechamento de comandas, incluindo a hora de encerramento.

**RF17 Alteração de comandas.** (Pr.: 2)

O sistema deve efetuar a alteração de comandas.

**RF18 Impressão de Pedidos** (Pr: 2)

O Sistema deve imprimir todos os pedidos adicionados às comandas.

**RF19 Gerar relatório de comandas ativas.** (Pr.: 1)

O sistema deve gerar o relatório de comandas ativas.

**RF20 Impressão de conta.** (Pr.: 2)

O sistema deve efetuar a impressão da conta.

**RF21 Consulta informações dos pedidos da comandas.** (Pr.: 2)

O sistema deve permitir a consulta de informações dos pedidos da comanda.

**RF22 Inclusão de pedido de produto na comanda.** (Pr.: 2)

O sistema deve efetuar a inclusão do pedido, incluindo a hora inicial do pedido.

**RF23 Entrega do pedido de Produto.** (Pr.: 2)

O sistema deve efetuar inclusão da hora da entrega do pedido.

**RF24 Cancelamento de pedidos.** (Pr.: 2)

O sistema deve permitir o cancelamento de pedidos que ainda não foram preparados.

**RF25 Informações de pedido.** (Pr.: 2)

O sistema deve permitir a consulta das informações dos pedidos.

**RF26 Opções de pagamento.** (Pr.: 2)

O sistema deverá informar sobre as opções de pagamento aceitas.

**RF27 Inclusão de produto.** (Pr.: 2)

O sistema deve efetuar o cadastro do produto.

RF28 Alteração de produto. (Pr.: 2)

O sistema deve efetuar a alteração do produto.

RF29 Exclusão de produto. (Pr.: 2)

O sistema deve efetuar a exclusão do produto.

RF30 Consulta de produto. (Pr.: 2)

O sistema deve efetuar a consulta de informações do produto.

RF31 Geração de relatório de itens por produto. (Pr.: 1)

O sistema deve efetuar a geração do relatório de itens que compõem um produto.

RF32 Validação do produto a partir dos itens. (Pr.: 1)

O sistema deve efetuar verificação da possibilidade de produção do produto a partir dos itens.

RF33 Inclusão de funcionários. (Pr.: 2)

O sistema deve efetuar o cadastro de funcionários.

RF34 Alteração de funcionários. (Pr.: 2)

O sistema deve efetuar a alteração de funcionários.

RF35 Demissão de funcionários. (Pr.: 2)

O sistema deve efetuar a demissão de funcionários.

RF36 Consulta de funcionários. (Pr.: 2)

O sistema deve efetuar consulta de funcionários.

RF37 Gerar relatório de funcionários (Pr.: 0)

O sistema deve gerar relatório com os dados de todos os funcionários.

RF38 Geração de relatórios das comandas encerradas. (Pr.: 0)

O sistema deve efetuar a geração de relatórios contendo as comandas encerradas do restaurante em um intervalo de tempo, em ordem de dias, definido pelo gerente.

RF39 Geração de relatórios de despesas. (Pr.: 1)

O Sistema deve efetuar a geração de relatórios contendo as despesas do restaurante em um intervalo de tempo, em ordem de dias, definido pelo gerente.

RF40 Geração de relatórios da arrecadação líquida. (Pr.: 1)

O Sistema deve efetuar a geração de relatórios contendo a arrecadação líquida do restaurante em um intervalo de tempo, em ordem de dias, definido pelo gerente.

### **1.7.2 Requisitos Não Funcionais**

RNF1 (Pr.: 1): O sistema deve retornar as consultas, ou seja, prover a exibição dos dados, em, no máximo, 6 segundos, em 90% dos casos.

RNF2 (Pr.: 1): O sistema deve processar a inclusão de dados em, no máximo, 8 segundos, em 90% dos casos.

RNF3 (Pr.: 0): O sistema deve processar a exclusão de dados em, no máximo, 8 segundos, em 90% dos casos.

RNF4 (Pr.: 1): O sistema deve gerar relatórios em, no máximo, 8 segundos, em 90% dos casos.

## **2 Plano de Projeto**

### 3 Casos de Uso

**Nome:** Cadastrar Funcionário.

**Descrição:** O Gerente cadastra um funcionário no sistema.

**Identificador:** CDU1.

**Importância:** 2.

**Autor Primário:** Gerente.

**Fluxo Principal:**

Sistema	Gerente	Funcionário
	1 - Solicita dados do funcionário	
		2 - Fornece dados ao Gerente
	3 - Insere os dados do funcionário no sistema	
4 - Solicita ao gerente confirmação dos dados		
	5a - Confirma dados 5b - teste	
6a - Registra os dados 6b - Finaliza a operação		
7a - Finaliza a operação		

**Nome:** Excluir Pedido.

**Descrição:** O Garçom remove o pedido da comanda mediante solicitação do Cliente.

**Identificador:** CDU2.

**Importância:** 2.

**Autor Primário:** Garçom.

**Pré-condições:** O pedido deve ter sido incluído na comanda.

**Fluxo Principal:**

Sistema	Garçom	Cliente
		1 - Solicita ao Garçom a exclusão do pedido
	2 - Consulta a comanda no sistema	
3 - Retorna a comanda consultada		
	4 - Solicita exclusão do pedido da comanda	
5 - Realiza exclusão		
6 - Finaliza operação		

**Nome:** Alterar Cadastro de Funcionário.

**Descrição:** O Gerente altera as informações de um funcionário no sistema.

**Identificador:** CDU3.

**Importância:** 2.

**Ator Primário:** Gerente.

**Fluxo Principal:**

Sistema	Gerente	Funcionário
	1 - Solicita identificação do funcionário	
		2 - Fornece identificação ao gerente
	3 - Solicita busca do cadastro do funcionário ao sistema	
4a - Retorna o cadastro do funcionário 4b - Retorna aviso que o funcionário não está cadastrado no sistema		
	5b - Solicita novos dados ao Funcionário	
		6b - Fornece dados ao Gerente
	7b - Solicita a alteração do cadastro do funcionário ao sistema	
8b - Altera os dados do funcionário no sistema		
9b - Finaliza operação		

**Nome:** Gerar relatório de itens em falta.

**Descrição:** O gerente gera o relatório de itens em falta.

**Identificador:** CDU4.

**Importância:** 1.

**Ator Primário:** Gerente.

**Fluxo Principal:**

Sistema	Gerente
	1 - Solicita o relatório de itens em falta
2 - Realiza busca pelos itens em falta	
3 - Gera o relatório de itens em falta	
4 - Exibe relatório gerado	
	5a - Solicita impressão do relatório gerado 5b - Solicita a gravação do relatório gerado 5c - Descarta o relatório gerado
6a - Envia relatório gerado para impressão 6b - Salva o relatório 6c - Exclui relatório	
7 - Finaliza operação	

**Nome:** Alertar sobre itens abaixo do limite.

**Descrição:** O gerente é alertado pelo sistema quando um item está abaixo do limite.

**Identificador:** CDU5.

**Importância:** 1.

**Autor Primário:** Gerente.

**Fluxo Principal:**

Sistema	Garçom	Gerente
	1 - Indica ao sistema que o pedido foi entregue	
2 - Sistema decrementa os itens dos produtos do pedido entregue		
3 - Compara a quantidade com o limite informado no cadastro do item		
4a - O sistema emite um alerta para o Gerente 4b - Finaliza a operação		
		5a - Recebe alerta de item abaixo do limite
6a - Finaliza operação		

# 4 Código do Projeto

## 4.1 Modelos

### 4.1.1 Model

```
1 package ds2.equipe1.restaurante.modelos;
2
3 import android.content.Context;
4 import android.util.Log;
5
6 import com.google.gson.Gson;
7 import com.google.gson.reflect.TypeToken;
8
9 import org.json.JSONException;
10 import org.json.JSONObject;
11
12 import java.lang.reflect.ParameterizedType;
13 import java.lang.reflect.Type;
14 import java.util.ArrayList;
15
16 import ds2.equipe1.restaurante.helpers.RequestCallback;
17 import ds2.equipe1.restaurante.helpers.ServerRequest;
18 import ds2.equipe1.restaurante.helpers.Utils;
19
20 /**
21 * Created by Fernando on 07/04/2016.
22 */
23
24 public class Model<T> {
25     //transient para a serializacao nao incluir.
26     private transient Context context;
27
28     private Integer id;
29
30     public Model(Context context){
31         this.context = context;
32     }
33
34     public void setContext(Context context){
35         this.context = context;
36     }
37
38     public void save(){
39         save(null);
40     }
41
42     public void save(final RequestCallback<Model> callback){
43         new ServerRequest(context).sendRequest(getControllerName(), ServerRequest.Action.SAVE, new Gson().toJson(this),
44             new RequestCallback<JSONObject>() {
45                 @Override
46                 public void execute(JSONObject json) throws Exception {
47                     setId(json.getInt("id"));
48                     if (callback != null){
49                         callback.execute(Model.this);
50                     }
51                     super.execute(json);
52                 }
53             });
54     }
55
56     public void delete(){
57         new ServerRequest(context).sendRequest(getControllerName(), ServerRequest.Action.DELETE, new Gson().toJson(this),
58             null);
59     }
60
61     public static <T extends Model> void find(Context context, Class<T> klass, final Type typeArray, final
62         RequestCallback<T> callback, int id){
63         find(context, klass, typeArray, callback, id + "");
64     }
65
66     public static <T extends Model> void find(final Context context, Class<T> klass, final Type typeArray, final
67         RequestCallback<T> callback, String where){
68         ServerRequest serverRequest = new ServerRequest(context);
69         String controller = ((Class<T>) ((ParameterizedType) klass.getGenericSuperclass()).getActualTypeArguments()[0])
70             .getSimpleName();
71         serverRequest.sendRequest(controller, ServerRequest.Action.FIND, where, new RequestCallback<JSONObject>() {
72             @Override
73             public void execute(JSONObject json) throws Exception {
74                 Log.w(Utils.TAG, "Trying to cast " + typeArray.toString());
75
76                 ArrayList<T> lista = new Gson().fromJson(json.getJSONArray("data").toString(), typeArray);
77                 for (T item : lista){
78                     item.setContext(context);
79                 }
80                 if (callback != null){
81                     if (lista.size() > 0)
82                         callback.execute(lista.get(0));
83                 }
84             }
85         });
86     }
87 }
```

```

79         }
80         callback.execute(lista);
81     }
82     super.execute(json);
83 }
84 });
85 }
86
87 //    public static <T extends Model> void find(Context context, Class<T> klass, RequestCallback callback){
88 //        find(context, klass, callback, "");
89 //    }
90
91     public Integer getId() {
92         return id;
93     }
94
95     public T setId(Integer id) {
96         this.id = id;
97         return (T) this;
98     }
99
100    protected String getControllerName(){
101        return this.getClass().getSimpleName();
102    }
103}

```

Código 1: Model.java

#### 4.1.2 Comanda

```

1 package ds2.equipe1.restaurante.modelos;
2
3 import android.content.Context;
4
5 import java.util.ArrayList;
6 import java.util.Date;
7
8 /**
9  * Created by Th on 24/03/2016.
10 */
11
12 public class Comanda extends Model<Comanda> {
13
14     private ArrayList<Pedido> pedidos = new ArrayList<>();
15     private String data;
16     private boolean ativa = false;
17     private float total = 0;
18     private String nome;
19
20     public Comanda(Context context, String nome, Pedido primeiro){
21         super(context);
22         Date agora = new Date();
23         pedidos.add(primeiro);
24         data = agora.toString();
25         this.nome = nome;
26         ativa = true;
27         total += primeiro.getCusto();
28     }
29
30     public void addPedido(Pedido outro){
31         pedidos.add(outro);
32         total += outro.getCusto();
33     }
34
35     public void setNome(String nome){
36         this.nome=nome;
37     }
38
39     public ArrayList< Pedido > getPedidos(){
40         return pedidos;
41     }
42
43     public Pedido getPedido(int indice){
44         return pedidos.get(indice);
45     }
46
47     public void removerPedido(Pedido pedido){
48         pedidos.remove(pedido);
49     }
50
51     public boolean estaAtiva(){
52         return ativa;
53     }
54
55     public void desativar(){
56         ativa = false;
57     }
58

```

```

59     public float getTotal(){
60         return total;
61     }
62 }

```

Código 2: Comanda.java

#### 4.1.3 Compra

```

1 package ds2.equipe1.restaurante.modelos;
2
3 import android.content.Context;
4
5 /**
6  * Created by Th on 24/03/2016.
7  */
8 public class Compra extends Model<Compra> {
9     private Item item;
10    private int quantidade;
11    private float preco;
12    private String data;
13
14    public Compra(Context context, Item item, int quantidade, float preco, String data) {
15        super(context);
16        this.item = item;
17        this.quantidade = quantidade;
18        this.preco = preco;
19        this.data = data;
20    }
21
22    public String getNomeDoItem(){
23        return this.item.getNome();
24    }
25
26    public int getQuantidade(){
27        return this.quantidade;
28    }
29
30    public float getPreco() {
31        return this.preco;
32    }
33
34    public String getData() {
35        return this.data;
36    }
37 }

```

Código 3: Compra.java

#### 4.1.4 Endereço

```

1 package ds2.equipe1.restaurante.modelos;
2
3 import android.content.Context;
4
5 /**
6  * Created by Th on 24/03/2016.
7  */
8 public class Endereco extends Model<Endereco> {
9     private String logradouro;
10    private String rua;
11    private int numero;
12    private String bairro;
13    private String cidade;
14    private String estado;
15    private String cep;
16
17    public Endereco(Context context, String logradouro, String rua, int numero, String bairro, String cidade, String
18        estado, String cep) {
19        super(context);
20        this.logradouro = logradouro;
21        this.rua = rua;
22        this.numero = numero;
23        this.bairro = bairro;
24        this.cidade = cidade;
25        this.cep = cep;
26        this.estado = estado;
27    }
28
29    public String getLogradouro() {
30        return logradouro;
31    }

```

```

30 }
31
32     public String getRua() {
33         return rua;
34     }
35
36     public int getNumero() {
37         return numero;
38     }
39
40     public String getBairro() {
41         return bairro;
42     }
43
44     public String getCidade() {
45         return cidade;
46     }
47
48     public String getCep() {
49         return cep;
50     }
51
52     public String getEstado() {
53         return estado;
54     }
55
56     public void setEstado(String estado) {
57         this.estado = estado;
58     }
59 }
```

Código 4: Endereco.java

#### 4.1.5 Fornecedor

```

1 package ds2.equipe1.restaurante.modelos;
2
3 import android.content.Context;
4
5 /**
6  * Created by Th on 24/03/2016.
7  */
8 public class Fornecedor extends Model<Fornecedor> {
9     private String nome;
10    private String telefone;
11    private String cnpj;
12    private String email;
13    private Integer id_endereco;
14
15    public Fornecedor(Context context){
16        super(context);
17    }
18
19    public Fornecedor(Context context, String nome, String telefone, String cnpj, String email){
20        super(context);
21        this.nome = nome;
22        this.telefone = telefone;
23        this.cnpj = cnpj;
24        this.email = email;
25    }
26
27    public String getNome(){
28        return nome;
29    }
30
31    public String getTelefone (){
32        return telefone;
33    }
34
35    public String getCnpj(){
36        return cnpj;
37    }
38
39    public String getEmail(){
40        return email;
41    }
42
43    public void setNome(String nome){
44        this.nome = nome;
45    }
46
47    public void setTelefone (String telefone){
48        this.telefone = telefone;
49    }
50
51    public void setCnpj(String cnpj){
52        this.cnpj = cnpj;
53    }
```

```

54     public void setEmail(String email){
55         this.email = email;
56     }
57
58     public Integer getIdEndereco() {
59         return id_endereco;
60     }
61
62     public void setIdEndereco(Integer id_endereco) {
63         this.id_endereco = id_endereco;
64     }
65 }

```

Código 5: Fornecedor.java

#### 4.1.6 Funcionário

```

1 package ds2.equipe1.restaurante.modelos;
2
3 import android.content.Context;
4 import android.content.Intent;
5
6 /**
7 * Created by Th on 24/03/2016.
8 */
9 public class Funcionario extends Model<Funcionario> {
10     private String nome;
11     private Integer id_endereco;
12     private String telefone;
13     private String cpf;
14     private String nome_de_usuario;
15
16     public Funcionario(Context context){
17         super(context);
18     }
19
20     public Funcionario(Context context, String nome, Integer id_endereco, String telefone, String cpf, String
21     nome_de_usuario) {
22         super(context);
23         this.nome = nome;
24         this.id_endereco = id_endereco;
25         this.telefone = telefone;
26         this.cpf = cpf;
27         this.nome_de_usuario = nome_de_usuario;
28     }
29
30     public String getNome() {
31         return nome;
32     }
33
34     public Integer getIdEndereco() {
35         return id_endereco;
36     }
37
38     public String getTelefone() {
39         return telefone;
40     }
41
42     public String getCpf() {
43         return cpf;
44     }
45
46     public String getNome_de_usuario() {
47         return nome_de_usuario;
48     }
49
50     public void setNome(String nome){
51         this.nome = nome;
52     }
53
54     public void setTelefone (String telefone){
55         this.telefone = telefone;
56     }
57
58     public void setCpf(String cnpj){
59         this.cpf = cnpj;
60     }
61
62     public void setIdEndereco(Integer id_endereco) {
63         this.id_endereco = id_endereco;
64     }
65
66     public void setNome_de_usuario (String nome_de_usuario) { this.nome_de_usuario = nome_de_usuario; }
67 }

```

Código 6: Funcionario.java

#### 4.1.7 Garçom

```
1 package ds2.equipe1.restaurante.modelos;
2
3 import android.content.Context;
4
5 /**
6  * Created by Th on 24/03/2016.
7  */
8 public class Garcom extends Funcionario {
9     public Garcom(Context context, String nome, Integer id_endereco, String telefone, String cpf, String
10        nome_de_usuario) {
11            super(context, nome, id_endereco, telefone, cpf, nome_de_usuario);
12        }
13    }
14 }
```

Código 7: Garcom.java

#### 4.1.8 Gerente

```
1 package ds2.equipe1.restaurante.modelos;
2
3 import android.content.Context;
4
5 /**
6  * Created by Th on 24/03/2016.
7  */
8 public class Gerente extends Funcionario {
9     public Gerente(Context context, String nome, Integer id_endereco, String telefone, String cpf, String
10        nome_de_usuario) {
11            super(context, nome, id_endereco, telefone, cpf, nome_de_usuario);
12        }
13    }
14 }
```

Código 8: Gerente.java

#### 4.1.9 Ingrediente

```
1 package ds2.equipe1.restaurante.modelos;
2
3 import android.content.Context;
4
5 /**
6  * Created by Fernando on 28/03/2016.
7  */
8 public class Ingrediente extends Model<Ingrediente> {
9     private Item item;
10    private int quantidade;
11
12    public Ingrediente(Context context, Item item, int quantidade) {
13        super(context);
14        this.item = item;
15        this.quantidade = quantidade;
16    }
17
18    public Item getItem() {
19        return item;
20    }
21
22    public int getQuantidade() {
23        return quantidade;
24    }
25 }
```

Código 9: Ingrediente.java

#### 4.1.10 Item

```
1 package ds2.equipe1.restaurante.modelos;
2
3 import android.content.Context;
4
5 import ds2.equipe1.restaurante.controles.ControleDeAtendimento;
```

```

6 /**
7  * Created by Th on 24/03/2016.
8 */
9
10 public class Item extends Model<Item> {
11     private String nome;
12     private String unidade;
13     private int quantidade;
14     private int limiteMinimo;
15
16
17     public Item(Context context){
18         super(context);
19     }
20
21     public Item(Context context, String nome, int quantidade, String unidade, int limiteMinimo){
22         super(context);
23         this.nome = nome;
24         this.unidade = unidade;
25         this.quantidade = quantidade;
26         this.limiteMinimo = limiteMinimo;
27     }
28
29     public boolean AlertarItemAbaixoDoLímite(){
30         if (quantidade < limiteMinimo) {
31             return true;
32         } else {
33             return false;
34         }
35     }
36
37     public String getNome(){
38         return nome;
39     }
40
41     public int getQuantidade(){
42         return quantidade;
43     }
44
45     public int getLimiteMinimo(){
46         return limiteMinimo;
47     }
48
49     /* Nome deve ser imutável
50     public void setNome(String nome){
51         this.nome = nome;
52     }
53     */
54
55     public void setQuantidade(int quantidade){
56         //alterar quantidade no banco
57         this.quantidade = quantidade;
58     }
59
60     public void setLimiteMinimo(int limiteMinimo){
61         //alterar limiteMinimo no banco
62         this.limiteMinimo = limiteMinimo;
63     }
64
65     public void setNome(String nome){
66         this.nome = nome;
67     }
68
69     public void setUnidade(String unidade){
70         this.unidade = unidade;
71     }
72 }
```

Código 10: Item.java

#### 4.1.11 Mesa

```

1 package ds2.equipe1.restaurante.modelos;
2
3 import android.content.Context;
4
5 import java.util.ArrayList;
6
7 /**
8  * Created by Th on 24/03/2016.
9 */
10
11 public class Mesa extends Model<Mesa> {
12     private int numero;
13     private ArrayList<Comanda> comandas = new ArrayList< Comanda >();
14
15     public Mesa(Context context, int numero){
16         super(context);
17     }
18 }
```

```

17     //inserir chamada do SQL para carregar comandas ativas
18     this.numero = numero;
19 }
20
21 public void addComanda(Comanda nova){
22     comandas.add(nova);
23 }
24
25 public ArrayList< Comanda > getComandas(){
26     return comandas;
27 }
28
29 public ArrayList < Comanda > getComandasAtivas(){
30     ArrayList < Comanda > ativas = new ArrayList < Comanda >();
31     for (int i = 0; i < comandas.size(); i++) {
32         Comanda atual = comandas.get(i);
33         if(atual.estaaAtiva()){
34             ativas.add(atual);
35         }
36     }
37     return ativas;
38 }
39
40 public Comanda getComanda(int indice){
41     return comandas.get(indice);
42 }
43
44 }

```

Código 11: Mesa.java

#### 4.1.12 Pedido

```

1 package ds2.equipe1.restaurante.modelos;
2
3 import android.content.Context;
4
5 import java.util.ArrayList;
6
7 /**
8  * Created by Th on 24/03/2016.
9 */
10
11 public class Pedido extends Model<Pedido> {
12
13     int quantidadeDeProdutos;
14     boolean entregue;
15     ArrayList < Produto > produtos = new ArrayList < Produto >();
16
17     public Pedido(Context context, ArrayList<Produto> produtos){
18         super(context);
19         this.produtos = produtos;
20         quantidadeDeProdutos = produtos.size();
21         entregue = false;
22     }
23
24     public float getCusto(){
25         float custo = 0;
26         for (int i = 0; i < produtos.size(); i++) {
27             custo += produtos.get(i).getPreco();
28         }
29         return custo;
30     }
31
32     public void setEntregue(){
33         entregue = true;
34     }
35
36     public int getQuantidadeDeProdutos(){
37         return quantidadeDeProdutos;
38     }
39
40     public ArrayList < Produto > getProdutos(){
41         return produtos;
42     }
43
44     public Produto getProduto(int indice){
45         return produtos.get(indice);
46     }
47
48 }

```

Código 12: Pedido.java

#### 4.1.13 Produto

```
1 package ds2.equipe1.restaurante.modelos;
2
3 import android.content.Context;
4
5 import java.util.ArrayList;
6
7 import ds2.equipe1.restaurante.BuscaItem;
8 import ds2.equipe1.restaurante.controles.ControleDeItem;
9 import ds2.equipe1.restaurante.helpers.RequestCallback;
10
11 /**
12 * Created by Th on 24/03/2016.
13 */
14 public class Produto extends Model<Produto> {
15     private int id;
16     private String nome;
17     private float preco;
18     private ArrayList<Ingrediente> ingredientes;
19
20     public Produto(Context context, int id, String nome, float preco, ArrayList<Ingrediente> ingredientes) {
21         super(context);
22         this.id = id;
23         this.nome = nome;
24         this.preco = preco;
25         this.ingredientes = ingredientes;
26     }
27
28     public Produto(Context context, String nome, float preco, ArrayList<Ingrediente> ingredientes) {
29         super(context);
30         this.nome = nome;
31         this.preco = preco;
32         this.ingredientes = ingredientes;
33         //TODO: criar no banco e gerar id
34     }
35
36     /*public static ArrayList<Produto> carregarProdutos() {
37         ArrayList<Produto> listaDeProdutos = new ArrayList<Produto>();
38
39         ArrayList<Produto> lista = new ArrayList<Produto>();
40
41         //conexao com o banco estabelecida
42         for (int i = 0; i < lista.count(); i++) {
43             listaDeProdutos.add(i, new Produto(lista.nome, lista.preco, lista.id, ...));
44         }
45
46         return listaDeProdutos; //(?)
47     }*/
48
49     public boolean validarProduto() {
50         /*for (int i = 0; i < ingredientes.size(); i++) {
51             if (ingredientes.get(i).getQuantidade() > ControleDeItem.consultarItem(ingredientes.get(i).getItem().getNome(), new RequestCallback<Item>() {
52                 @Override
53                 public void execute(ArrayList<Item> itens) {
54                     BuscaItem.this.itens.clear();
55                     BuscaItem.this.itens.addAll(itens);
56                     adapter.notifyDataSetChanged();
57                     super.execute(itens);
58                 }
59                 }).getQuantidade()) {
60                     return false;
61                 }
62             }*/
63         return true;
64     }
65
66     public String getNome(){
67         return nome;
68     }
69
70     public float getPreco(){
71         return preco;
72     }
73
74     public ArrayList <Ingrediente> getItens(){
75         return ingredientes;
76     }
77
78     public void setPreco(float preco){
79         this.preco = preco;
80     }
81 }
```

Código 13: Produto.java

## 4.2 Controles

### 4.2.1 Controle de Atendimento

```
1 package ds2.equipe1.restaurante.controles;
2
3 import android.content.Context;
4
5 import java.util.ArrayList;
6
7 import ds2.equipe1.restaurante.modelos.Comanda;
8 import ds2.equipe1.restaurante.modelos.Mesa;
9 import ds2.equipe1.restaurante.modelos.Pedido;
10 import ds2.equipe1.restaurante.modelos.Produto;
11
12 /**
13 * Created by Th on 24/03/2016.
14 */
15 public class ControleDeAtendimento {
16     private Context context;
17     private ControleDeImpressao controleDeImpressao;
18
19     private ArrayList<Mesa> mesas = new ArrayList<Mesa>();
20
21     public ControleDeAtendimento(Context context){
22         this.context = context;
23         this.controleDeImpressao = new ControleDeImpressao(context);
24
25         //inserir comando sql pra saber numero de mesas
26         int numeroDeMesas = 20; //puxar numero do sql
27         for(int i = 1; i <= numeroDeMesas; i++) {
28             mesas.add(new Mesa(context, i));
29         }
30     }
31
32     public void criarComanda(Mesa mesa, Pedido pedido, String nome){
33         mesa.addComanda(new Comanda(context, nome, pedido));
34     }
35
36     public void encerrarComanda(Comanda comanda){
37         comanda.desativar();
38     }
39
40     public void imprimirPedido(Pedido pedido){
41         ArrayList<Produto> listaDeProdutos = pedido.getProdutos();
42         for(int i = 0; i < listaDeProdutos.size(); i++){
43             controleDeImpressao.imprimir(listaDeProdutos.get(i).getNome());
44         }
45     }
46
47     public void imprimirConta(Comanda comanda){
48         String saida = "";
49         ArrayList < Pedido > listaDePedidos = comanda.getPedidos();
50         for(int i = 0; i < listaDePedidos.size(); i++) {
51             ArrayList < Produto > listaDeProdutos = listaDePedidos.get(i).getProdutos();
52             for(int j = 0; j < listaDeProdutos.size(); j++) {
53                 saida += listaDeProdutos.get(j).getNome() + " " + listaDeProdutos.get(j).getPreco() + "\n";
54             }
55             float total = comanda.getTotal();
56             saida += "Total de Pedidos" + total + "\n 10% = " + 0.1*total + "\n Total " + 1.1*total;
57         }
58         controleDeImpressao.imprimir(saida);
59     }
60
61     public ArrayList < Produto > consultarPedido(Pedido pedido){
62         return pedido.getProdutos();
63     }
64
65     public void incluirPedido(Comanda comanda, Pedido pedido){
66         comanda.addPedido(pedido);
67     }
68
69     public void entregarPedido(Pedido pedido){
70         pedido.setEntregue();
71     }
72
73     public void cancelarPedido(Comanda comanda, Pedido pedido) {
74         comanda.removerPedido(pedido);
75     }
76
77     public ArrayList < Produto > consultarCardapio() {
78         ArrayList < Produto > cardapio = new ArrayList < Produto >();
79         //Aqui so com consulta ao DB ne? pq precisa pegar a lista de produtos.
80         //fica pra outra hora beijo no coracao!
81         return cardapio;
82     }
83
84     public ArrayList < Comanda > relatorioComandasAtivas(){
85         ArrayList < Comanda > ativas = new ArrayList < Comanda >();
86         //preciso de consulta ao banco neste ponto
87     }
```

```

88     return ativas;
89 }
90
91
92     public ArrayList < Comanda > relatorioComandasEncerradas(){
93         ArrayList < Comanda > encerradas = new ArrayList < Comanda >();
94         //preciso de consulta ao banco neste ponto
95         return encerradas;
96     }
97
98     public void relatorioArrecadacaoBruta(){
99         //preciso de consulta ao banco neste ponto
100    }
101
102}

```

Código 14: ControleDeAtendimento.java

#### 4.2.2 Controle de Compra

#### 4.2.3 Controle de Fornecedor

#### 4.2.4 Controle de Funcionário

#### 4.2.5 Controle de Impressão

```

1 package ds2.equipe1.restaurante.controles;
2
3 import android.content.Context;
4 import android.content.Intent;
5
6 import ds2.equipe1.restaurante.TelaRelatorios;
7
8 /**
9  * Created by Fernando on 28/03/2016.
10 */
11 public class ControleDeImpressao {
12     private Context context;
13
14     public ControleDeImpressao(Context context) {
15         this.context = context;
16     }
17
18     public void imprimir(String dados){
19         Intent i = new Intent(context, TelaRelatorios.class);
20         i.putExtra("dados", dados);
21         context.startActivity(i);
22     }
23 }

```

Código 15: ControleDeImpressao.java

#### 4.2.6 Controle de Item

#### 4.2.7 Controle de Produto

```

1 package ds2.equipe1.restaurante.controles;
2
3 import android.content.Context;
4
5 import java.util.ArrayList;
6
7 import ds2.equipe1.restaurante.modelos.Ingrediente;
8 import ds2.equipe1.restaurante.modelos.Item;
9 import ds2.equipe1.restaurante.modelos.Produto;
10 import java.sql.*;
11
12 /**
13  * Created by Th on 24/03/2016.
14  */
15 public class ControleDeProduto {
16
17     private ArrayList < Produto > produtos;
18     private Context context;
19
20     public ControleDeProduto(Context context){
21         this.context = context;
22     }

```

```

23
24
25     public void carregarProdutosDoBanco(){
26         produtos = new ArrayList <Produto>();
27         int tamanhoDoCardapio = 5; //Ler o tamanho do banco
28
29         for (int i = 0; i < tamanhoDoCardapio; i++) {
30             String nome = "teste"; //Ler o nome do banco
31             //produtos.add(new Produto(nome));
32         }
33     }
34
35     public void cadastrarProduto(String nome, float preco, ArrayList <Ingrediente> ingredientes){
36         produtos.add(new Produto(context, nome, preco, ingredientes));
37     }
38
39     public void excluirProduto(String nome){
40         for (int i = 0; i < produtos.size(); i++) {
41             if (produtos.get(i).getNome() == nome) {
42                 //Remover do banco
43                 produtos.remove(i); //ver se e este mesmo o metodo de remocao
44             }
45         }
46     }
47
48     public void alterarPrecoDeProduto(String nome, float novoPreco){
49         for (int i = 0; i < produtos.size(); i++) {
50             if (produtos.get(i).getNome() == nome) {
51                 //Alterar preco no banco
52                 produtos.get(i).setPreco(novoPreco);
53             }
54         }
55     }
56
57     public Produto consultarProduto(String nome){
58         for (int i = 0; i < produtos.size(); i++) {
59             if (produtos.get(i).getNome() == nome) {
60                 return produtos.get(i);
61             }
62         }
63         return null;
64     }
65
66     public void relatorioProdutos(){
67         //Gerar pdf?
68     }
69
70     public void relatorioItensPorProduto(){
71         //Gerar pdf?
72     }
73 }

```

Código 16: ControleDeProduto.java

## 4.3 Fronteiras

### 4.3.1 Main Activity

```

1 package ds2.equipe1.restaurante;
2
3 import android.support.v7.app.AppCompatActivity;
4 import android.os.Bundle;
5
6 public class MainActivity extends AppCompatActivity {
7
8     @Override
9     protected void onCreate(Bundle savedInstanceState) {
10         super.onCreate(savedInstanceState);
11         setContentView(R.layout.activity_main);
12
13         //TODO: Tela de log in e botao de log off
14     }
15 }

```

Código 17: MainActivity.java

### 4.3.2 Menu Principal

```

1 package ds2.equipe1.restaurante;
2
3 import android.content.Intent;
4 import android.support.v7.app.AppCompatActivity;
5 import android.os.Bundle;
6 import android.view.Menu;
7 import android.view.MenuItem;
8 import android.view.View;
9
10 import ds2.equipe1.restaurante.controles.ControleDeFornecedor;
11 import ds2.equipe1.restaurante.controles.ControleDeImpressao;
12 import ds2.equipe1.restaurante.helpers.Utils;
13 import ds2.equipe1.restaurante.modelos.Fornecedor;
14 import ds2.equipe1.restaurante.modelos.Funcionario;
15
16 public class MenuPrincipal extends AppCompatActivity {
17
18     @Override
19     protected void onCreate(Bundle savedInstanceState) {
20         super.onCreate(savedInstanceState);
21         setContentView(R.layout.activity_menu_principal);
22
23         findViewById(R.id.menu_fornecedores).setOnClickListener(new View.OnClickListener() {
24             @Override
25             public void onClick(View v) {
26                 open(MenuFornecedores.class);
27             }
28         });
29
30         findViewById(R.id.menu_garcom).setOnClickListener(new View.OnClickListener() {
31             @Override
32             public void onClick(View v) {
33                 open(TelaGarcom.class);
34             }
35         });
36
37         findViewById(R.id.menu_itens).setOnClickListener(new View.OnClickListener() {
38             @Override
39             public void onClick(View v) {
40                 open(MenuItens.class);
41             }
42         });
43
44         findViewById(R.id.menu_produto).setOnClickListener(new View.OnClickListener() {
45             @Override
46             public void onClick(View v) {
47                 open(MenuProdutos.class);
48             }
49         });
50
51         findViewById(R.id.menu_relatorios).setOnClickListener(new View.OnClickListener() {
52             @Override
53             public void onClick(View v) {
54                 open(MenuRelatorio.class);
55             }
56         });
57
58         findViewById(R.id.menu_funcionarios).setOnClickListener(new View.OnClickListener() {
59             @Override
60             public void onClick(View v) {
61                 open(MenuFuncionarios.class);
62             }
63         });
64     }
65
66     @Override
67     public boolean onCreateOptionsMenu(Menu menu) {
68         // Inflate the menu; this adds items to the action bar if it is present.
69         getMenuInflater().inflate(R.menu.menu_principal, menu);
70         return true;
71     }
72
73     @Override
74     public boolean onOptionsItemSelected(MenuItem item) {
75         // Handle action bar item clicks here. The action bar will
76         // automatically handle clicks on the Home/Up button, so long
77         // as you specify a parent activity in AndroidManifest.xml.
78         int id = item.getItemId();
79
80         if (id == R.id.configurar_host) {
81             final Utils utils = new Utils(this);
82             utilsInputDialog("Configurar host", utils.getData("host", ""), "http://192.168.1.100/restaurante/", new
83             Utils.DialogCallback() {
84                 @Override
85                 public void execute(String text) {
86                     utils.setData("host", text);
87                 }
88             });
89             return true;
90         }
91
92         return super.onOptionsItemSelected(item);
93     }

```

```

93     public void open(Class activity){
94         this.startActivity(new Intent(this, activity));
95     }
96 }
97 
```

Código 18: MenuPrincipal.java

#### 4.3.3 Menu Fornecedores

```

1 package ds2.equipe1.restaurante;
2
3 import android.content.Intent;
4 import android.support.v7.app.AppCompatActivity;
5 import android.os.Bundle;
6 import android.view.View;
7
8 public class MenuFornecedores extends AppCompatActivity {
9
10    @Override
11    protected void onCreate(Bundle savedInstanceState) {
12        super.onCreate(savedInstanceState);
13        setContentView(R.layout.activity_menu_fornecedores);
14
15        findViewById(R.id.menu_cadastrar_fornecedor).setOnClickListener(new View.OnClickListener() {
16            @Override
17            public void onClick(View v) {
18                open(CadastroFornecedor.class);
19            }
20        });
21
22        findViewById(R.id.menu_buscar_fornecedor).setOnClickListener(new View.OnClickListener() {
23            @Override
24            public void onClick(View v) {
25                open(BuscaFornecedor.class);
26            }
27        });
28    }
29
30    public void open(Class activity){
31        startActivity(new Intent(this, activity));
32    }
33 }
34 
```

Código 19: MenuFornecedores.java

#### 4.3.4 Menu Funcionários

```

1 package ds2.equipe1.restaurante;
2
3 import android.content.Intent;
4 import android.support.v7.app.AppCompatActivity;
5 import android.os.Bundle;
6 import android.view.View;
7
8 public class MenuFuncionarios extends AppCompatActivity {
9
10    @Override
11    protected void onCreate(Bundle savedInstanceState) {
12        super.onCreate(savedInstanceState);
13        setContentView(R.layout.activity_menu_funcionarios);
14
15        findViewById(R.id.menu_cadastrar_funcionario).setOnClickListener(new View.OnClickListener() {
16            @Override
17            public void onClick(View v) {
18                open(CadastroFuncionario.class);
19            }
20        });
21
22        findViewById(R.id.menu_buscar_funcionario).setOnClickListener(new View.OnClickListener() {
23            @Override
24            public void onClick(View v) {
25                open(BuscaFuncionario.class);
26            }
27        });
28
29        findViewById(R.id.menu_alterar_funcionario).setOnClickListener(new View.OnClickListener() {
30            @Override
31            public void onClick(View v) {
32                open(CadastroFuncionario.class);
33            }
34 
```

```

34     });
35 }
36
37 public void open(Class activity){
38     startActivity(new Intent(this, activity));
39 }
40

```

Código 20: MenuFuncionarios.java

#### 4.3.5 Menu Garçom

```

1 package ds2.equipe1.restaurante;
2
3 import android.content.Intent;
4 import android.support.v7.app.AppCompatActivity;
5 import android.os.Bundle;
6 import android.view.View;
7 import android.widget.Toast;
8
9 public class MenuGarcom extends AppCompatActivity {
10
11     @Override
12     protected void onCreate(Bundle savedInstanceState) {
13         super.onCreate(savedInstanceState);
14         setContentView(R.layout.activity_menu_garcom);
15
16         findViewById(R.id.menu_criar_comanda).setOnClickListener(new View.OnClickListener() {
17             @Override
18             public void onClick(View v) {
19                 open(CriacaoComanda.class);
20             }
21         });
22
23         findViewById(R.id.menu_consulta_cardapio).setOnClickListener(new View.OnClickListener() {
24             @Override
25             public void onClick(View v) {
26                 open(BuscaProduto.class);
27             }
28         });
29
30         findViewById(R.id.menu_comandas_ativas).setOnClickListener(new View.OnClickListener() {
31             @Override
32             public void onClick(View v) {
33                 Toast.makeText(MenuGarcom.this, "Falta criar tela e adicionar Requisito Funcional, Diagramas etc",
34                     Toast.LENGTH_LONG).show();
35             }
36         });
37
38         public void open(Class activity){
39             startActivity(new Intent(this, activity));
40         }
41

```

Código 21: MenuGarcom.java

#### 4.3.6 Menu Itens

```

1 package ds2.equipe1.restaurante;
2
3 import android.content.Intent;
4 import android.support.v7.app.AppCompatActivity;
5 import android.os.Bundle;
6 import android.view.View;
7
8 public class MenuItem extends AppCompatActivity {
9
10     @Override
11     protected void onCreate(Bundle savedInstanceState) {
12         super.onCreate(savedInstanceState);
13         setContentView(R.layout.activity_menu_itens);
14
15         findViewById(R.id.menu_cadastrar_item).setOnClickListener(new View.OnClickListener() {
16             @Override
17             public void onClick(View v) {
18                 open(CadastroItem.class);
19             }
20         });
21
22         findViewById(R.id.menu_alterar_item).setOnClickListener(new View.OnClickListener() {
23             @Override

```

```

24     public void onClick(View v) {
25         open(CadastroItem.class);
26     });
27 );
28
29 findViewById(R.id.menu_buscar_item).setOnClickListener(new View.OnClickListener() {
30     @Override
31     public void onClick(View v) {
32         open(BuscaItem.class);
33     }
34 );
35
36
37     public void open(Class activity){
38         startActivity(new Intent(this, activity));
39     }
40 }

```

Código 22: MenuItens.java

#### 4.3.7 Menu Produtos

```

1 package ds2.equipe1.restaurante;
2
3 import android.content.Intent;
4 import android.support.v7.app.AppCompatActivity;
5 import android.os.Bundle;
6 import android.view.View;
7
8 public class MenuProdutos extends AppCompatActivity {
9
10    @Override
11    protected void onCreate(Bundle savedInstanceState) {
12        super.onCreate(savedInstanceState);
13        setContentView(R.layout.activity_menu_produtos);
14
15        findViewById(R.id.menu_cadastrar_produto).setOnClickListener(new View.OnClickListener() {
16            @Override
17            public void onClick(View v) {
18                open(CadastroProduto.class);
19            }
20        );
21
22        findViewById(R.id.menu_buscar_produto).setOnClickListener(new View.OnClickListener() {
23            @Override
24            public void onClick(View v) {
25                open(BuscaProduto.class);
26            }
27        );
28
29        findViewById(R.id.menu_alterar_produto).setOnClickListener(new View.OnClickListener() {
30            @Override
31            public void onClick(View v) {
32                open(CadastroProduto.class);
33            }
34        );
35
36
37        public void open(Class activity){
38            startActivity(new Intent(this, activity));
39        }
40    }

```

Código 23: MenuProdutos.java

#### 4.3.8 Menu Relatório

```

1 package ds2.equipe1.restaurante;
2
3 import android.support.v7.app.AppCompatActivity;
4 import android.os.Bundle;
5
6 public class MenuRelatorio extends AppCompatActivity {
7
8    @Override
9    protected void onCreate(Bundle savedInstanceState) {
10        super.onCreate(savedInstanceState);
11        setContentView(R.layout.activity_menu_relatorio);
12    }
13

```

Código 24: MenuRelatorio.java

#### 4.3.9 Busca Fornecedor

#### 4.3.10 Busca Funcionário

#### 4.3.11 Busca Item

#### 4.3.12 Busca Produto

```
1 package ds2.equipe1.restaurante;
2
3 import android.content.Intent;
4 import android.support.v7.app.AppCompatActivity;
5 import android.os.Bundle;
6 import android.view.View;
7
8 public class BuscaProduto extends AppCompatActivity {
9
10    @Override
11    protected void onCreate(Bundle savedInstanceState) {
12        super.onCreate(savedInstanceState);
13        setContentView(R.layout.activity_busca_produto);
14    }
15
16    public void onItemClick(View v){
17        startActivity(new Intent(this, CadastroProduto.class));
18    }
19}
```

Código 25: BuscaProduto.java

#### 4.3.13 Cadastro Endereço

```
1 package ds2.equipe1.restaurante;
2
3 import android.content.Intent;
4 import android.support.v7.app.AppCompatActivity;
5 import android.os.Bundle;
6 import android.view.View;
7 import android.widget.Button;
8 import android.widget.EditText;
9
10 import org.json.JSONObject;
11
12 import ds2.equipe1.restaurante.helpers.RequestCallback;
13 import ds2.equipe1.restaurante.modelos.Endereco;
14 import ds2.equipe1.restaurante.modelos.Model;
15
16 public class CadastroEndereco extends AppCompatActivity {
17
18     private EditText edtCEP, edtLogradouro, edtRua, edtBairro, edtCidade, edtEstado, edtNumero;
19     private Button btnCadastrar;
20
21     @Override
22     protected void onCreate(Bundle savedInstanceState) {
23         super.onCreate(savedInstanceState);
24         setContentView(R.layout.activity_cadastro_endereco);
25
26         init();
27     }
28
29     private void init(){
30         btnCadastrar = (Button) findViewById(R.id.btnCadastrar);
31         edtRua = (EditText) findViewById(R.id.edtRua);
32         edtCEP = (EditText) findViewById(R.id.edtCEP);
33         edtLogradouro = (EditText) findViewById(R.id.edtLogradouro);
34         edtBairro = (EditText) findViewById(R.id.edtBairro);
35         edtCidade = (EditText) findViewById(R.id.edtCidade);
36         edtEstado = (EditText) findViewById(R.id.edtEstado);
37         edtNumero = (EditText) findViewById(R.id.edtNumero);
38
39         btnCadastrar.setOnClickListener(new View.OnClickListener() {
40             @Override
41             public void onClick(View v) {
42
43                 final Endereco endereco = new Endereco(CadastroEndereco.this, edtLogradouro.getText().toString(),
44                 edtRua.getText().toString(), Integer.parseInt(edtNumero.getText().toString()), edtBairro.getText().toString(),
45                 edtCidade.getText().toString(), edtEstado.getText().toString(), edtCEP.getText().toString());
46
47                 endereco.save(new RequestCallback<Model>() {
48                     @Override
49                     public void execute(Model model) throws Exception {
50                         super.execute(model);
51                     }
52                 });
53             }
54         });
55     }
56 }
```

```

50         Intent i = getIntent();
51         i.putExtra("rua", endereco.getRua());
52         i.putExtra("id_endereco", model.getId());
53         setResult(RESULT_OK, i);
54         finish();
55     }
56 }
57 }
58 }
59 }
60 }

```

Código 26: CadastroEndereco.java

#### 4.3.14 Cadastro Fornecedor

#### 4.3.15 Cadastro Funcionário

#### 4.3.16 Cadastro Item

#### 4.3.17 Cadastro Produto

#### 4.3.18 Criação Comanda

```

1 package ds2.equipe1.restaurante;
2
3 import android.content.Context;
4 import android.content.Intent;
5 import android.support.v7.app.AppCompatActivity;
6 import android.os.Bundle;
7 import android.view.Menu;
8 import android.view.MenuItem;
9 import android.view.View;
10
11 import ds2.equipe1.restaurante.controles.ControleDeAtendimento;
12 import ds2.equipe1.restaurante.modelos.Comanda;
13
14 public class CriacaoComanda extends AppCompatActivity {
15
16     private ControleDeAtendimento controleDeAtendimento;
17     Comanda comanda;
18     private Context context;
19
20
21     @Override
22     protected void onCreate(Bundle savedInstanceState) {
23         super.onCreate(savedInstanceState);
24         setContentView(R.layout.activity_criacao_comanda);
25         controleDeAtendimento = new ControleDeAtendimento(this);
26
27         findViewById(R.id.btnAdicionarPedido).setOnClickListener(new View.OnClickListener() {
28             @Override
29             public void onClick(View v) {
30                 startActivity(new Intent(CriacaoComanda.this, BuscaProduto.class));
31             }
32         });
33
34         findViewById(R.id.btnEncerrarComanda).setOnClickListener(new View.OnClickListener() {
35             @Override
36             public void onClick(View v) {
37                 //encerrar comanda
38                 //imprimir comanda
39             }
40         });
41     }
42
43     @Override
44     public boolean onCreateOptionsMenu(Menu menu) {
45         // Inflate the menu; this adds items to the action bar if it is present.
46         getMenuInflater().inflate(R.menu.comandas, menu);
47         return true;
48     }
49
50     @Override
51     public boolean onOptionsItemSelected(MenuItem item) {
52         // Handle action bar item clicks here. The action bar will
53         // automatically handle clicks on the Home/Up button, so long
54         // as you specify a parent activity in AndroidManifest.xml.
55         int id = item.getItemId();
56
57         if (id == android.R.id.home) {
58             onBackPressed();
59             return true;

```

```

60     }
61
62     return super.onOptionsItemSelected(item);
63 }
64
65     public void onItemClick(View v){
66         startActivity(new Intent(this, ExibirPedido.class));
67     }
68 }
```

Código 27: CriacaoComanda.java

#### 4.3.19 Exibir Pedido

```

1 package ds2.equipe1.restaurante;
2
3 import android.support.v7.app.AppCompatActivity;
4 import android.os.Bundle;
5
6 public class ExibirPedido extends AppCompatActivity {
7
8     @Override
9     protected void onCreate(Bundle savedInstanceState) {
10        super.onCreate(savedInstanceState);
11        setContentView(R.layout.activity_pedido);
12    }
13 }
```

Código 28: ExibirPedido.java

#### 4.3.20 Seleccionar Ingredientes

```

1 package ds2.equipe1.restaurante;
2
3 import android.support.v7.app.AppCompatActivity;
4
5 /**
6  * Created by Th on 18/04/2016.
7  */
8 public class SeleccionarIngredientes extends AppCompatActivity {
9 }
```

Código 29: SeleccionarIngredientes.java

#### 4.3.21 Tela Garçom

```

1 package ds2.equipe1.restaurante;
2
3 import android.content.Intent;
4 import android.support.v7.app.AppCompatActivity;
5 import android.os.Bundle;
6 import android.view.View;
7
8 public class TelaGarcom extends AppCompatActivity {
9
10    @Override
11    protected void onCreate(Bundle savedInstanceState) {
12        super.onCreate(savedInstanceState);
13        setContentView(R.layout.activity_tela_garcom);
14
15        findViewById(R.id.menu_criar_comanda).setOnClickListener(new View.OnClickListener() {
16            @Override
17            public void onClick(View v) {
18                open(CriacaoComanda.class);
19            }
20        });
21
22        findViewById(R.id.menu_consulta_cardapio).setOnClickListener(new View.OnClickListener() {
23            @Override
24            public void onClick(View v) {
25                open(BuscaProduto.class);
26            }
27        });
28    }
29
30    public void open(Class activity){
```

```

31     } startActivity(new Intent(this, activity));
32 }
33 }
```

Código 30: TelaGarcom.java

#### 4.3.22 Tela Relatórios

### 4.4 Adapters

#### 4.4.1 Fornecedor Adapter

```

1 package ds2.equipe1.restaurante.listas;
2
3 import android.content.Context;
4 import android.view.LayoutInflater;
5 import android.view.View;
6 import android.view.ViewGroup;
7 import android.widget.BaseAdapter;
8 import android.widget.EditText;
9 import android.widget.TextView;
10
11 import java.util.ArrayList;
12
13 import ds2.equipe1.restaurante.R;
14 import ds2.equipe1.restaurante.modelos.Fornecedor;
15
16 /**
17 * Created by Fernando on 08/04/2016.
18 */
19 public class FornecedorAdapter extends BaseAdapter {
20     private Context context;
21     private ArrayList<Fornecedor> fornecedores;
22
23     public FornecedorAdapter(Context context, ArrayList<Fornecedor> fornecedores) {
24         this.context = context;
25         this.fornecedores = fornecedores;
26     }
27
28     @Override
29     public int getCount() {
30         return fornecedores.size();
31     }
32
33     @Override
34     public Fornecedor getItem(int position) {
35         return fornecedores.get(position);
36     }
37
38     @Override
39     public long getItemId(int position) {
40         return fornecedores.get(position).getId();
41     }
42
43     @Override
44     public View getView(int position, View root, ViewGroup parent) {
45         if (root == null) {
46             LayoutInflater vi = (LayoutInflater) context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
47             root = vi.inflate(R.layout.item_fornecedor, parent, false);
48         }
49
50         TextView edtNome, edtCNPJ, edtTelefone;
51         edtNome = (TextView) root.findViewById(R.id.tvNome);
52         edtCNPJ = (TextView) root.findViewById(R.id.tvCNPJ);
53         edtTelefone = (TextView) root.findViewById(R.id.tvTelefone);
54
55         Fornecedor fornecedor = fornecedores.get(position);
56
57         edtNome.setText(fornecedor.getNome());
58         edtCNPJ.setText(fornecedor.getNome());
59         edtTelefone.setText(fornecedor.getNome());
60
61         return root;
62     }
63 }
```

Código 31: FornecedorAdapter.java

#### 4.4.2 Funcionario Adapter

```

1 package ds2.equipe1.restaurante.listas;
2
3 import android.content.Context;
4 import android.view.LayoutInflater;
5 import android.view.View;
6 import android.view.ViewGroup;
7 import android.widget.BaseAdapter;
8 import android.widget.TextView;
9
10 import java.util.ArrayList;
11
12 import ds2.equipe1.restaurante.R;
13 import ds2.equipe1.restaurante.modelos.Funcionario;
14
15 /**
16 * Created by elton on 07/05/2016.
17 */
18
19 public class FuncionarioAdapter extends BaseAdapter {
20     private Context context;
21     private ArrayList<Funcionario> funcionarios;
22
23     public FuncionarioAdapter(Context context, ArrayList<Funcionario> funcionarios) {
24         this.context = context;
25         this.funcionarios = funcionarios;
26     }
27
28     @Override
29     public int getCount() {
30         return funcionarios.size();
31     }
32
33     @Override
34     public Funcionario getItem(int position) {
35         return funcionarios.get(position);
36     }
37
38     @Override
39     public long getItemId(int position) {
40         return funcionarios.get(position).getId();
41     }
42
43     @Override
44     public View getView(int position, View root, ViewGroup parent) {
45         if (root == null) {
46             LayoutInflater vi = (LayoutInflater) context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
47             root = vi.inflate(R.layout.item_funcionario, parent, false);
48         }
49
50         TextView edtNome, edtCPF, edtTelefone;
51         edtNome = (TextView) root.findViewById(R.id.tvNome);
52         edtCPF = (TextView) root.findViewById(R.id.tvCPF);
53         edtTelefone = (TextView) root.findViewById(R.id.tvTelefone);
54
55         Funcionario funcionario = funcionarios.get(position);
56
57         edtNome.setText(funcionario.getNome());
58         edtCPF.setText(funcionario.getCpf());
59         edtTelefone.setText(funcionario.getTelefone());
60
61         return root;
62     }
63 }

```

Código 32: FuncionarioAdapter.java

#### 4.4.3 Item Adapter

```

1 package ds2.equipe1.restaurante.listas;
2
3 import android.content.Context;
4 import android.view.LayoutInflater;
5 import android.view.View;
6 import android.view.ViewGroup;
7 import android.widget.BaseAdapter;
8 import android.widget.EditText;
9 import android.widget.TextView;
10
11 import java.util.ArrayList;
12
13 import ds2.equipe1.restaurante.R;
14 import ds2.equipe1.restaurante.modelos.Item;
15
16 /**
17 * Created by Fernando on 08/04/2016.
18 */
19 public class ItemAdapter extends BaseAdapter {

```

```

20     private Context context;
21     private ArrayList<Item> itens;
22
23     public ItemAdapter(Context context, ArrayList<Item> fornecedores) {
24         this.context = context;
25         this.itens = fornecedores;
26     }
27
28     @Override
29     public int getCount() {
30         return itens.size();
31     }
32
33     @Override
34     public Item getItem(int position) {
35         return itens.get(position);
36     }
37
38     @Override
39     public long getItemId(int position) {
40         return itens.get(position).getId();
41     }
42
43     @Override
44     public View getView(int position, View root, ViewGroup parent) {
45         if (root == null) {
46             LayoutInflator vi = (LayoutInflator) context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
47             root = vi.inflate(R.layout.item_itens, parent, false);
48         }
49
50         TextView edtNome, edtEstoque;
51         edtNome = (TextView) root.findViewById(R.id.tvNome);
52         edtEstoque = (TextView) root.findViewById(R.id.tvEstoque);
53
54         Item item = itens.get(position);
55         edtNome.setText(item.getNome());
56         edtEstoque.setText(item.getQuantidade());
57
58         return root;
59     }
60 }

```

Código 33: ItemAdapter.java

#### 4.4.4 Produto Adapter

```

1 package ds2.equipe1.restaurante.listas;
2
3 import android.content.Context;
4 import android.view.LayoutInflater;
5 import android.view.View;
6 import android.view.ViewGroup;
7 import android.widget.BaseAdapter;
8 import android.widget.TextView;
9
10 import java.util.ArrayList;
11
12 import ds2.equipe1.restaurante.R;
13 import ds2.equipe1.restaurante.modelos.Produto;
14
15 /**
16  * Created by Fernando on 08/04/2016.
17  */
18 public class ProdutoAdapter extends BaseAdapter {
19     private Context context;
20     private ArrayList<Produto> produtos;
21
22     public ProdutoAdapter(Context context, ArrayList<Produto> produtos) {
23         this.context = context;
24         this.produtos = produtos;
25     }
26
27     @Override
28     public int getCount() {
29         return produtos.size();
30     }
31
32     @Override
33     public Produto getItem(int position) {
34         return produtos.get(position);
35     }
36
37     @Override
38     public long getItemId(int position) {
39         return produtos.get(position).getId();
40     }
41
42     @Override

```

```

43     public View getView(int position, View root, ViewGroup parent) {
44         if (root == null) {
45             LayoutInflator vi = (LayoutInflator) context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
46             root = vi.inflate(R.layout.item_produto, parent, false);
47         }
48
49         TextView edtNome, edtPreco;
50         edtNome = (TextView) root.findViewById(R.id.tvNome);
51         edtPreco = (TextView) root.findViewById(R.id.tvPreco);
52
53         Produto produto = produtos.get(position);
54
55         edtNome.setText(produto.getNome());
56         edtPreco.setText(Float.toString(produto.getPreco()));
57
58         return root;
59     }
60 }
```

Código 34: ProdutoAdapter.java

## 4.5 Helpers

### 4.5.1 Server Request

### 4.5.2 Request Callback

```

1 package ds2.equipe1.restaurante.helpers;
2
3 import android.content.Context;
4
5 import java.util.ArrayList;
6
7 /**
8 * Created by Fernando on 13/04/2016.
9 */
10 public abstract class RequestCallback<T> {
11     private Context context;
12
13     //Cria request callback SEM tela de loading
14     public RequestCallback(){
15         onStart();
16     }
17
18     //Cria request callback com uma tela de loading
19     public RequestCallback(Context context){
20         this.context = context;
21         onStart();
22     }
23
24     public void onStart(){
25         if (context != null){
26             Utils.launchProgress(context);
27         }
28     }
29
30     public void execute(){
31         onFinish();
32     }
33     public void execute(ArrayList<T> lista) throws Exception {
34         onFinish();
35     }
36     public void execute(T object) throws Exception {
37         onFinish();
38     }
39
40     public void onFinish(){
41         if (context != null) {
42             Utils.dismissProgress();
43         }
44     }
}
```

Código 35: RequestCallback.java

### 4.5.3 Utilitários

```

1 package ds2.equipe1.restaurante.helpers;
2
```

```

3 import android.app.ProgressDialog;
4 import android.content.Context;
5 import android.content.DialogInterface;
6 import android.content.SharedPreferences;
7 import android.support.v7.app.AlertDialog;
8 import android.view.LayoutInflater;
9 import android.widget.EditText;
10 import android.widget.LinearLayout;
11 import android.widget.Spinner;
12 import android.widget.Toast;
13
14 import ds2.equipe1.restaurante.R;
15
16 /**
17 * Created by Fernando on 28/03/2016.
18 */
19 public class Utils {
20     public static final String TAG = "Restaurante";
21     private Context context;
22     private static ProgressDialog progress;
23
24     public Utils(Context context) {
25         this.context = context;
26     }
27
28     public void deleteData(String key){
29         SharedPreferences settings = context.getSharedPreferences("EsporteNet", 0);
30         SharedPreferences.Editor e = settings.edit();
31         e.remove(key);
32         e.commit();
33     }
34
35     public float getData(String key, float defValue) {
36         SharedPreferences settings = context.getSharedPreferences("EsporteNet", 0);
37         return settings.getFloat(key, defValue);
38     }
39
40     public String getData(String key, String defValue) {
41         SharedPreferences settings = context.getSharedPreferences("EsporteNet", 0);
42         return settings.getString(key, defValue);
43     }
44
45     public int getData(String key, int defValue) {
46         SharedPreferences settings = context.getSharedPreferences("EsporteNet", 0);
47         return settings.getInt(key, defValue);
48     }
49
50     public long getData(String key, long defValue) {
51         SharedPreferences settings = context.getSharedPreferences("EsporteNet", 0);
52         return settings.getLong(key, defValue);
53     }
54
55     public boolean getData(String key, boolean defValue) {
56         SharedPreferences settings = context.getSharedPreferences("EsporteNet", 0);
57         return settings.getBoolean(key, defValue);
58     }
59
60     public void setData(String key, float value) {
61         SharedPreferences.Editor settings = context.getSharedPreferences("EsporteNet", 0).edit();
62         settings.putFloat(key, value);
63         settings.commit();
64     }
65
66     public void setData(String key, String value) {
67         SharedPreferences.Editor settings = context.getSharedPreferences("EsporteNet", 0).edit();
68         settings.putString(key, value);
69         settings.commit();
70     }
71
72     public void setData(String key, int value) {
73         SharedPreferences.Editor settings = context.getSharedPreferences("EsporteNet", 0).edit();
74         settings.putInt(key, value);
75         settings.commit();
76     }
77
78     public void setData(String key, long value) {
79         SharedPreferences.Editor settings = context.getSharedPreferences("EsporteNet", 0).edit();
80         settings.putLong(key, value);
81         settings.commit();
82     }
83
84     public void setData(String key, boolean value) {
85         SharedPreferences.Editor settings = context.getSharedPreferences("EsporteNet", 0).edit();
86         settings.putBoolean(key, value);
87         settings.commit();
88     }
89
90     public void clearData(){
91         SharedPreferences settings = context.getSharedPreferences("EsporteNet", 0);
92         SharedPreferences.Editor editor = settings.edit();
93         editor.clear();
94         editor.commit();
95     }
96

```

```

97     public static void launchProgress(Context context){
98         launchProgress(context, null, "Carregando...");
99     }
100
101    public static void launchProgress(Context context, String title, String text) {
102        launchProgress(context, title, text, null);
103    }
104
105    public static void launchProgress(Context context, String title, String text, DialogInterface.OnDismissListener
106        func) {
107        progress = ProgressDialog.show(context, title, text, true);
108        progress.setCancelable(true);
109        if (func != null) progress.setOnDismissListener(func);
110    }
111
112    public static void dismissProgress() {
113        if (progress != null) {
114            progress.dismiss();
115        }
116    }
117
118    public void toast(String text){
119        Toast.makeText(context, text, Toast.LENGTH_LONG).show();
120    }
121
122    public void inputDialog(String title, String defValue, String hint, final DialogCallback callback){
123        AlertDialog.Builder builder = new AlertDialog.Builder(context);
124        builder.setTitle(title);
125
126        // Set up the input
127        LayoutInflator li = (LayoutInflator) context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
128        LinearLayout linearLayout = (LinearLayout) li.inflate(R.layout.input_dialog, null);
129        final EditText edtInput = (EditText) linearLayout.findViewById(R.id.edtInput);
130        edtInput.setHint(hint);
131        edtInput.setText(defValue);
132        builder.setView(linearLayout);
133
134        // Set up the buttons
135        builder.setPositiveButton("OK", new DialogInterface.OnClickListener() {
136            @Override
137            public void onClick(DialogInterface dialog, int which) {
138                if (callback != null) {
139                    callback.execute(edtInput.getText().toString());
140                }
141            }
142        });
143        builder.setNegativeButton("Cancelar", null);
144
145        builder.show();
146    }
147
148    public void selectPopup(String title, final DialogCallback callback){
149        AlertDialog.Builder builder = new AlertDialog.Builder(context);
150        builder.setTitle(title);
151
152        // Set up the input
153        LayoutInflator li = (LayoutInflator) context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
154        LinearLayout linearLayout = (LinearLayout) li.inflate(R.layout.select_popup, null);
155
156        final Spinner spinnerItens = (Spinner) linearLayout.findViewById(R.id.spinnerItens);
157        final EditText edtInput = (EditText) linearLayout.findViewById(R.id.edtInput);
158        builder.setView(linearLayout);
159
160        // Set up the buttons
161        builder.setPositiveButton("OK", new DialogInterface.OnClickListener() {
162            @Override
163            public void onClick(DialogInterface dialog, int which) {
164                if (callback != null) {
165                    callback.execute(edtInput.getText().toString());
166                }
167            }
168        });
169        builder.setNegativeButton("Cancelar", null);
170
171        builder.show();
172    }
173
174    public interface DialogCallback {
175        public void execute(String text);
176    }
}

```

Código 36: Utils.java

## 4.6 Banco de Dados

```

1 DROP TABLE endereco CASCADE;
2 DROP TABLE funcionario CASCADE;

```

```

3 | DROP TABLE mesa CASCADE;
4 | DROP TABLE comanda CASCADE;
5 | DROP TABLE produto CASCADE;
6 | DROP TABLE pedido CASCADE;
7 | DROP TABLE item CASCADE;
8 | DROP TABLE ingrediente CASCADE;
9 | DROP TABLE fornecedor CASCADE;
10| DROP TABLE compra CASCADE;
11|
12| CREATE TABLE endereco (
13|   id serial,
14|   logradouro varchar(128),
15|   rua varchar(255),
16|   numero integer,
17|   bairro varchar(128),
18|   cidade varchar(128),
19|   estado varchar(128),
20|   CEP varchar(32),
21|   CONSTRAINT pk_endereco PRIMARY KEY (id)
22| );
23|
24| CREATE TABLE funcionario (
25|   id serial,
26|   tipo integer,
27|   cpf varchar(20),
28|   nome varchar(128),
29|   telefone varchar(20),
30|   id_endereco integer,
31|   nome_usuario varchar(64),
32|   CONSTRAINT pk_funcionario PRIMARY KEY (id),
33|   CONSTRAINT fk_id_endereco FOREIGN KEY (id_endereco) REFERENCES endereco (id),
34|   CONSTRAINT unique_cpf UNIQUE (cpf)
35| );
36|
37| CREATE TABLE mesa (
38|   numero integer,
39|   CONSTRAINT pk_mesa PRIMARY KEY (numero)
40| );
41|
42| --uma comanda esta sempre associada a uma mesa
43| CREATE TABLE comanda (
44|   id serial,
45|   data timestamp,
46|   numero_mesa integer,
47|   CONSTRAINT pk_comanda PRIMARY KEY (id),
48|   CONSTRAINT fk_comanda_mesa FOREIGN KEY (numero_mesa) REFERENCES mesa (numero)
49| );
50|
51| CREATE TABLE produto (
52|   id serial,
53|   nome varchar(128),
54|   preco float,
55|   CONSTRAINT pk_produto PRIMARY KEY (id)
56| );
57|
58| --um pedido pode estar em apenas uma comanda, e possui apenas um produto.
59| CREATE TABLE pedido (
60|   id serial,
61|   quantidade integer,
62|   entrega boolean,
63|   id_produto integer,
64|   id_comanda integer,
65|   CONSTRAINT pk_pedido PRIMARY KEY (id),
66|   CONSTRAINT fk_pedido_comanda FOREIGN KEY (id_comanda) REFERENCES comanda (id),
67|   CONSTRAINT fk_pedido_produto FOREIGN KEY (id_produto) REFERENCES produto (id)
68| );
69|
70| CREATE TABLE item (
71|   id serial,
72|   nome varchar(128),
73|   quantidade integer,
74|   limite integer,
75|   CONSTRAINT pk_item PRIMARY KEY (id)
76| );
77|
78| CREATE TABLE ingrediente (
79|   id_item integer,
80|   id_produto integer,
81|   quantidade integer,
82|   CONSTRAINT pk_item_produto PRIMARY KEY (id_item, id_produto),
83|   CONSTRAINT fk_item FOREIGN KEY (id_item) REFERENCES item (id),
84|   CONSTRAINT fk_produto FOREIGN KEY (id_produto) REFERENCES produto (id)
85| );
86|
87| CREATE TABLE fornecedor (
88|   id serial,
89|   nome varchar(128),
90|   telefone varchar(20),
91|   cnpj varchar(25),
92|   email varchar(128),
93|   id_endereco integer,
94|   CONSTRAINT pk_fornecedor PRIMARY KEY (id),
95|   CONSTRAINT fk_id_endereco FOREIGN KEY (id_endereco) REFERENCES endereco (id),
96|   CONSTRAINT unique_cnpj UNIQUE (cnpj)

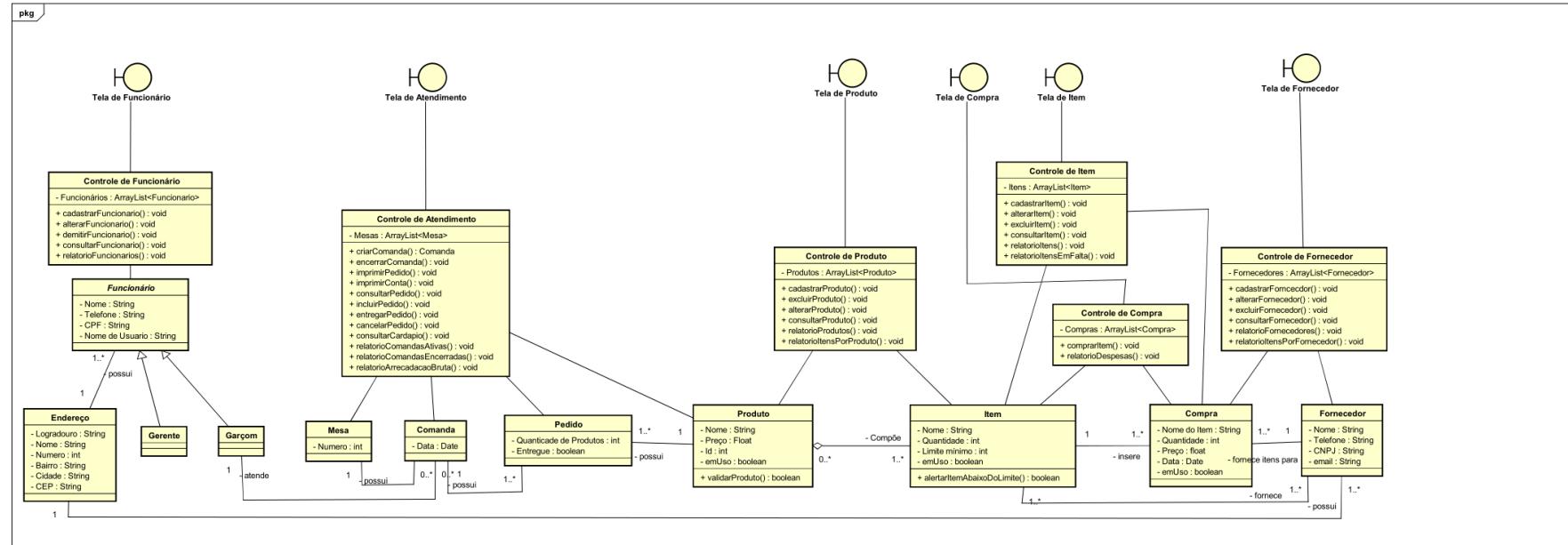
```

```
97 );
98 CREATE TABLE compra (
99   id serial,
100   id_item integer,
101   id_fornecedor integer,
102   quantidade integer,
103   preco float,
104   data timestamp,
105   CONSTRAINT pk_compra PRIMARY KEY (id),
106   CONSTRAINT fk_compra_item FOREIGN KEY (id_item) REFERENCES item (id),
107   CONSTRAINT fk_compra_fornecedor FOREIGN KEY (id_fornecedor) REFERENCES fornecedor (id)
108 );
109 );
```

Código 37: CriarBanco.sql

# 5 Diagramas

## 5.1 Diagrama de Classes



## 5.2 Diagrama de Casos de Uso

